Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

DISEÑO DE UN PUNTO DE ACCESO BLUETOOTH PARA EL ENVÍO DE DATOS BAJO DEMANDA Y POR DIFUSIÓN SELECTIVA

Steve Gerardo Morales García
Asesorado por el Ing. PhD. Enrique Edmundo Ruiz Carballo

Guatemala, mayo de 2012

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO DE UN PUNTO DE ACCESO BLUETOOTH PARA EL ENVÍO DE DATOS BAJO DEMANDA Y POR DIFUSIÓN SELECTIVA

TRABAJO DE GRADUACIÓN

PRESENTADO A JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

STEVE GERARDO MORALES GARCÍA
ASESORADO POR EL ING. PHD. ENRIQUE EDMUNDO RUIZ CARBALLO

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO ELECTRÓNICO

GUATEMALA, MAYO DE 2012

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Juan Carlos Molina Jiménez
VOCAL V	Br. Mario Maldonado Muralles
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Carlos Eduardo Guzmán Salazar
EXAMINADOR	Ing. Romeo Neftalí López Orozco
EXAMINADOR	Ing. Luis Eduardo Durán Córdoba
SECRETARIA	Inga. Marcia Ivónne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO DE UN PUNTO DE ACCESO BLUETOOTH PARA EL ENVÍO DE DATOS BAJO DEMANDA Y POR DIFUSIÓN SELECTIVA

Tema que me fuera asignado por la Coordinación de Electrónica de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 14 de julio de 2009.

Steve Gerardo Morales García

Guatemala 22 de Febrero del 2011

Ingeniero Carlos Guzmán
Coordinador del Área de Electrónica
Presente

Ingeniero Guzmán:

Por medio de la presente me permito informarle que he revisado el Trabajo de Graduación del estudiante Steve Gerardo Morales García carné 2002-12119, titulado " Diseño de un punto de acceso Bluetooth para el envío de datos bajo demanda y por difusión selectiva". Habiendo llenado la misma los requisitos necesarios me permito darle mi aprobación.

Atentamente

Enrique E Ruiz Carballo
Ingeniero

Enrique E Ruiz C.
INGENIERO ELECTRICISTA
COL No 2219

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERIA

Ref. EIME 58 2011
Guatemala, 5 de SEPTIEMBRE 2011.

Señor Director
Ing. Guillermo Antonio Puente Romero
Escuela de Ingeniería Mecànica Elèctrica
Facultad de Ingeniería, USAC.

Señor Director:

Me permito dar aprobación al trabajo de Graduación titulado: DISEÑO DE UN PUNTO DE ACCESO BLUETOOTH PARA EL ENVÍO DE DATOS BAJO DEMANDA Y POR DIFUSIÓN SELECTIVA. del estudiante STEVE GERARDO MORALES GARCÍA, que cumple con los requisitos establecidos para tal fin.

Sin otro particular, aprovecho la oportunidad para saludarle.

Atentamente,
ID Y ENSEÑAD A TODOS

Ing. Carlos Eduardo Guzmán Salazar
Coordinador de Electrónica

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

CEGS/sro

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

REF. EIME 66. 2011.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de Área, al trabajo de Graduación del estudiante; STEVE GERARDO MORALES GARCÍA titulado: DISEÑO DE UN PUNTO DE ACCESO BLUETOOTH PARA EL ENVÍO DE DATOS BAJO DEMANDA Y POR DIFUSIÓN SELECTIVA, procede a la autorización del mismo.

Ing. Guillermo Antonio Puente Romero

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

GUATEMALA, 17 DE OCTUBRE 2,011.

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref. DTG.244-2012

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al trabajo de graduación titulado: **DISEÑO DE UN PUNTO DE ACCESO BLUETOOTH PARA EL ENVÍO DE DATOS BAJO DEMANDA Y POR DIFUSIÓN SELECTIVA,** presentado por el estudiante universitario **Steve Gerardo Morales García,** autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Murphy Olympo Paiz Recinos
Decano

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, mayo de 2012

/cc

ACTO QUE DEDICO A:

Dios	Por ser la fuente de sabiduría que me ha permitido alcanzar las metas propuestas en la vida.
Mis padres Lucrecia y Hector	Cuyo amor y apoyo incondicional me han dado la confianza y la oportunidad de salir adelante.
Mi abuela Lidia (q.e.p.d.)	Por su amor y sus sabios consejos de vida.
Mis hermanos Hector, Marck y Wendy	Por estar presentes en cada una de las etapas de mi vida brindándome todo su apoyo.
Mis amigos	Por su constante motivación y compañerismo. Cada uno de ustedes ha sido importante en todo momento.

AGRADECIMIENTOS A:

Mis padres	Por todo el apoyo brindado en los buenos y malos momentos de mi carrera y de la vida.
Mis hermanos	Por ser un ejemplo de vida, por sus consejos y por compartir siempre juntos en familia.
Mi asesor	Por su valiosa colaboración y tiempo dedicado a la revisión de este trabajo de graduación.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
dB	Decibeles
dBm	Decibeles con referencia de una milésima de Watt
GHz	Giga Hertz
°C	Grados Celsius
+	Más
MHz	Mega Hertz
µs	Microsegundos (10^{-6})
kbps o Kbit/s	Miles de bits por segundo
mm	Milímetros (10^{-3})
ms	Milisegundos (10^{-3})
mW	Miliwatt

Mbps o Mbit/s	Millones de bits por segundo
π	PI, tiene un valor aproximado de 3,1416
x	Por
%	Por ciento
V	Voltios
W	Watt

GLOSARIO

AD-HOC En redes de comunicación se refiere a una red en la que no hay un nodo central, sino que todos los dispositivos están en igualdad de condiciones.

Adaptador En informática, un adaptador es un dispositivo de *hardware* que convierte datos transmitidos en un formato a otro.

ALU Unidad Aritmética Lógica. Es un circuito digital que calcula operaciones aritméticas y operaciones lógicas entre dos números.

Banda base En telecomunicaciones, el término banda base se refiere a la banda de frecuencias que no se adapta al medio por el que se va a transmitir.

Banda ISM Son bandas reservadas internacionalmente para uso no comercial de radiofrecuencia electromagnética en áreas industriales, científicas y médicas.

Calidad de servicio Son especificaciones que garantizan la transmisión de cierta cantidad de información en un tiempo dado.

Datagrama	Es un fragmento de paquete que es enviado con la suficiente información como para que la red pueda simplemente encaminar el fragmento hacia la terminal de datos.
Demodulación	Técnicas utilizadas para recuperar la información transportada por una onda portadora, que en el extremo transmisor había sido modulada con dicha información.
DNS	El sistema de nombre de dominio traduce nombres inteligibles para los humanos en identificadores binarios asociados con los equipos conectados a la red.
DPSK	La modulación diferencial por desplazamiento de fase, es una forma de modulación angular que consiste en hacer variar la fase portadora entre un número de valores discretos.
DQPSK	Modulación diferencial de desplazamiento de fase de 4 símbolos, separados entre sí por un ángulo de 90^{o}.
Encriptado	Es el proceso para volver ilegible información considerada importante. La información una vez encriptada solo se puede leer o interpretar si se tiene la clave.

Ethernet	Es un estándar de redes de computadoras de área local. Define las características de cableado y señalización de nivel físico y los formatos de tramas de datos del nivel de enlace de datos del modelo OSI.
Filtro Gaussiano	Es un filtro cuya respuesta a un impulso es una función Gaussiana.
Firmware	Es un bloque de instrucciones de programa para propósitos específicos, grabado en una memoria de tipo no volátil. Establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo.
Frecuencia	Es una magnitud que mide el número de repeticiones por unidad de tiempo de cualquier fenómeno o suceso periódico. La frecuencia tiene una relación inversa con el concepto de longitud de onda.
FTP	Es un protocolo de transferencia de ficheros entre sistemas conectados a una red TCP basado en la arquitectura cliente-servidor, de manera que desde un equipo cliente se puede conectar a un servidor para descargar ficheros desde él, o para enviarle los propios archivos independientemente del Sistema Operativo utilizado en cada equipo.

GFSK	Modulación por desplazamiento de frecuencia Gaussiana. Es un tipo de modulación donde un 1 lógico es representado mediante una desviación positiva de la frecuencia de la onda portadora y un 0 mediante una desviación negativa de la misma.
HTTP	Es el protocolo usado en cada transacción de la World Wide Web. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.
Host	Es una computadora que se encuentra conectada a cualquier tipo de red de datos.
Interfaz	Dispositivo a través del que se envían o reciben señales desde un sistema hacia otros.
Interoperabilidad	Es la condición mediante la cual sistemas heterogéneos pueden intercambiar procesos o datos.
Led	Es un dispositivos semiconductor que emite luz incoherente de espectro reducido cuando se polariza de forma directa.

Mapeo	Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.
Micro-ops	Se refiere a instrucciones de bajo nivel que son usadas en algunos diseños para implementar instrucciones de máquinas más complejas.
Microcontrolador	Circuito integrado que incluye en su interior las tres unidades funcionales de una computadora: unidad central de procesamiento, memoria y unidades de entrada y salida.
Microprocesador	Circuito integrado que incorpora en su interior una unidad de proceso y todo un conjunto de elementos lógicos que permiten enlazar otros dispositivos como memorias y puertos de entrada y salida.
Multiplataforma	Es un término usado para referirse a los programas, Sistemas Operativos, lenguajes de programación u otra clase de *software*, que puedan funcionar en diversas plataformas.

Multiplexor	En telecomunicaciones el multiplexor se utiliza como dispositivo que puede recibir varias entradas y transmitirlas por un medio compartido. Para ello, lo que hace es dividir el medio de transmisión en múltiples canales.
Payload	El *payload* es la información que es transmitida a través de una red de datos y transportada por los encabezados en un paquete de datos.
Protocolo	Conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red.
Puerto serial	Interfaz de comunicaciones de datos digitales, frecuentemente utilizado por computadoras y periféricos, en donde la información es transmitida bit a bit enviando un solo bit a la vez.
RF	El término radiofrecuencia, también denominado espectro de radiofrecuencia o RF, se aplica a la porción del espectro electromagnético en el que se pueden generar ondas electromagnéticas aplicando corriente alterna a una antena.

Ruteador	Es un dispositivo de *hardware* para interconexión de red de ordenadores que opera en la capa tres del modelo OSI. Permite asegurar el enrutamiento de paquetes entre redes o determinar la ruta que debe tomar el paquete de datos.
Script	Es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano. Casi siempre son interpretados.
Socket	Designa un concepto abstracto por el cual dos programas situados en computadoras distintas pueden intercambiar cualquier flujo de datos.
Subred	Una subred es un rango de direcciones lógicas. Cuando una red de computadoras se vuelve muy grande, conviene dividirla en subredes para reducir el tamaño de los dominios de *broadcast* y hacer la red más manejable.
TCP/IP	Se utiliza para enlazar computadoras que utilizan diferentes Sistemas Operativos, incluyendo PC, minicomputadoras y computadoras centrales sobre redes de área local (LAN) y área extensa (WAN).

TDD	División de tiempo dúplex es un sistema capaz de mantener una comunicación bidireccional, enviando y recibiendo mensajes de forma simultánea.
Tipado	Se refiere a la capacidad de una variable para tomar valores de distinto tipo en cualquier momento dentro de cierto lenguaje de programación.
Trailer	En un paquete de datos que generalmente contiene información para verificar errores que puedan ocurrir durante el proceso de transmisión.
Transceptor	Es un dispositivo que realiza, dentro de una misma caja o chasis, funciones tanto de transmisión como de recepción, utilizando componentes de circuito comunes para ambas funciones.

RESUMEN

Una de las estrategias comúnmente usadas de mercadeo al momento de realizar una campaña es la de publicar información en puntos específicos en un área seleccionada. El procedimiento es sencillo: se escoge un lugar estratégico, se diseña el medio gráfico con la información a presentar y se contrata una empresa que se encargará de presentarla a las personas que pasen por el lugar.

Como una alternativa a este tipo de estrategias se pretende proponer un diseño que pueda focalizar contenidos audiovisuales con información relevante y de interés para pequeños grupos de personas que puedan estar ubicados en diferentes puntos geográficos. Tomando en cuenta que los teléfonos celulares son los dispositivos electrónicos portátiles más utilizados por las personas y de estos un alto porcentaje son de gama media y alta; se pretende proponer el diseño de un dispositivo capaz de interactuar con los celulares a través de la tecnología inalámbrica *Bluetooth*.

El diseño del punto de acceso *Bluetooth* a proponer cumplirá con la característica de enviar información directamente a los dispositivo cercanos; ya sea por solicitud directa del usuario o enviando periódicamente información a los dispositivos dentro del área de cobertura. Debido a que esta tecnología se caracteriza por su bajo consumo de energía, se desea que el diseño siga estos lineamientos.

OBJETIVOS

General

Proponer el diseño para un punto de acceso de tecnología *Bluetooth* para establecer comunicación con dispositivos portátiles.

Específicos

1. Describir las características y conceptos principales sobre la tecnología *Bluetooth*.

2. Definir los conceptos esenciales para la programación de recursos *Bluetooth* a través de un lenguaje de programación.

3. Diseñar un sistema de comunicación confiable, de fácil implementación y mantenimiento.

4. Utilizar *software* libre y *hardware* básico para que sea fácil de implementar.

INTRODUCCIÓN

La comunicación inalámbrica ha estado presente desde finales del siglo IXX y se ha hecho presente en la radio, infrarojos, televisión y recientemente en el protocolo 802.11. Lo que distingue a la teconología *Bluetooth* es su corta distancia de comunicación, usualmente menor a 100 metros; su *hardware y software* son afectados por esta especial característica.

Bluetooth es una especificación industrial para redes inalámbricas de área personal que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia segura y globalmente libre (2,4 GHz). Los principales objetivos que se pretenden conseguir con esta norma son:

- Facilitar las comunicaciones entre equipos móviles y fijos
- Eliminar cables y conectores entre estos
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre los equipos personales.

Los dispositivos que con mayor frecuencia utilizan esta tecnología son los de los sectores de las telecomunicaciones y la informática personal, como PDAs, teléfonos móviles, computadoras portátiles, ordenadores personales, impresoras y cámaras digitales.

Bluetooth es una tecnología de comunicaciones diseñado especialmente para dispositivos de bajo consumo, con corto alcance de comunicación y su fabricación con base en transceptores de bajo costo.

Debido a esta tecnología, los dispositivos que la poseen pueden comunicarse entre ellos sin la necesidad de componentes físicos adicionales. Los enlaces se realizan por radiofrecuencia de forma que los dispositivos no tienen por qué estar alineados, pueden incluso estar en habitaciones separadas si la potencia de transmisión cumple con las características necesarias.

1. TECNOLOGÍA *BLUETOOTH*

1.1. Generalidades

Bluetooth es un estándar de comunicación inalámbrico que utiliza señales de radiofrecuencia de corto alcance el cual fue creado con el fin de eliminar el uso de cables entre dispositivos, para la transferencia de datos a través de redes de área personal.

Las principales características de esta tecnología son poseer un bajo consumo de potencia y un bajo costo de implementación. El funcionamiento de *Bluetooth* depende principalmente de las señales de radiofrecuencia, banda base y de la pila de protocolos.

1.2. Redes *Bluetooth*

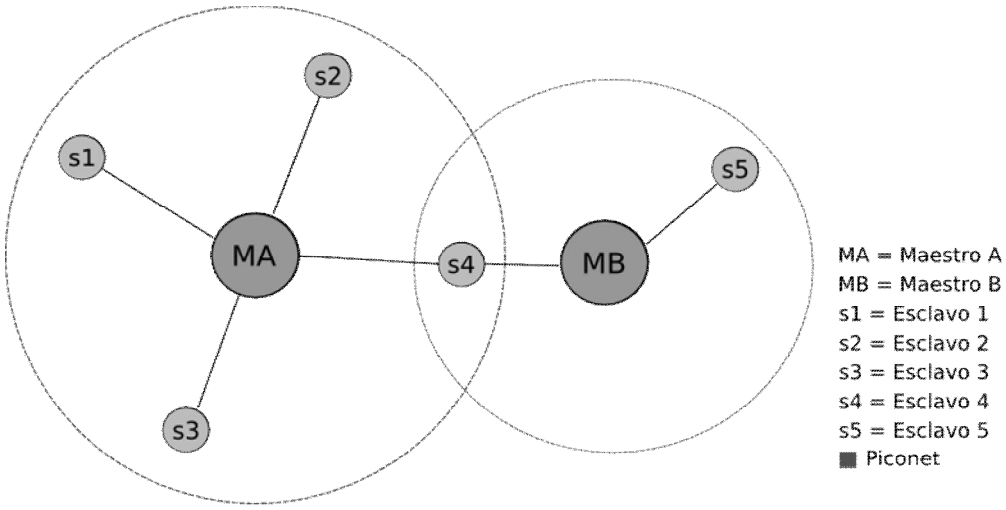
Una red de dispositivos que estén conectados por medio de *Bluetooth* generalmente es conocida como una red de área personal.

La tecnología *Bluetooth* soporta conexiones punto a punto y punto a multipunto asumiendo que solamente un pequeño grupo de unidades estarán conectados en un tiempo dado.

Dentro de una red, un dispositivo puede comportarse como un dispositivo maestro o un dispositivo esclavo, siendo la única diferencia entre ellos, que el dispositivo maestro es quien iniciará la conexión.

Las redes *Bluetooth* son llamadas *piconets*. Estas constan de un dispositivo maestro y pueden tener hasta siete dispositivos esclavos activos al mismo tiempo. Si una unidad forma parte de varias *piconets,* ésta puede utilizarse como un puente que permita unir las diferentes redes que conforma, de esta forma se crea una *scatternet*.

Figura 1. **Creación de una red scatternet**



Fuente: BAKKER, D.M.; MCMICHAEL, Diane. *Bluetooth end to end*. p. 10.

Bluetooth permite una red *Ad-Hoc*. Esto quiere decir, que puede ser creada de forma espontanea ya que no necesita una estructura formal que mantenga la conexión de red y de alguna manera se encuentra limitada de forma temporal y espacial. En la figura 1 se muestra la creación de una red tipo scatternet a través de la unidad s4.

1.3. Arquitectura

El sistema *Bluetooth* ofrece servicios que habilitan conexiones entre dispositivos y permiten el intercambio de datos entre ellos. El sistema principal consiste en diferentes partes básicas; un transceptor de radiofrecuencia para la transmisión y recepción de señales, un sistema de banda base que establece los procedimientos para establecer un enlace físico y una pila de protocolo que permite la interoperabilidad entre dispositivos.

1.3.1. Radiofrecuencia

Los dispositivos *Bluetooth* operan en la banda de frecuencia de 2,4 GHz también conocida como banda ISM (*Industrial Scientific Medical*, por sus siglas en inglés). La banda ISM es compartida con otros dispositivos, como teléfonos inalámbricos, monitores de bebés, *Internet* inalámbrico y otros servicios de uso común.

Bluetooth utiliza baja potencia para el envío de las señales de RF. En el mercado existen tres diferentes clases de radios los cuales proveen a cada dispositivo *Bluetooth* con una potencia de transmisión particular la cuál determinará el alcance de la señal RF emitida.

Tabla I. **Características de los radios *Bluetooth* en el mercado**

Power class	Potencia máxima	Potencia mínima	Rango (aproximado)
1	100 mW (20 dBm)	1 mW (0 dBm)	~100 metros
2	2,5 mW (4 dBm)	0,25 mW (-6 dBm)	~10 metros
3	1 mW (0 dBm)	N/A	~1 metro

Fuente: <http://es.wikipedia.org/wiki/Bluetooth>. [Consulta: mayo de 2010].

1.3.1.1. Espectro ensanchado

Para evitar interferencia entre dispositivos que operan dentro de la misma banda de frecuencia, *Bluetooth* utiliza la técnica llamada espectro ensanchado por salto de frecuencia (*Frequency Hopping Spread Spectrum*, FHSS por sus siglas en inglés). Básicamente, el ancho de banda se divide en 79 canales emitidos sobre una serie de radiofrecuencias aparentemente aleatorias.

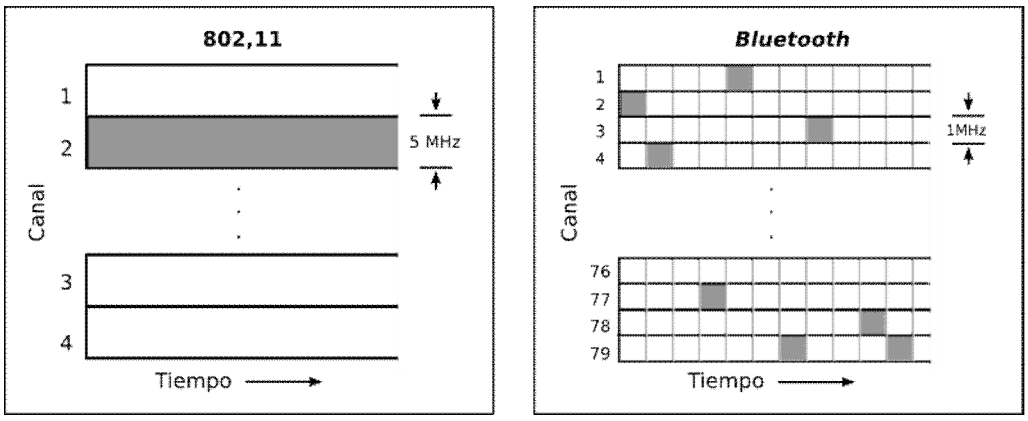
Tabla II. **Rango de frecuencias del espectro radioeléctrico**

Rango reglamentario	Banda inferior de guarda	Banda superior de guarda	Canales RF
2 400 - 2 4835 GHz	2 MHz	3,5 MHz	f=2 402+k MHz, k = 0,…,78

Fuente: Bluetooth SIG, *Core V2.1 + EDR*. p. 31.

En una comunicación activa entre dispositivos se experimenta un cambio de frecuencia cada 625 μs; estos cambios deben estar sincronizados para que la conexión permanezca activa.

Figura 2. **Comparación entre canales 802,11 y canales *Bluetooth***



Fuente: HUAN, Albert; RUDOLPH, Larry. *Bluetooth essentials for programmers*. p. 28.

1.3.1.2. Características de modulación

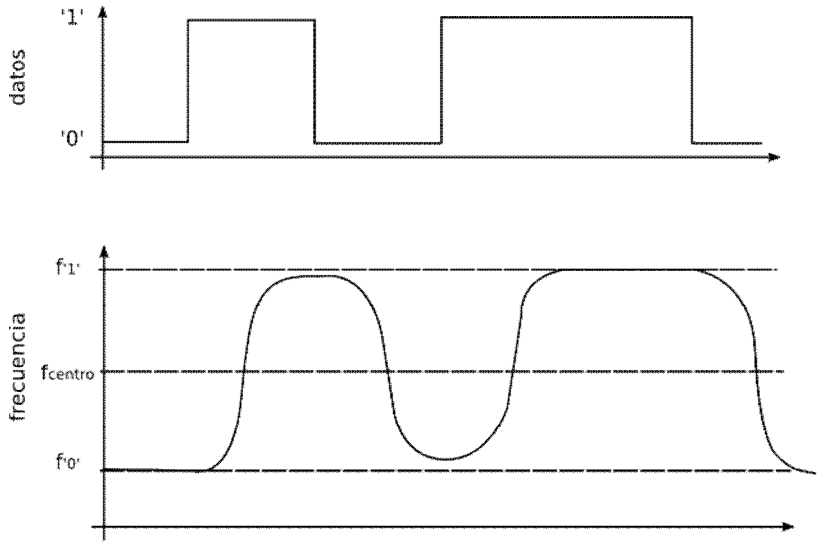
La tecnología *Bluetooth* presenta dos modos para modular las señales transmitidas; un modo obligatorio BR (*Basic Rate*) y un modo opcional EDR (*Enhanced Data Rate*).

El modo BR utiliza un tipo de modulación por desplazamiento de frecuencia Gausiana (GFSK, *Gaussian Frequency Shift Keying* por sus siglas en inglés).

GFSK es un tipo de modulación en donde el valor lógico 1 se representa mediante una desviación positiva o un incremento de la frecuencia de la onda portadora y un 0 lógico mediante una desviación negativa o disminución de la misma.

La información se pasa por un filtro Gausiano antes de ser modulada; esto proporciona un espectro de energía más estrecho de la señal modulada permitiendo mayores velocidades de transmisión sobre el mismo canal.

Figura 3. **Efectos de un filtro Gaussiano sobre la banda base**



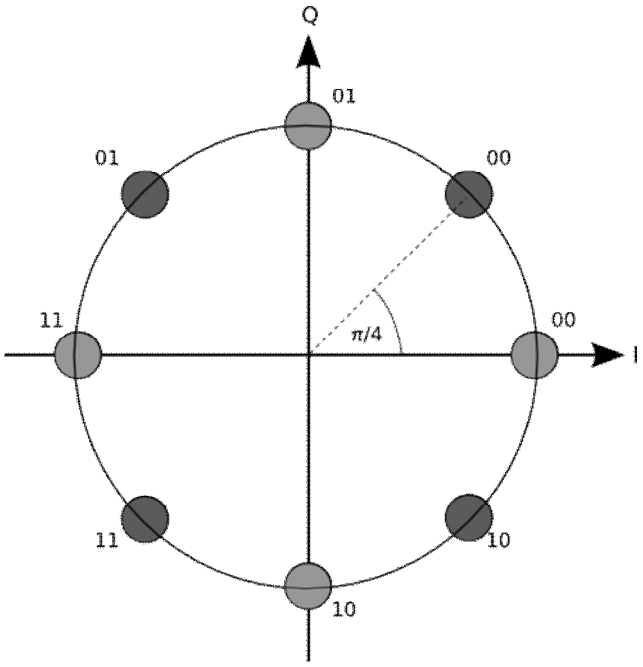
Fuente: <http://upload.wikimedia.org/wikipedia/commons/6/65/Filtro_gausiano_para_datos.jpg>. [Consulta: mayo de 2010].

El modo EDR transmite cierta parte del paquete de datos con un tipo de modulación por desplazamiento de fase (PSK, *Phase Shift Keying* por sus siglas en inglés). Se utilizan dos variantes para modular en PSK, π/4-DQPSK y 8DPSK.

La modulación PSK consiste en hacer variar la fase de la portadora entre un número de valores discretos. Mientras mayor sea la cantidad de variaciones de fase permitidas mayor cantidad de información se puede transmitir utilizando el mismo ancho de banda, pero, también aumentará la sensibilidad frente a ruidos e interferencias.

Para transmitir datos a 2 Mbps se utiliza la modulación π/4-DQPSK. Con este tipo de modulación se obtienen 2 constelaciones iguales desplazadas π/4 radianes entre ellas y cada una con un desplazamiento de fase de 4 símbolos, desplazados 90º entre sí. El PSK diferencial no necesita recuperar la señal portadora para realizar la demodulación debido que la información no está contenida en la fase absoluta, sino en las transiciones.

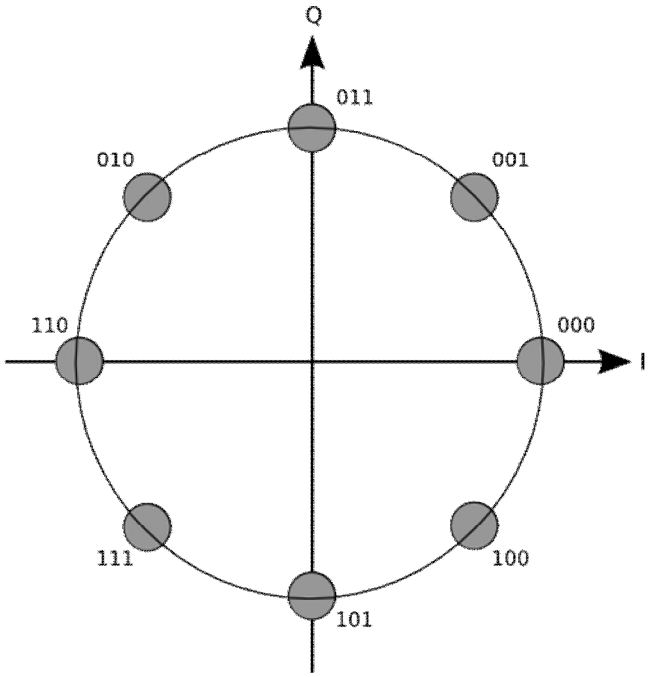
Figura 4. **Diagrama de constelación para modulación π/4-DQPSK**



Fuente: <http://en.wikipedia.org/wiki/Phase-shift_keying>. [Consulta: junio de 2010].

Para transmitir datos a 3 Mbps se utiliza la otra variación de PSK, la modulación 8DPSK. Esta modulación utiliza un desplazamiento de fase diferencial de 8 símbolos permitiendo transmitir mayor información en el mismo ancho de banda.

Figura 5. **Diagrama de constelación para modulación 8DPSK**



Fuente: <http://en.wikipedia.org/wiki/Phase-shift_keying>. [Consulta: junio de 2010].

1.3.2. Banda base

La capa física del sistema *Bluetooth* posee una parte que se encarga de implementar el acceso de medios y los procedimientos que se establecen entre dispositivos, esta parte del sistema se denomina banda base y es conformada por el *hardware* del sistema, generalmente un microcontrolador.

La banda base posee ciertas características físicas que permiten procesar las señales recibidas a través del bloque RF del sistema. Los protocolos de la banda base se ejecutan por medio del controlador de enlaces (*Link Controller*).

Para poder establecer una comunicación entre dispositivos, se debe definir un medio físico de transporte entre ellos. La tecnología *Bluetooth* provee a través de ciertas características físicas de la banda base los recursos para establecer el medio de transporte.

1.3.2.1. Canales físicos

Los canales físicos están definidos por una secuencia pseudo aleatoria de saltos en los canales RF, el tiempo de duración de un *slot* y un código de acceso.

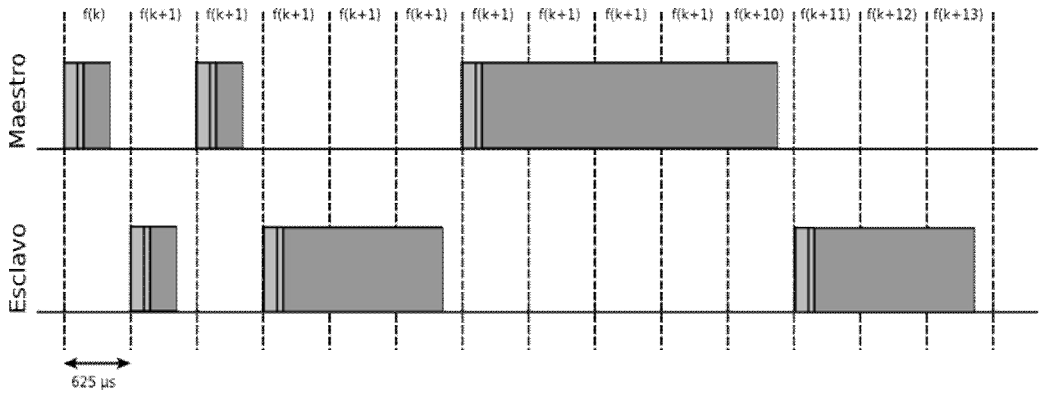
Un dispositivo que se encuentra en una piconet compartirá un canal físico solamente si el transceptor este sintonizado a la misma frecuencia en el mismo instante de tiempo, que se tenga el mismo código de acceso y que los dispositivos se encuentren a una distancia nominal uno del otro.

Dentro del sistema *Bluetooth* se definen cierto número de canales físicos, cada uno es optimizado y utilizado para diferente propósito. Los siguientes canales físicos se encuentran definidos:

- Canal físico básico de la piconet
- Canal físico adaptado de la piconet
- Canal físico de detección de paginación
- Canal físico de detección de búsqueda

Los canales físicos básicos y adaptados de la piconet son los canales usados para que dos dispositivos conectados se comuniquen. Estos canales son definidos por el maestro quien es el que controla el tráfico del canal físico de la piconet mediante una secuencia de consultas. El canal básico se encuentra dividido en *slots* de 625 µs cada uno. El canal adaptado puede ser usado en dispositivos conectados con la función de salto adaptable de frecuencia activa; se diferencia del canal físico básico en el mecanismo del canal y que este canal puede no utilizar las 79 frecuencias que utiliza un canal básico.

Figura 6. **Paquetes de datos transmitidos en un canal básico**



Fuente: Bluetooth SIG, *Core V2.1 + EDR*. p. 66.

El canal de detección de paginación se encarga de realizar la conexión entre dispositivos mientras el canal de detección de búsqueda se encarga de descubrirlos dentro de la piconet. Estos canales siguen una secuencia de saltos más lenta que el canal físico básico.

Un dispositivo puede utilizar solamente uno de estos canales a la vez. Para que puedan ser soportadas operaciones múltiples se utiliza el esquema de comunicación TDD (*Time Division Duplex*, por sus siglas en inglés); de esta forma un dispositivo puede aparentar operar en varias piconets simultáneamente.

1.3.2.1.1. Selección de salto

En total *Bluetooth* tiene seis secuencias de salto definidas, de las cuales cinco son utilizadas por el sistema básico de salto y una por el sistema adaptado, las secuencias definidas son:

- Secuencia de paginación
- Secuencia de respuesta a la paginación
- Secuencia de búsqueda
- Secuencia de respuesta a la búsqueda
- Secuencia básica de saltos de canal
- Secuencia de saltos de canal adaptada

Las secuencias de paginación y búsqueda distribuyen de forma equitativa las 32 frecuencias de despertar entre los 79 MHz. Las secuencias de respuesta responden 32 frecuencias en correspondencia con las secuencias de solicitud. La secuencia básica y la adaptada no muestran algún patrón repetitivo durante un pequeño intervalo de tiempo y utilizan el mismo mecanismo de canales; la secuencia de salto adaptada puede utilizar menos de 79 frecuencias.

El esquema general de selección consiste básicamente de dos partes, seleccionar la secuencia y mapear esta secuencia en el salto de frecuencias.

1.3.2.2. Enlaces físicos

Una conexión de banda base viene siendo representada por un enlace físico. Estos enlaces están asociados únicamente a un canal físico. Un enlace físico posee propiedades comunes que pueden ser aplicadas a todas las comunicaciones lógicas del enlace. Estas propiedades son:

- Control de energía
- Supervisión de los enlaces
- Encriptado
- Cambio de velocidad de transmisión dictado por la calidad del canal
- Control de paquetes en múltiples *slots*

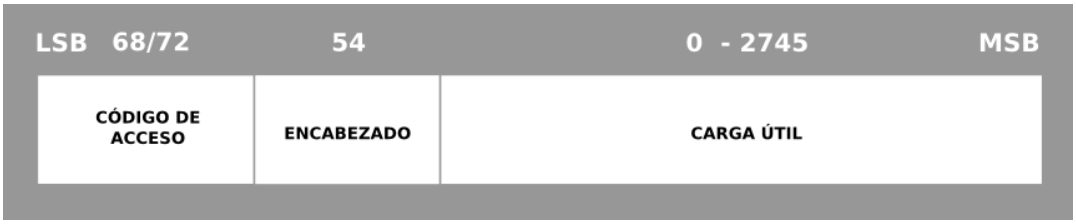
Estas características son controladas por el protocolo de gestión de enlace (*Link Manager Protocol*), lo cual permite ajustar los parámetros de un enlace físico entre dispositivos cuando resulte necesario.

1.3.2.3. Paquete de datos

Los datos en una piconet son transmitidos en paquetes. Existe cierta diferencia en la estructura de un paquete transmitido en modo BR con un paquete transmitido en modo EDR.

En el modo BR de transmisión consta de 3 partes, el código de acceso, el encabezado y la carga útil.

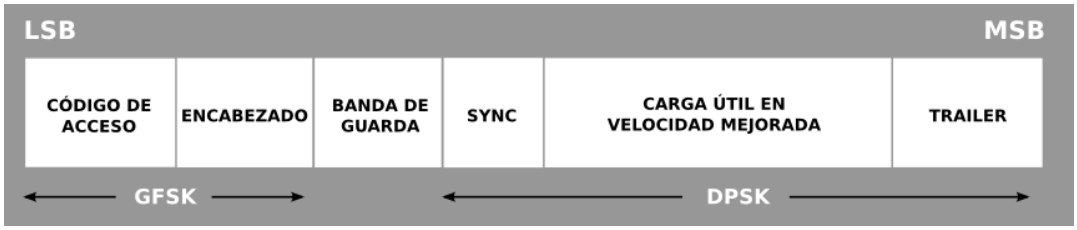
Figura 7. **Estructura de un paquete de datos en velocidad básica**



Fuente: Bluetooth SIG, *Core V2.1 + EDR.* p. 102.

El modo EDR utiliza de igual forma un código de acceso y un encabezado, pero posee una banda de guarda y un bloque de sincronización que son necesarios para realizar el intercambio de modulación entre GFSK y DPSK.

Figura 8. **Estructura de un paquete de datos en velocidad mejorada**



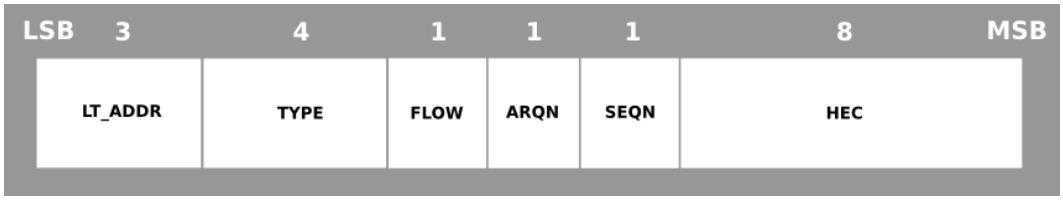
Fuente: Bluetooth SIG, *Core V2.1 + EDR.* p. 102.

El código de acceso es usado para sincronización, compensación del desfase del reloj y para identificación del paquete; este tiene una longitud de 68 si no es seguido por un encabezado y no contiene los 4 bits que conforman el *trailer* y es de 72 bits si es seguido por el código de un encabezado.

El encabezado se divide en 6 bloques:

- LT_ADDR: identifica la dirección de la comunicación lógica asignada a un dispositivo esclavo.
- TYPE: identifica el tipo de paquete enviado y la cantidad de *time slots* utilizados por el mismo.
- FLOW: controla el flujo de los paquetes
- ARQN: informa a la unidad maestra que se transmitirán los datos de la carga útil.
- SEQN: provee una numeración consecutiva para ordenar la secuencia de datos.
- HEC: chequea que el encabezado no posea errores

Figura 9. **Formato del encabezado de un paquete de datos**



Fuente: Bluetooth SIG, *Core V2.1 + EDR*. p. 109.

La carga útil (*Payload*) de un paquete de datos contiene los datos que se desean transmitir. La forma del *payload* varía dependiendo si el paquete transporta voz o datos. También varía en longitud, puede ser de 0 bits en el proceso de interrogación o para un mensaje de error y hasta de 2 745 bits para un paquete de datos.

1.3.2.3.1. Tipo de paquetes

Los paquetes usados en una piconet se relacionan con la comunicación lógica utilizada dentro de una piconet. En una piconet se utilizan comunicaciones lógicas SCO, eSCO y el ACL; y para cada uno de estos se pueden definir diferentes tipos de paquetes.

Dentro de los paquetes asociados al control de enlace se encuentran los paquetes ID, NULL, POLL, FHS. Todos estos paquetes, a excepción del paquete FHS, no tienen información de carga útil, no son codificados y tampoco poseen bits CRC.

Tabla III. **Paquetes de control de enlace**

Tipo	Carga útil	FEC	CRC	Descripción
ID	N/A	N/A	N/A	Consiste en el código de acceso del dispositivo (DAC) o en código de acceso de interrogación (IAC). Posee una longitud fija de 68 bits.
NULL	N/A	N/A	N/A	Puede ser usado para regresar información del enlace, como el bit de indicación de reconocimiento (ARQN) o el bit de control de flujo (FLOW). Tiene una longitud de 126 bits.
POLL	N/A	N/A	N/A	Puede ser usado por el maestro para sondear a los esclavos. Los esclavos deben responder con un paquete aunque no se tenga información que enviar.
FHS	18 bytes	2/3	16 bits	Paquete de control que contiene la dirección del dispositivo Bluetooth y el reloj de quien envia el paquete.

Fuente: Bluetooth SIG, *Core V2.1 + EDR*. p. 127.

Los paquetes ACL se transfieren por medio de las comunicaciones lógicas asíncronos. Pueden transferir datos del usuario o datos de control.

Se definen siete tipos de paquetes para el modo de velocidad básica: DM1, DH1, DM3, DH3, DM5, DH5 y AUX1. Adicionalmente se encuentran definidos seis paquetes para el modo de velocidad mejorada: 2-DH1, 3-DH1, 2-DH3, 3-DH3, 2-DH5 y 3-DH5.

Tabla IV. **Paquetes ACL**

Tipo	Carga útil de encabezado (bytes)	Carga útil de usuario (bytes)	FEC	CRC (bits)	Descripción
DM1	1	0-17	2/3	16	Transporta únicamente información de datos en 1 *time slot*.
DH1	1	0-27	N/A	16	Transporta información de datos sin codificación FEC en 1 *time slot*.
DM3	2	0-121	2/3	16	Trasporta información y puede ocupar 3 *time slots*.
DH3	2	0-183	N/A	16	Transporta información de datos sin codificación FEC y puede ocupar 3 *times slot*.
DM5	2	0-224	2/3	16	Paquete de información que puede llegar a ocupar 5 *time slots*.
DH5	2	0-339	N/A	16	Paquete de información sin codificación FEC que puede llegar a ocupar 5 *time slots*.
AUX1	1	0-29	N/A	N/A	Este paquete se asemeja al DH1 pero sin código CRC. Este puede ser descartado y no debe ser usado en enlaces lógicos ACL-U o ACL-C.
2-DH1	2	0-54	N/A	16	Paquete usado en modo EDR y modulado usando π/4-DQPSK. Similar al paquete DH1.
2-DH3	2	0-367	N/A	16	Paquete usado en modo EDR y modulado usando π/4-DQPSK. Similar al paquete DH3.
2-DH5	2	0-679	N/A	16	Paquete usado en modo EDR y modulado usando π/4-DQPSK. Similar al paquete DH5.
3-DH1	2	0-83	N/A	16	Paquete usado en modo EDR y modulado usando 8DPSK. Similar al paquete DH1.
3-DH3	2	0-552	N/A	16	Paquete usado en modo EDR y modulado usando 8DPSK. Similar al paquete DH3.
3-DH5	2	0-1 021	N/A	16	Paquete usado en modo EDR y modulado usando 8DPSK. Similar al paquete DH5.

Fuente: Bluetooth SIG, *Core V2.1 + EDR.* p. 127.

Los paquetes HV y DV son usados en la comunicación lógica síncrona SCO. Los paquetes HV no deben ser retransmitidos; en los paquetes DV la sección de datos debe ser retransmitida. Estos paquetes son usualmente usados para transmitir voz a 64 Kb/s. Los paquetes EV también son usados de forma síncrona con la comunicación lógica eSCO; estos paquetes son usados de igual forma para transmitir audio a 64 Kb/s así como datos a 64 Kb/s o a otras velocidades. Los paquetes eSCO deben ser retransmitidos si no reciben la señal de reconocimiento dentro de la ventana de retransmisión.

Tabla V. **Paquetes síncronos**

Tipo	Carga útil de encabezado (bytes)	Carga útil de usuario (bytes)	FEC	CRC (bits)	Descripción
HV1	N/A	10	1/3	N/A	Paquete SCO de 10 bytes de información.
HV2	N/A	20	2/3	N/A	Paquete SCO de 20 butes de información
HV3	N/A	30	N/A	N/A	Paquete SCO de 30 bytes de información.
DV*	1 D	10+(0-9) D	2/3 D	16	Combina paquetes de voz y datos. La carga útil se dibide en 80 bits para voz y 150 bits para datos.
EV3	N/A	1-30	N/A	16	Paquete eSCO, contiene entre 1 y 30 bytes de información.
EV4	N/A	1-120	2/3	16	Paquete eSCO, contiene entre 1 y 120 bytes de información.
EV5	N/A	1-180	N/A	16	Paquete eSCO, contiene entre 1 y 180 bytes de información.
2-EV3	N/A	1-60	N/A	16	Paquete eSCO, contiene entre 1 y 60 bytes de información. Usando modulación π/4-DQPSK.
2-EV5	N/A	1-360	N/A	16	Paquete eSCO, contiene entre 1 y 360 bytes de información. Usando modulación π/4-DQPSK.
3-EV3	N/A	1-90	N/A	16	Paquete eSCO, contiene entre 1 y 90 bytes de información. Usando modulación 8DPSK.
3-EV5	N/A	1-540	N/A	16	Paquete eSCO, contiene entre 1 y 540 bytes de información. Usando modulación 8DPSK.

Fuente: Bluetooth SIG, *Core V2.1 + EDR.* p. 128.

1.3.2.4. Comunicaciones lógicas

Entre un dispositivo maestro y los esclavos que se encuentran en una piconet se establecen cinco diferentes medios de comunicación lógica.

- *Synchronous Connection-Oriented* (SCO)
- *Extended Synchronous Connection-Oriented* (eSCO)
- *Asynchronous Connection-Oriented* (ACL)
- *Active Slave Broadcast* (ASB)
- *Parked Slave Broadcast* (PSB)

Las comunicaciones lógicas síncronas son conexiones punto a punto entre el dispositivo maestro y el esclavo de una piconet. Estas conexiones soportan información que requiere estar ligada con una referencia de tiempo, como lo es la voz y otro tipo de datos. El maestro mantiene la conexión por medio de *slots* reservados y puede mantener tres enlaces SCO; un esclavo puede mantener tres enlaces SCO del mismo dispositivo maestro o dos enlaces SCO si son de diferentes maestros. Los paquetes SCO nunca son retransmitidos. La conexión eSCO soporta una ventana de retransmisión después de los slots reservados, esta ventana de retransmisión junto a los slots forman la ventana eSCO. Estos enlaces pueden considerarse como circuitos de conmutación debido a los slots de reserva.

El transporte ACL también establece una comunicación punto a punto entre esclavo y maestro y provee una conexión de paquetes conmutados entre dispositivos.

Entre un dispositivo maestro y un esclavo solamente puede existir una conexión ACL. Para asegurar la integridad de los datos los paquetes son retransmitidos.

Los transportes ASB y PSB son usados por el dispositivo maestro de la red para comunicarse con los esclavos activos o en espera dentro de la piconet, respectivamente.

1.3.2.5. Enlaces lógicos

Dentro de la tecnología *Bluetooth* se definen cinco enlaces lógicos:

- *Link Control* (LC)
- *ACL Control* (ACL-C)
- *User Asynchronous/Isochronous* (ACL-U)
- *User Synchronous* (SCO-S)
- *User Extended Synchronous* (eSCO-S)

El enlace lógico LC es mapeado en la cabecera de los paquetes, el cual es enviado en todos los paquetes, a excepción en los paquetes de identificación (*ID packet*). Este enlace transporta información de bajo nivel, como el control de flujo de la información, la caracterización de la carga útil y el requerimiento de reconocimiento.

El enlace ACL-C intercambia información de control entre el manejador de enlaces del dispositivo master y los esclavos. En este tipo de enlace se intercambian los paquetes DM1 y DV.

Un enlace ACL-U transmite datos del usuario en el protocolo L2CAP de forma síncrona y asíncrona. Estos mensajes serán transmitidos en uno o más paquetes de banda base.

Los enlaces SCO-S y eSCO-S transportan datos del usuario de forma síncrona y se envían por las comunicaciones lógicas SCO y eSCO, respectivamente.

1.4. Pila de protocolos

La tecnología *Bluetooth* consta de transceptores de radio incrustados como *hardware* y una pila de protocolos como *software* que trabajan en conjunto para lograr la creación de redes inalámbricas.

Los protocolos dentro de la pila definen la manera en la que los mensajes son enviados y codificados dentro de un enlace de datos y entre las diferentes capas del modelo *Bluetooth*. La pila *Bluetooth* se basa en estándares ya definidos para los sistemas de comunicaciones y de redes.

Bluetooth utiliza el modelo OSI (*Open Systems Interconnection*) como referencia. Este separa las diferentes funciones de una red para permitir la interoperabilidad entre el *hardware y software* de diferentes fabricantes. El modelo OSI en sí mismo no se puede considerar una arquitectura, ya que no especifica el protocolo que debe ser usado en cada capa. Este modelo ha sido referencia en el desarrollo de muchas de las tecnologías abiertas de redes. El modelo se encuentra dividido en siete capas.

Figura 10. **Capas del modelo OSI**

Fuente: <http://es.wikipedia.org/wiki/Archivo:Pila-osi-es.svg>. [Consulta: mayo de 2010].

1.4.1. Capas del protocolo *Bluetooth*

Bluetooth utiliza un sistema de capas dentro de la pila y cada una de ellas tiene su propio sistema de protocolos. La finalidad de que el sistema se encuentre dividido en capas es permitir la interoperabilidad entre dispositivos y aplicaciones que sean desarrollados con las especificaciones *Bluetooth*.

Bluetooth se divide en cuatro capas de operación de la siguiente forma:

- Capa *core* o del sistema central se encarga de las funciones principales para los dispositivos que se encuentran en una piconet.

- Capa de reemplazo de cables que permiten la creación de puertos seriales virtuales.

- Capa de control telefónico para poder configurar llamadas de datos, de voz y controlar los teléfonos móviles.

- Capa de protocolos adoptados para la interoperabilidad con otras tecnologías ya existentes.

1.4.2. Protocolo L2CAP

El protocolo L2CAP (*Logical link control and adaptation protocol*) opera sobre una comunicación lógica ACL y provee servicios a las capas superiores de la pila transmitiendo datos sobre canales lógicos denominados canales L2CAP. Estos canales son asignados a enlaces lógicos L2CAP.

Es compatible con el multiplexado de protocolos de capas superiores, la segmentación y unificación de paquetes y la comunicación de información sobre la calidad del servicio (QoS); estas características permiten que soporte protocolos de capas superiores como el TCP/IP.

L2CAP hace posible que los protocolos y aplicaciones de las capas superiores transmitan y reciban paquetes de datos de hasta 64 kilobytes de longitud, también hace posible el control de flujo por canal y la retransmisión. Cada extremo de un canal L2CAP se distingue mediante un identificador de canal (CID).

El protocolo L2CAP tiene tres modos de operación seleccionados para cada canal por una capa superior:

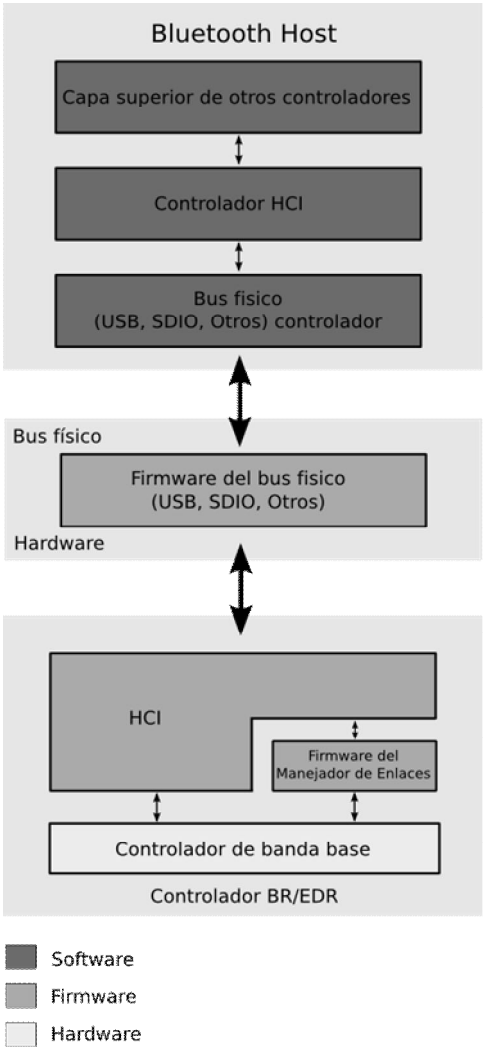
- Modo L2CAP básico
- Modo de control de flujo
- Modo de retransmisión

1.4.3. Interfaz de control del anfitrión (HCI)

La HCI (*Host Controller Interface*) proporciona una interfaz de comandos entre el controlador de la banda base y el gestor de enlaces, así también, proporciona acceso a los parámetros de configuración.

La interfaz HCI proporciona un método uniforme de acceso a todas las funciones de la banda base *Bluetooth*. Pueden existir varias capas entre los controladores del HCI en el sistema anfitrión y el *firmware* del HCI en el controlador.

Figura 11. **Descripción general de la capa inferior de *software***



Fuente: Bluetooth SIG, *Core V2.1 + EDR.* p. 357.

La utilización del HCI permite que una aplicación *Bluetooth* acceda al *hardware* sin la necesidad de conocer la capa de transporte.

1.5. *Bluetooth Link Manager* (LM)

El *link manager* o gestor de enlaces se encarga del manejo de la seguridad, la configuración de enlaces y de establecer los enlaces L2CAP.

El protocolo de gestión de enlace (LMP) se usa para controlar y negociar todos los aspectos de funcionamiento de la conexión *Bluetooth* entre dos dispositivos. LMP es el protocolo utilizado para la comunicación entre los gestores de enlaces entre dispositivos. El protocolo LMP funciona mediante transacciones; en donde una transacción consiste en el intercambio de mensajes con un fin determinado.

El gestor de enlaces se encarga del manejo de la autenticación y encriptación de datos entre dispositivos, para esto se requieren conexiones entre capas de nivel más alto por lo que se utilizan enlaces L2CAP para este fin.

La autenticación se da al momento que una unidad interroga o verifica a otro dispositivo en la red; este otro dispositivo responde enviando unidades de protocolo de datos (PDUs) basados en la interrogación. Para que se logre el proceso de autenticación los dispositivos generan claves de enlaces.

Cuatro claves son usadas para este propósito: clave de inicialización, clave de la unidad, clave maestra y una clave combinada.

Tabla VI. **Claves de enlaces**

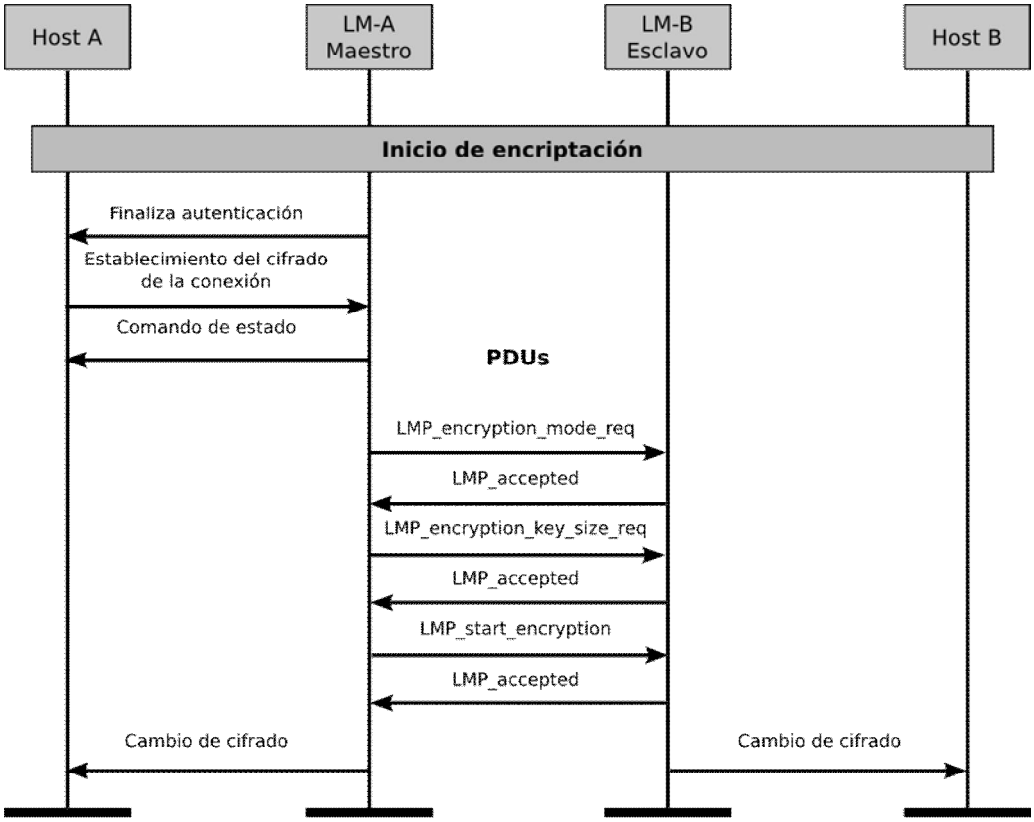
Tipo de clave	Descripción
Clave de inicialización	Utilizada para el proceso de inicialización. Protege la transferencia de los parámetros de configuración.
Clave de unidad	Es generada en el momento que se instala una uinidad *Bluetooth*, raramente es cambiada.
Clave maestra	Reemplaza temporalmente la clave original del enlace. Es usada únicamente en la sesión actual. Se usa cuando el dispositivo maestro desea transmitir a varias unidades al mismo tiempo.
Clave combinada	Se deriva de la información de las unidades interconectadas. Se usa en aplicaciones que requieran un nivel de suguirad alto. Requiere el uso de más memoria.

Fuente: BAKKER, D.M.; MCMICHAEL, Diane. *Bluetooth end to end*. p. 95.

El gestor de enlaces también se encarga de encriptar los datos que serán transmitidos entre dispositivos. Este proceso se logra mediante un sistema de cifrado de flujo que se aplica a la carga útil de los paquetes transmitidos y el cual es resincronizado para cada carga útil que se transmitirá.

Para encriptar los datos la unidad maestra y la esclava deben de acordar si es conveniente aplicar la encriptación, el modo que utilizarán y deben negociar el tamaño de la clave.

Figura 12. **Secuencia para encriptación de datos entre dispositivos**



Fuente: Bluetooth SIG, *Core V2.1 + EDR.* p. 712.

Las unidades de protocolos de datos son transmitidas por un gestor de enlaces a otros LMs por medio del protocolo LMP, estos permiten el envío de mensajes entre unidades. Un PDU puede ser obligatorio u opcional. El gestor de enlaces no está obligado a transmitir un PDU opcional, pero debe enviar una respuesta válida.

Tabla VII. **PDUs obligatorios del manejador de enlaces**

Uso	PDU
Respuesta general	LMP_accepted LMP_not_accepted
Autenticación	LMP_au_rand LMP_sres
Emparejado	LMP_in_rand LMP_au_rand LMP_sres LMP_comb_key LMP_unit_key
Cambio de clave de enlace	LMP_comb_key
Cambio de clave de enlace actual	LMP_temp_rand LMP_temp_key LMP_use_semi_permanent_key
Petición para compensación de reloj	LMP_clkoffset_req LMP_clkoffser_res
Versión del LMP	LMP_version_req LMP_version_res
Características soportadas	LMP_features_req LMP_features_res
Petición de nombre	LMP_name_req LMP_name_res
Separación	LMP_detach
Calidad de servicio (QoS)	LMP_quality_of_service LMP_quality_of_service_req
Control de paquetes de multiples ranuras	LMP_max_slot LMP_max_slot_req
Supervisión de enlace	LMP_supervision_timeout
Establecimiento de la conexión	LMP_host_connection_req LMP_setup_complete
Modo de prueba	LMP_test_control
Manejo de errores	LMP_not_accepted

Fuente: BAKKER, D.M.; MCMICHAEL, Diane. *Bluetooth end to end*. p. 100-102.

Tabla VIII. **PDUs opcionales del manejador de enlaces**

Uso	PDU
Petición para compensación de ranura	LMP_slot_*offset*
Requerimiento de información sobre la exactitud de la sincronización	LMP_timing_accuracy_req LMP_timing_accuracy_res
Conmutación Maestro-Esclavo	LMP_switch_req LMP_slot_offset
Modo *hold*	LMP_hold LMP_hold_req
Modo *sniff*	LMP_sniff_req LMP_unsniff_req
Modo *park*	LMP_park_req LMP_unpark_PM_ADDR_req LMP_unpark_BD_ADDR_req LMP_set_broadcast_scan_window LMP_modify_beacon
Control de energia	LMP_incr_power_req LMP_decr_power_req LMP_max_power LMP_min_power
Cambio de calidad de canal	LMP_auto_rate LMP_preferred_rate
Enlace SCO	LMP_SCO_link_req LMP_remove_SCO_link_req
Esquema de paginación	LMP_page_mode_req LMP_page_scan_mode_req
Cifrado	LMP_encryption_mode_req LMP_encryption_key_size_req LMP_start_encryption_req LMP_stop_encryption_req

Fuente: BAKKER, D.M.; MCMICHAEL, Diane. *Bluetooth end to end*. p. 100-102.

1.6. Estados de operación

Para poder controlar las actividades de una unidad en un canal de una piconet el enlace de control *Bluetooth* tiene varios estados de operación. Las unidades *Bluetooth* intercambian de estado por medio de comandos ejecutados a través del manejador de enlaces o por señales internas enviadas por el controlador de enlaces.

Cada unidad posee características únicas que la distinguen de otras unidades y que es esencial para que las unidades envíen y reciban información. Una de las características es la dirección *Bluetooth* de la unidad maestra que determina el código de acceso y el salto de frecuencia en una piconet. El reloj interno natural de cada unidad también es único y establece la fase del salto de frecuencia.

1.6.1. Estados mayores

El estado de operación mayor controla el estado de la conexión de un dispositivo dentro de una piconet. Existen dos estados mayores, el de espera y el de conexión.

- Estado de espera: este estado es el predeterminado para cualquier unidad. Este modo requiere bajo consumo de energía y puede ser usado para liberar la capacidad de una piconet.

- Estado de conexión: en este estado, dos o más unidades tienen una conexión activa establecida y pueden intercambiar paquetes entre si.

1.6.2. Subestados

Un dispositivo *Bluetooth* que no se encuentre activo en una piconet se encuentra en un estado de espera; en este estado existe un proceso denominado escanear el cual permite detectar mensajes que son usados para agregar o eliminar unidades de una piconet. El proceso de escanear se divide en dos grupos: subestado de paginación y subestado de interrogación.

1.6.2.1. Subestados de paginación

Este estado es usado por los dispositivos *Bluetooth* para localizar y establecer una conexión entre una unidad maestra y una esclava.

- *Page scan*: en este subestado existe una ventana de exploración de 11,25 ms. Durante este tiempo los dispositivos esclavos exploran su DAC (*Device Access Code*) por medio de uno de los 32 posibles saltos de frecuencia.

- *Page*: en este subestado la unidad maestra envía un tren de escaneo repetidamente a través de cada salto de frecuencia. Durante este tren se envia repetidamente el DAC de la unidad esclava tratando de establecer una conexión entre el maestro y el esclavo.

- *Page response*: este subestado la unidad maestra y la unidad esclava intercambian información necesaria para establecer una conexión.

1.6.2.2. Subestados de interrogación

En estos subestados la unidad maestra recolecta información necesaria para lograr establecer una conexión. Las unidades utilizan tres subestados de interrogación.

- *Inquiry scan*: este subestado permite que la unidad maestra localice unidades esclavas potenciales. En este subestado no se conoce el DAC para lograr establecer una conexión.

- *Inquiry*: en este subestado la unidad maestra obtiene el DAC necesario para establecer la conexión.

- *Inquiry response*: en este estado la unidad esclava responde un mensaje que contiene su DAC para que el dispositivo maestro entre en el subestado de paginación (*page*) y empiece a establecer una conexión.

1.6.2.3. Subestados de conexión

Cuando existe una comunicación entre dos dispositivos y se encuentran en el estado de conexión pueden operar en cuatro modos diferentes:

- *Active*: cuando un dispositivo se encuentra participando en una piconet este opera en el modo activo y el tráfico de información es programado dependiendo de sus necesidades. Cuando un dispositivo se encuentra en este modo se le asigna una dirección de miembro activo (AM_ADDR, *Active Member Address* por sus siglas en inglés).

- *Hold*: el modo en espera utiliza menos energía. En este modo un dispositivo puede realizar otras tareas, como escaneos de paginación o interrogación y aún así permanecer activo en la piconet.

- *Sniff*: en este modo se reduce el uso de energía y se puede compartir tiempo de conexión entre piconets.

- *Park*: este modo permite a que una unidad permanezca sincronizada con su canal y que reciba mensajes de tipo *broadcast*, sin necesidad de participar en una piconet. En este modo se pierde la dirección AM_ADDR y se asigna una dirección de miembro estacionado (PM_ADDR, *Parked Member Addres* por sus siglas en inglés).

1.7. Perfiles *Bluetooth*

Un perfil describe las distintas aplicaciones que se pueden encontrar dentro de esta tecnología inalámbrica. Un perfil es una guía que indica los procedimientos por los que un dispositivo con tecnología *Bluetooth* se comunica con otro. Existen varios perfiles que detallan los diferentes tipos de uso y aplicaciones.

Cada perfil al menos debe incluir información sobre:

- La dependencia de otros perfiles
- Propuesta sobre el formato de interfaz de usuario
- Características concretas de la pila de protocolos *Bluetooth* utilizada por el perfil. Para realizar su función, cada uno de los perfiles necesita ciertas opciones y parámetros en cada capa de la pila.

Tabla IX. **Perfiles *Bluetooth***

Perfil	Descripción
Perfil de distribución de audio avanzado (A2DP)	El perfil A2DP describe cómo transferir sonido estéreo de alta calidad de una fuente de sonido a un dispositivo receptor.
Protocolo de control de audio y vídeo (AVRCP)	Proporciona una interfaz estándar para manejar televisiones, equipos de alta fidelidad o cualquier otro equipo electrónico, y permitir así que un único control remoto, o cualquier otro tipo de mando, controle todo el equipo de audio y vídeo al que el usuario tiene acceso.
Perfil básico de imagen (BIP)	Establece cómo puede controlarse remotamente un dispositivo de imagen, así como la forma de enviarle órdenes de impresión y de transferencia de imágenes a un dispositivo de almacenamiento.
Perfil básico de impresión (BPP)	El perfil BPP permite enviar mensajes de texto, de correo electrónico, tarjetas de visita electrónicas e imágenes, entre otras cosas, a las impresoras disponibles dependiendo de las tareas de impresión.
Perfil de acceso RDSI común (CIP)	El perfil CIP establece cómo se deben transferir las señales RDSI a través de una conexión inalámbrica *Bluetooth*.
Perfil de telefonía inalámbrica (CTP)	El perfil CTP describe la implementación de un teléfono inalámbrico a través de un enlace inalámbrico *Bluetooth*.
Perfil de red de marcado (DUN)	El perfil DUN proporciona un acceso telefónico estándar a Internet y a otros servicios de marcado a través de una conexión *Bluetooth*.
Perfil de fax (FAX)	El perfil FAX describe cómo un dispositivo terminal puede utilizar a otro como puerta de enlace para la transmisión de faxes.
Perfil de transferencia de archivos (FTP)	El perfil FTP establece los procedimientos de exploración de carpetas y archivos de un servidor a través de un dispositivo cliente.
Perfil de distribución genérica de audio y vídeo (GAVDP)	El perfil GAVDP sienta las bases de los perfiles A2DP y VDP, pilar de los sistemas diseñados para la transmisión de sonido e imagen mediante la tecnología *Bluetooth*.
Perfil genérico de intercambio de objetos (GOEP)	El GOEP se utiliza para transferir objetos de un dispositivo a otro.
Perfil manos libres (HFP)	Describe cómo un dispositivo que actúa como puerta de enlace puede utilizarse para realizar y recibir llamadas a través de un dispositivo manos libres.
Perfil de sustitución de cable de copia impresa (HCRP)	El perfil HCRP describe cómo imprimir archivos mediante un enlace inalámbrico Bluetooth utilizando controladores en el proceso.
Perfil de auricular (HSP)	El HSP describe cómo un auricular con tecnología Bluetooth se comunica con otro dispositivo con tecnología *Bluetooth*.
Perfil de dispositivo de interfaz humana (HID)	El perfil HID recoge los protocolos, procedimientos y características empleados por las interfaces de usuario *Bluetooth* tales como teclados, dispositivos punteros, consolas o aparatos de control remoto.
Perfil de intercomunicador (ICP)	El perfil ICP establece cómo conectar dos teléfonos móviles con tecnología *Bluetooth* dentro la misma red sin utilizar la red telefónica pública.
Perfil de introducción de objetos (OPP)	Este perfil distingue entre servidor y cliente de introducción (*push*) de objetos.
Perfil de redes de área personal (PAN)	El perfil PAN describe cómo dos o más dispositivos con tecnología *Bluetooth* pueden formar una red *ad hoc* y cómo ese mismo mecanismo permite acceder a la red de forma remota a través de un punto de acceso.
Perfil de aplicación de descubrimiento de servicio (SDAP)	El perfil SDAP detalla cómo una aplicación debe utilizar el perfil SDP para identificar los servicios de un dispositivo remoto.
Perfil de servicio de puerto (SPP)	El perfil SPP describe cómo configurar puertos de serie y conectar dos dispositivos con tecnología *Bluetooth*.
Perfil de sincronización (SYNC)	El perfil SYNC se utiliza junto al GOEP para sincronizar los elementos del administrador de información personal (PIM), como agendas y datos de contacto, entre dispositivos con tecnología *Bluetooth*.
Perfil de distribución de vídeo (VDP)	Este perfil dicta los pasos que deben seguir los dispositivos Bluetooth con tecnología *Bluetooth* para la transferencia de flujos de datos de vídeo.

Fuente: <http://en.wikipedia.org/wiki/Bluetooth_profile>. [Consulta: julio de 2010].

2. FUNDAMENTOS DEL PROTOCOLO TCP/IP

2.1. Arquitectura del modelo TCP/IP

El protocolo TCP/IP es el grupo de protocolos de comunicación usado para la interconexión de redes. Su nombre se deriva del protocolo de control de transmisión (*Transmission Control Protocol*, *TCP*) y del protocolo de Internet (*Internet Protocol*, IP).

2.1.1. *Internetworking*

El término *Internetwork* o *Internet* se refiere a la interconexión de redes alrededor de todo el mundo. El protocolo TCP/IP fue diseñado principalmente para la interconexión de estas redes y proveer servicios de comunicación a través de ellas.

La Internet consiste en cuatro grupos de redes denominados como *backbones*, redes regionales, redes comerciales y redes locales.

El protocolo TCP/IP permite estandarizar la abstracción de los mecanismos de comunicación de cada red. Cada red física posee tecnología propia en su interfaz de comunicación y el protocolo TCP/IP establece una interfaz común por medio de servicios que se ejecutan entre la interface de programación de la red física y las aplicaciones de usuarios.

2.1.2. Capas del protocolo TCP/IP

La división de la pila de protocolos en capas permite la división de tareas y una fácil implementación. Cada capa se comunica con las capas inferiores y superiores por medio de interfaces. Dentro de cada una de las capas se encapsulan los datos para que de esta forma se permita la abstracción de protocolos y servicios.

El modelo TCP/IP está dividido en cuatro capas:

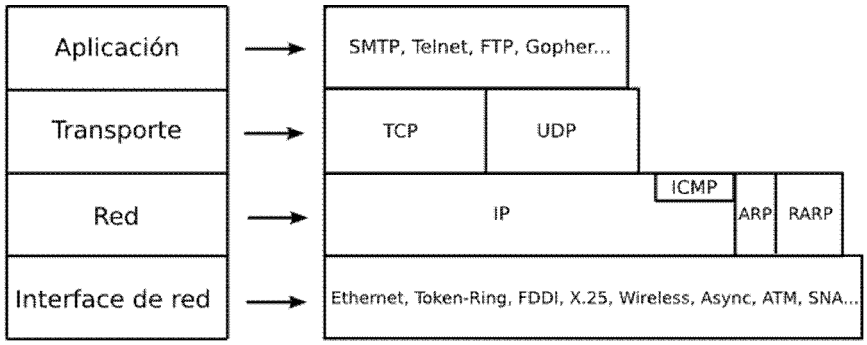
- Capa de aplicación: esta capa la define el programa o aplicación que utiliza el protocolo TCP/IP para comunicarse. Una aplicación es un proceso ejecutado por usuario el cual trabaja en conjunto con otro proceso que usualmente se está ejecutando en otro *host*.

- Capa de transporte: la capa de transporte permite transferencia de información de extremo a extremo por medio de la entrega de datos de la aplicación a la conexión remota. Los protocolos más usados en esta capa son el protocolo TCP y el UDP (*User Datagram Protocol,* por sus siglas en inglés).

- Capa de red: ésta capa permite transportar paquetes a través de las fronteras de la red y así conectar múltiples redes una con otra. El protocolo más importante de esta capa el IP ya que realiza la función de enrutamiento para la entrega de los mensajes transmitidos a su destino.

- Capa de enlace: también llamada capa de interfaz de red. Es la interface con el *hardware* de la red y puede ser orientada a paquetes o a flujo de datos. TCP/IP no especifica ningún protocolo en esta capa, pero se puede usar casi cualquier interfaz de red disponible.

Figura 13. **Detalle del modelo TCP/IP**



Fuente: PARZIALE, Lydia; BRITT, David. *TCP/IP tutorial and technical overview*. p. 9.

En la figura 13 se muestran las diferentes capas del modelo TCP/IP y los protocolos soportados por cada una de ellas. La capa de más bajo nivel es la capa de interfaz de red o capa de enlace y la capa de nivel más alto es la capa de aplicación.

2.1.3. Aplicaciones

Los protocolos de la capa de aplicaciones son los de mayor nivel en el modelo TCP/IP siendo estos los que se comunican con otros servidores en la red y permiten al usuario tener una interfaz visible.

Todos los protocolos de aplicación presentan ciertas características en común:

- Son aplicaciones escritas por el usuario o aplicaciones ya incluidas dentro del modelo (Telnet, FTP, SMTP).

- Usan como mecanismo de transporte los protocolos UPD o TCP

- La mayoría de las aplicaciones utilizan el modelo cliente/servidor

En el modelo cliente/servidor el servidor es una aplicación que provee servicios a los usuarios de la red mediante recepción de peticiones. El cliente simplemente solicita el uso de esos servicios enviando las peticiones correspondientes al servidor usando el protocolo TCP/IP como transporte. Una aplicación que utilice este modelo consiste en ambas partes, un cliente y un servidor.

2.2. Protocolo de Internet (IP)

El protocolo IP es el protocolo principal utilizado en la capa de red. Es usado para la comunicación de datos a través de redes de conmutación de paquetes. Este protocolo esconde la capa física mediante la creación de una vista virtual de la red; esto crea un protocolo no confiable el cuál puede tener pérdida de paquetes o incluso enviar paquetes repetidos a un destino. IP asume que una capa superior se ocupará de estos problemas.

2.2.1. Direccionamiento IP

El direccionamiento IP se refiere a cómo se asignan las direcciones IP a un *host* en particular y cómo las subredes de direcciones IP son divididas en grupos.

La dirección IP es el número que identifica de manera lógica a la interfaz de un dispositivo dentro de una red IP. Son representadas por un valor binario de 32-bit que es dividido en cuatro octetos. El valor decimal de cada octeto puede ser entre 0 y 255. Los primeros bits de la dirección IP determinan la forma en la que la dirección se dividirá en la parte de red (*netID*) y del *host* (*hostID*).

Existe una clasificación que divide las direcciones IP en diferentes clases generalmente dependiendo de la cantidad de redes y de *hosts* que se necesitan. Las clases de direcciones IP se dividen de la siguiente forma:

- Clase A: las direcciones clase A utilizan 7 bits para identificar la red y 24 bits para identificar al *host*. Esta configuración permite crear 126 redes, cada una con 16 777 214 *hosts*.

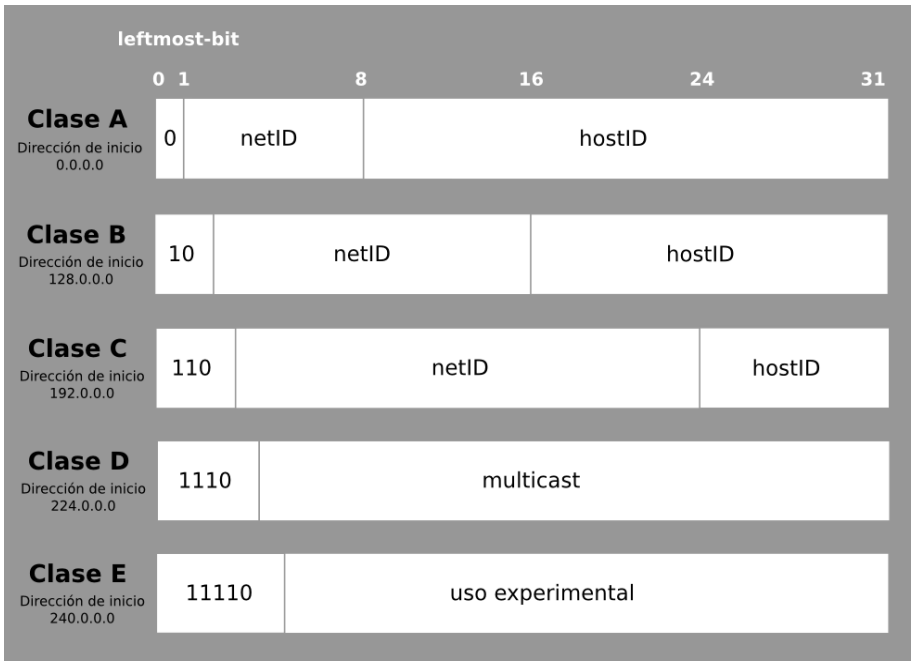
- Case B: estas direcciones utilizan 14 bits para la red y 16 bits para los *hosts* permitiendo tener 16 382 redes de 65 534 *hosts* cada una.

- Clase C: utilizan 21 bits para la red y 8 bits para los *hosts* lo que permite crear 2 097 150 redes de 254 *hosts* cada una.

- Clase D: son direcciones reservadas para *multicasting*

- Clase E: son direcciones reservadas para uso experimental

Figura 14. **Clasificación de direcciones IP**



Fuente: PARZIALE, Lydia; BRITT, David. *TCP/IP tutorial and technical overview*. p. 70.

En la figura 14 se muestran las diferentes clasificaciones de las direcciones IP. Los bits hacia la izquierda (*leftmost-bit*) determinarán la dirección de inicio y de qué forma se dividirán los bits para formar el ID de red y el ID del *host*.

2.2.2. Subredes IP

Una subred es una red que es dividida en varias direcciones lógicas debido al crecimiento de la red permitiendo de esta forma que la red sea más manejable. El concepto de subred fue introducido para evitar tener que solicitar una nueva dirección IP al momento de tener que expandir y realizar cambios en la configuración de una red interna.

Para crear una subred la parte de identificación del *host* es subdividida en una segunda dirección de red y en otro número de *host*. Esta segunda dirección de red es la llamada subred de trabajo o simplemente subred. La parte de la dirección IP que identifica la subred y al *host* se conoce como la parte local de la dirección IP. Esta división se realiza mediante una máscara de subred de 32 bits donde los bits con valor lógico 1 corresponden a las posiciones atribuidas a la dirección de subred y los bits con valor lógico 0 a las posiciones atribuidas a número de *host*.

Existen dos tipos de subredes, la estática y de la de tamaño variable. Una subred estática implica que todas las subredes que se han obtenido de una misma red utilicen la misma máscara de subred. Las subredes de tamaño variable de una misma red pueden utilizar diferentes máscaras de subred permitiendo de esta forma que la red sea más flexible que una subred estática.

2.2.3. Enrutamiento IP

Una de las funciones principales de la capa de red es el enrutamiento IP ya que permite que los ruteadores (*routers*) interconecten diferentes redes físicas.

Existen dos tipos de enrutamiento:

- Enrutamiento directo: este caso se da cuando los paquetes pueden ser intercambiados directamente debido a que el *host* de destino se encuentra en la misma red física que el *host* de origen.

- Enrutamiento indirecto: cuando el *host* destino no se encuentra en la misma red física que el *host* origen, los paquetes deben ser enviados a través de uno o varios *gateways*.

Si dos *hosts* que deseen comunicarse entre sí se encuentran en la misma red física pero están definidos en diferentes subredes se debe utilizar un enrutamiento indirecto.

El enrutamiento directo se deriva del listado de las interfaces locales y es creado automáticamente al inicializar el proceso de enrutamiento IP. Para los enrutamientos indirectos se puede configurar una lista de las redes y de los *gateways* asociados. Cada *host* mapea la dirección IP de destino y las rutas para los próximos *gateways* en una tabla llamada tabla de enrutamiento.

En la tabla de enrutamiento existen tres tipos de mapeos:

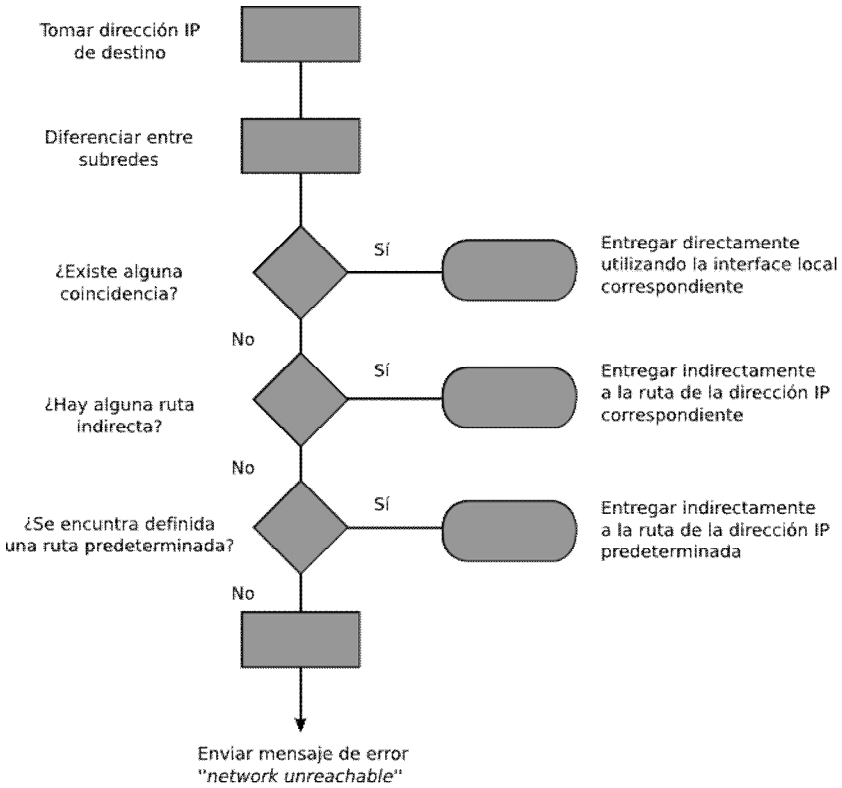
- Rutas directas que describen redes asociadas localmente

- Rutas indirectas que describen redes que pueden ser alcanzadas por medio de uno o más *gateways*.

- La ruta predeterminada que es usada cuando la red IP destino no se encuentra mapeada en la tabla.

En la figura 15 se muestra el diagrama de flujo del algoritmo de ruteo. El algoritmo toma la IP de destino, posteriormente verifica si esta pertenece a algún *host* definido en otra subred. Mediante una serie de interrogaciones toma ciertas decisiones que permitirán la entrega de los paquetes enviados, de lo contrario envía mensaje de error.

Figura 15. **Diagrama del algoritmo de ruteo**



Fuente: PARZIALE, Lydia; BRITT, David. *TCP/IP tutorial and technical overview*. p. 83.

2.2.4. Esquemas de enrutamiento

En la mayoría de los casos las direcciones IP se refieren a un solo destinatario, sin embargo, hay casos especiales donde ciertas direcciones son usadas para enrutar la información a múltiples destinatarios.

Las conexiones que requieren una relación uno a uno entre la fuente y un destinatario utilizan el método denominado *unicast*. Un protocolo que requiera que ambos *hosts* estén conectados entre ellos debe utilizar este método. Para direccionar a múltiples destinatarios son utilizados tres métodos de entrega diferentes:

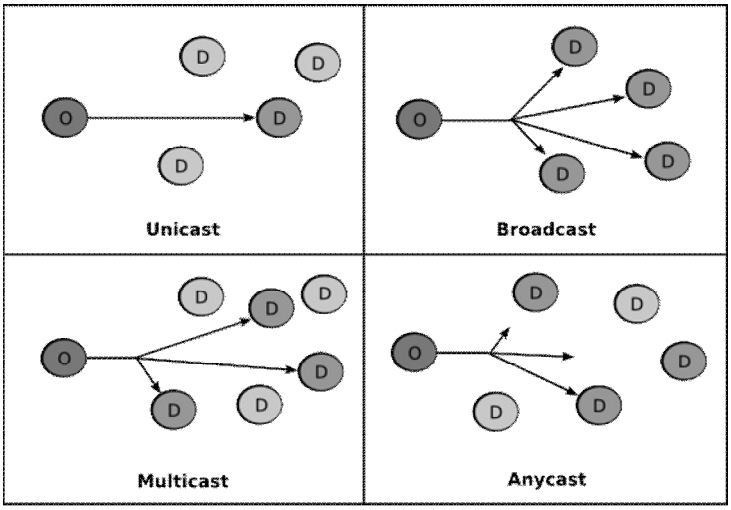
- *Broadcasting*: este método permite direccionar mensajes a todos los nodos dentro de una red. Las direcciones para *broadcast* no son válidas como direcciones de origen ya que deben especificar la dirección de destino.

- *Multicasting*: con este método se direccionan mensajes solamente al grupo de nodos que han expresado interés en recibir la información enviada.

- *Anycasting*: este método de direccionamiento enruta la información al nodo destino que posea la mejor conexión hacia el servidor. Los ruteadores permiten encaminar un paquete por el destino más cercano a una dirección IP que se anuncie desde varios puntos diferentes.

En la figura 16 se muestran las cuatro formas de direccionamiento usados para la entrega de información en una red. Los nodos identificados con una “O” representan el nodo donde se origina un mensaje. Los nodos identificados con una “D” representan los nodos que pueden desempeñar la función de destinatario.

Figura 16. **Esquemas de enrutamiento**



Fuente: <http://en.wikipedia.org/wiki/Routing>. [Consulta: julio de 2010].

2.3. Protocolos de capa de transporte

Los protocolos de transporte se encargan de entregar los datos a las aplicaciones que se ejecutan en una red por medio del uso de puertos. Los protocolos de transporte aportan funcionalidades como control de congestión, entrega confiable de información, control de flujo y eliminación de datos duplicados.

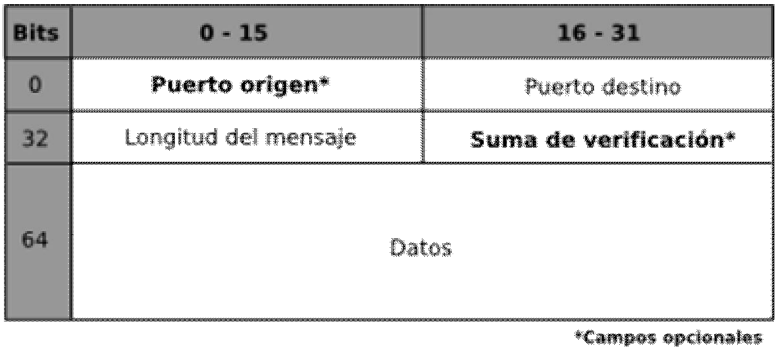
Los protocolos más usados en la capa de transporte TCP/IP son el *User Datagram Protocol* (UDP) y el *Transmission Control Protocol* (TCP).

2.3.1. *User Datagram Protocol* (UDP)

Es un protocolo de la capa de transporte el cuál basa su funcionamiento en el intercambio de datagramas. Permite el envío de datagramas sin la necesidad de establecer una conexión previa ya que el datagrama posee suficiente información sobre su direccionamiento en su cabecera.

UDP es una interfaz sencilla entre la capa de red y la de aplicación. Una de sus desventajas es que no proporciona ninguna garantía para transmitir información ya que el *host* que origina los mensajes no retiene el estado de los mismos; estas garantías deben ser implementadas en capas superiores.

Figura 17. **Formato de un datagrama UDP**



Fuente: <http://es.wikipedia.org/wiki/User_Datagram_Protocol>. [Consulta: agosto de 2010]

En la figura 17 se muestra el formato de un datagrama el cual consta de cuatro campos de cabecera donde dos de ellos son opcionales. El puerto de origen y destino son campos de 16 bits. La longitud del mensaje es un campo obligatorio que indica el tamaño en bytes del datagrama. El último campo de la cabecera es una suma de verificación de 16 bits.

El protocolo UDP es utilizado en ocasiones donde la velocidad de transmisión es más importante que garantizar el envío correcto de todos los bytes del mensaje, este es el caso para la transmisión de audio o video.

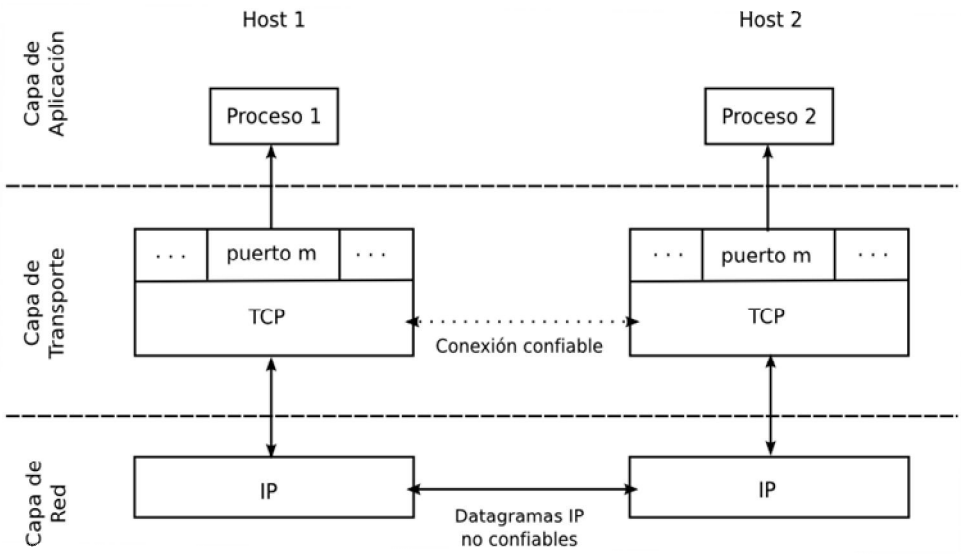
2.3.2. *Transmission Control Protocol* (TCP)

TCP es un protocolo orientado a conexión por lo que aporta funciones a las aplicaciones para controlar la información enviada a través de la red. TCP posee varias características, entre ellas la verificación de errores, control de flujo y confiabilidad. El protocolo garantiza que la información será entregada al *host* destino sin errores y en el mismo orden que fue transmitida.

El propósito principal del protocolo TCP es proveer una conexión confiable entre un par de procesos. TCP no asume que los protocolos de capas inferiores (como el protocolo IP) proporcionen datos confiables, por lo que él debe garantizarlo por sí mismo.

En la figura 18 se visualiza la forma en la que dos procesos se comunican por medio del protocolo TCP. Los procesos 1 y 2 se comunican a través de TCP llevados por datagramas transmitidos en la capa de red mediante el protocolo IP.

Figura 18. **Conexión entre procesos a través de TCP**



Fuente: PARZIALE, Lydia; BRITT, David. *TCP/IP tutorial and technical overview*. p. 150.

Un protocolo de transporte simple puede que utilice el principio de ventana. Básicamente este principio consiste en el envío de un paquete y luego esperar el reconocimiento del receptor antes de enviar el próximo paquete. Si el reconocimiento no es recibido después de un tiempo estipulado el paquete es retransmitido. TCP utiliza este principio enviando segmentos y recibiendo reconocimientos que transportan los números de secuencia de los bytes transmitidos. El tamaño de la ventana es expresado en número de bytes y es definido por el receptor cuando la conexión se establece y varía durante la transmisión de datos.

TCP soporta muchas de las aplicaciones más conocidas de Internet, entre ellas HTTP, SMTP, SSH, FTP. Para distinguir entre las diferentes aplicaciones se utiliza el concepto de puertos.

2.4. Puertos y *sockets*

Un puerto es un número de 16 bits que es utilizado por el protocolo de transporte para identificar al proceso o programa que recibirá los mensajes transmitidos a una capa superior. El uso de puertos permite ejecutar varias aplicaciones que utilicen el mismo protocolo de transporte simultáneamente en el mismo dispositivo.

En programación de redes un *socket* es una interface al protocolo de comunicación y representa el punto final de un enlace de comunicación. Un *socket* queda definido por un protocolo de transporte, una dirección IP y un número de puerto. Los *sockets* permiten implementar una arquitectura cliente-servidor donde la comunicación es iniciada por la aplicación cliente.

3. PROGRAMACIÓN DE DISPOSITIVOS *BLUETOOTH*

3.1. Conceptos de programación

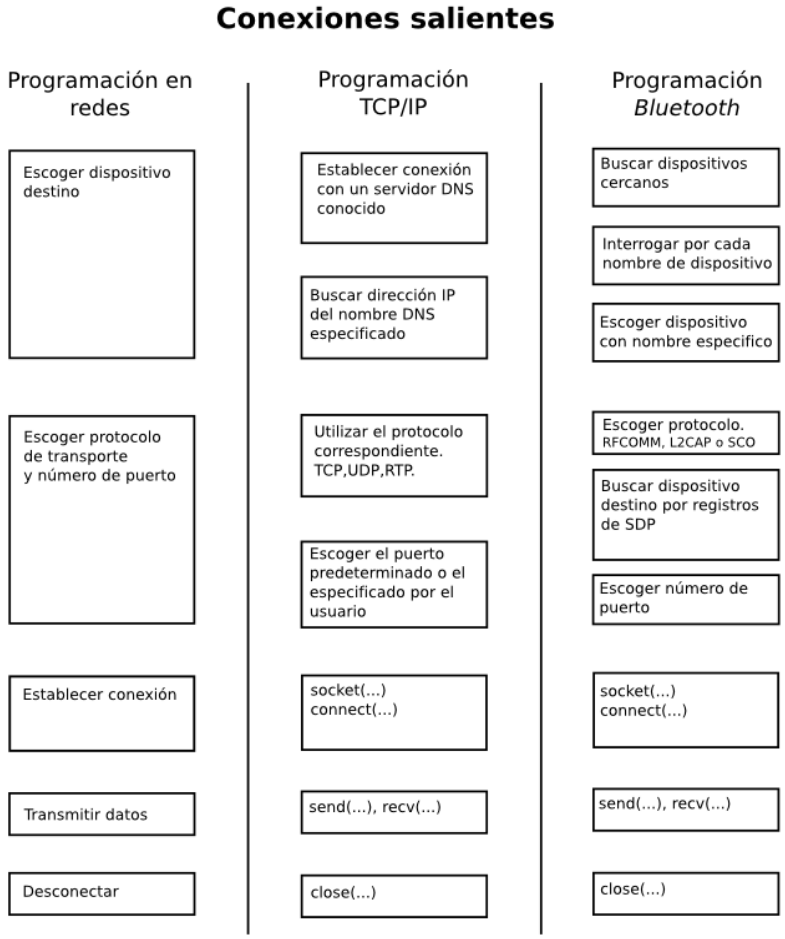
Para entender los conceptos de programación para dispositivos *Bluetooth* se puede realizar cierta comparación con los conceptos aplicados para la programación de aplicaciones para Internet ya que ambas tecnologías comparten los mismos principios de comunicación y comparten datos con otros dispositivos con los cuales se interconectan.

El comparar TCP/IP con *Bluetooth* permite entender con mayor claridad el proceso de comunicación entre dispositivos, debido a la madurez de la programación TCP/IP.

Durante el proceso de conexión, un dispositivo que establece una conexión saliente es denominado como dispositivo cliente; y el dispositivo que establece una conexión entrante se denomina como dispositivo servidor. Esta denominación es independiente del modelo cliente-servidor, únicamente es utilizada para identificar quien inicia la comunicación.

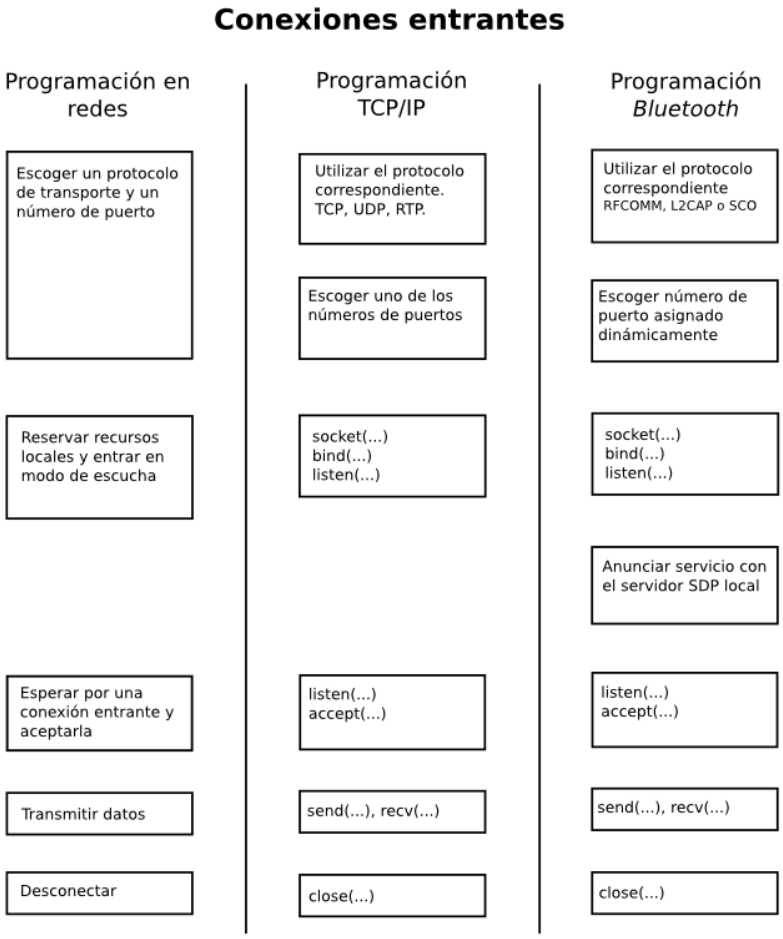
En las figuras 19 y 20 se muestra el concepto básico para programación de las conexiones salientes y entrantes para TCP/IP y *Bluetooth*. Un dispositivo que inicie una conexión saliente debe escoger un dispositivo destino y un protocolo de transporte; y un dispositivo que establezca una conexión entrante debe escoger el protocolo de transporte y posteriormente escuchar antes de aceptar la conexión.

Figura 19. **Pasos para establecer una conexión saliente (cliente)**



Fuente: HUAN, Albert; RUDOLPH, Larry. *Bluetooth essentials for programmers*. p. 4.

Figura 20. **Pasos para establecer una conexión entrante (servidor)**



Fuente: HUAN, Albert; RUDOLPH, Larry. *Bluetooth essentials for programmers*. p. 5.

Algunos de los pasos mostrados en la figura 20 no aplican para todos los casos y pueden ejecutarse en diferente orden. Esto debido a que en ciertas ocasiones el cliente puede contar con algunos datos para establecer la conexión por lo que ya no es necesario consultarlos en otro servidor o por medio de interrogación a otro dispositivo.

3.2. Direcciones y nombres de dispositivos *Bluetooth*

A cada chip *Bluetooth* se le asigna una dirección de 48-bit. Esta dirección cumple la misma función que las direcciones MAC para las interfaces *Ethernet* la cuál es proveer un número de identificación único para cada dispositivo.

Para que un dispositivo *Bluetooth* pueda comunicarse con otro debe conocer la dirección del dispositivo con el que se desea comunicar. Al contrario que el protocolo TCP/IP, *Bluetooth* utiliza esta dirección en todas las capas durante el proceso de comunicación.

Además de la dirección de 48-bit los dispositivos *Bluetooth* casi siempre tendrán un nombre amigable. Este nombre permite que las personas identifiquen más fácilmente un dispositivo debido a que los humanos no asimilan fácilmente un número binario. Los nombres en los dispositivos *Bluetooth* son similares a los nombres *DNS* (Por sus siglas en inglés, *Domain Name Server*) en el sentido que ambos son fáciles de memorizar por las personas. A diferencia de los nombres DNS los nombres *Bluetooth* pueden estar repetidos en diferentes dispositivos.

Para conseguir esta información un dispositivo *Bluetooth* ejecuta un proceso denominado interrogación del dispositivo (*device inquiry*). De esta forma busca y detecta dispositivos cercanos a él, y consolida una lista de las direcciones de los dispositivos encontrados.

El proceso de interrogación usualmente requiere de 5 segundos aproximadamente para detectar los dispositivos cercanos y algunas veces podría tardar entre 10 y 15 segundos.

Este proceso puede llegar a ser muy lento y debe ser considerado durante la programación de aplicaciones y dispositivos.

3.3. Configuración de conectividad y visibilidad

Por razones de seguridad y por ahorro de energía los subestados de operación *Inquiry y Page Scan* en un dispositivo pueden ser configurados en algunas ocasiones por el usuario. En la Tabla X se muestran las diferentes combinaciones en las que se pueden configurar estos subestados y se describe el comportamiento del dispositivo local.

Tabla X. **Configuración de subestados *Inquiry y Page Scan***

Inquiry Scan	Page Scan	Comportamiento del dispositivo
On	On	Puede ser detectado por otros dispositivos *Bluetooh* y responde las solicitudes de conexión. Generalmente es una de las configuraciones estándar.
Off	On	No es detectado por otros dispositivos, pero responde las solicitudes de conexión de los dispositivos que conozcan su dirección *Bluetooth*.
On	Off	El dispositivo es detectado por otras unidades *Bluetooth* pero no aceptará solicitudes de conexión. Prácticamente ésta configuración no es funcional.
Off	Off	No será detectado por otras unidades y no aceptará las solicitudes de conexión. Esta configuración puede ser útil si el dispositivo solo establecerá conexiones salientes.

Fuente: HUAN, Albert; RUDOLPH, Larry. *Bluetooth essentials for programmers*. p. 9.

3.4. Protocolo de transporte

Dependiendo del tipo de información que se deseen transmitir se escogerá el protocolo de transporte a utilizar. Hay que tomar en cuenta dos factores importantes al momento de escoger el protocolo, el primer factor dependerá de que tan confiable se desea la conexión, es decir, si se quiere que el protocolo reenvíe los datos enviados por la aplicación la mayor cantidad de veces posibles para garantizar la entrega de los paquetes. El segundo factor dependerá si se quiere que la conexión sea basada en paquetes o en flujo (*stream*) de datos.

Bajo la perspectiva de un programador de redes si necesitará un protocolo confiable escogería el protocolo TCP orientado a conexión debido a la habilidad del reenvío de segmentos en caso de detectar fallas en el envío de los mismos; y si se requiere de un protocolo donde la confiabilidad no sea prioritaria y se requiera de una tasa más alta de transmisión por segundo, se optaría por la utilización del protocolo UDP, siendo este basado en el envío de paquetes o datagramas.

De los protocolos de *Bluetooth*, cuatro de ellos son esenciales al momento de programar una aplicación, RFCOMM (*Radio Frecuency Communication*), L2CAP, ACL y SCO.

3.4.1. Protocolo RFCOMM

El protocolo RFCOMM es un protocolo confiable basado en flujo de datos. Las especificaciones de este protocolo indican que es una emulación de una comunicación serial. Se puede decir que este protocolo simula el comportamiento de los actuales puertos seriales.

El protocolo RFCOMM puede ser comparado con el protocolo TCP ya que ambos proporcionan servicios y confiabilidad similares al momento de implementarlos. A diferencia del protocolo TCP al utilizar RFCOMM solamente se tienen 30 puertos disponibles en un mismo dispositivo.

3.4.2. Protocolo L2CAP

Este protocolo puede ser comparado con el protocolo UDP ya que ambos son protocolos basados en el envío de paquetes con la diferencia que L2CAP entrega los paquetes en el orden que fueron enviados y puede ser configurado con varios niveles de confiabilidad por medio de un esquema de transmisión y reconocimiento. Paquetes no reconocidos pueden ser retransmitidos o no, para este fin existen tres políticas: nunca retransmitir, siempre retransmitir hasta que no exista error o falle la conexión totalmente y eliminar paquetes que no hayan sido reconocidos después de cierta cantidad de tiempo (0 – 1 279 ms).

3.4.3. Protocolos ACL y SCO

El protocolo ACL puede ser comparado con el protoco IP, casi nunca se llegará a usar directamente para transportar datos ya que es un protocolo fundamental que es utilizado para encapsular la información de capas superiores.

El protocolo SCO es utilizado para transmitir exclusivamente audio a una velocidad de 64 kbps, por lo que es útil para transmisión de voz. Una conexión SCO nunca retransmite los paquetes y siempre garantiza una conexión de 64 kbps ya que los dispositivos *Bluetooth* no pueden tener más de tres conexiones SCO activas; en la práctica casi siempre se tendrá una única conexión SCO activa por dispositivo.

3.4.4. *Object Exchange* (OBEX)

Es un protocolo de comunicación que facilita el intercambio de objetos binarios entre dispositivos. Obex utiliza un modelo cliente-servidor y es independiente del mecanismo de transporte. El protocolo siempre define un objeto con el listado de carpetas, el cuál es usado para explorar el contenido de las carpetas en el dispositivo remoto. RFCOMM es usado como la capa de transporte principal.

3.5. Puertos

Una aplicación que se está ejecutando en un dispositivo utiliza los puertos para poder utilizar el mismo protocolo para diferentes conexiones, asignando un puerto a cada conexión.

Bluetooth utiliza el mismo concepto, con la diferencia que los puertos toman diferentes nombres dependiendo del protocolo que se esté utilizando. En L2CAP son llamadas PSM (*Protocol Service Multiplexers*, Protocolo de Servicio Multiplexor) y pueden ser números impares entre 1 y 32, 767. En RFCOMM pueden asignarse los puertos del 1 al 30 y son llamados canales.

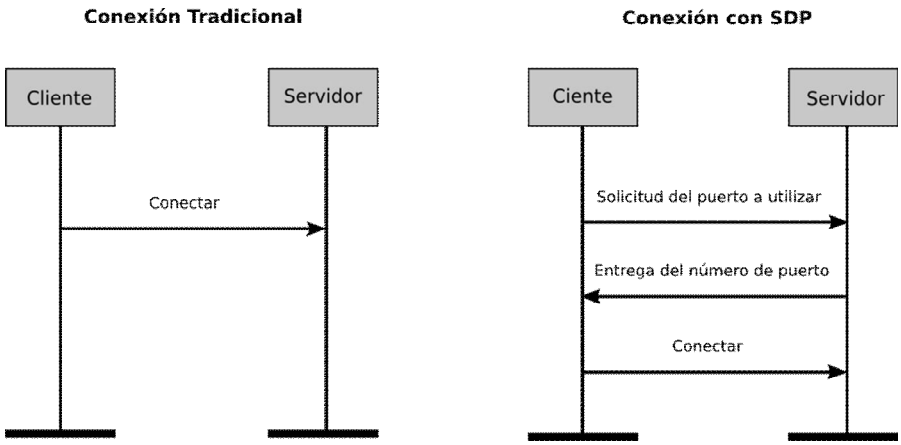
3.6. *Service Discovery Protocol* (SDP)

El protocolo de servicio de descubrimiento es utilizado para determinar el puerto en el cuál un servidor se encuentra escuchando para establecer una conexión. Cada dispositivo mantiene un servidor SDP escuchando en un puerto reservado.

Al ser iniciada una aplicación en el servidor, este guarda una descripción de él mismo y del número de puerto que se utiliza, en este caso el dispositivo local mantiene el SDP. Cuando un cliente remoto establece una conexión por primera vez al servidor entrega un listado de los servicios que busca en el SDP, el servidor responde entregando el listado de servicios que coinciden con los buscados por el cliente.

Con el SDP no se tienen conflictos con los puertos asignados debido que al ser iniciado el servidor escoge un puerto arbitrario que esté disponible. En la figura 21 se muestra la comparación entre el establecimiento de una conexión tradicional y una conexión usando SDP.

Figura 21. **Comparación entre conexión tradicional y con SDP**



Fuente: HUAN, Albert; RUDOLPH, Larry. *Bluetooth essentials for programmers*. p. 17.

3.6.1. Registro de servicio

Un registro de servicio o un registro SDP es un listado de atributos y de valores. Cada atributo es un número entero de 16-bit. Básicamente un registro SDP puede ser interpretado como un diccionario de las diferentes entradas que describe el servicio que se ofrece. Los dos atributos de mayor relevancia que se encuentran en un registro de servicio ya que son los que un cliente usualmente usa para identificar el servicio deseado son, el *Service ID* (Servicio de identificación) y el *Class ID List* (Lista de identificación de clase).

3.6.1.1. Servicio de identificación

El servicio de identificación es un número que permite identificar los servicios que se desean ejecutar. Este número es asignado al momento de ser creada la aplicación por el desarrollador, los puertos no deben ser asignados al momento de diseñar, únicamente se debe asignar el identificador de servicio. El identificador posee 128 bits y es denominado como un UUID (*Universally Unique Identifiers,* Identificador universal único).

El servicio de identificación está pensado para aplicaciones personalizadas construidas por un solo grupo de desarrolladores. Para poder distinguir las aplicaciones y las clases de aplicaciones que realizan las mismas tareas se introducen un segundo UUID, el cuál es llamado *Service Class ID*.

3.6.1.2. Servicio de identificación de clase

Con este servicio se pueden ejecutar aplicaciones comunes, desarrollados por diferentes compañías pero que realizan las mismas tareas. Las compañías que deseen compartir aplicaciones deben acordar en el identificador de servicio de clase que usarán.

Existen UUIDs reservados para identificar clases de servicios predefinidos, para protocolos de transporte y para perfiles. Los 96 bits menos significativos son los mismos para todos los UUIDs reservados por lo que únicamente quedan 32 bits para identificar cada UUID. En la tabla XI se muestran algunos UUIDs *Bluetooth*.

Tabla XI. **UUIDs reservados para servicios predefinidos**

UUIDs Reservados	
SDP	0x0001
RFCOMM	0x0003
L2CAP	0x0100
SDP Server Service Class	0x1000
Serial Port Service Class	0x1101
Headset Service Class	0x1108

Fuente: HUAN, Albert; RUDOLPH, Larry. *Bluetooth essentials for programmers*. p. 20.

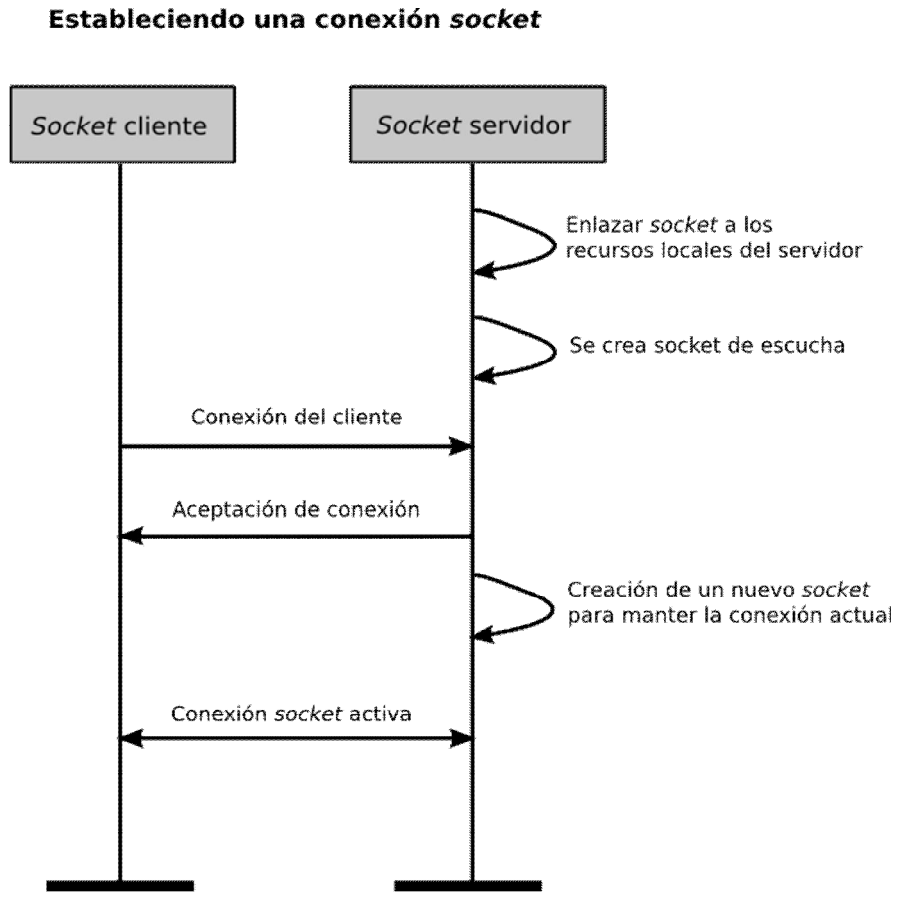
Para obtener el UUID entero de 128 bits se reemplazan los 16 o 32 bits más significativos del UUID *Bluetooth* base por el valor del UUID reservado.

3.7. *Sockets*

El punto final de un enlace de comunicación en programación es conocido como *socket*. Debido que un servidor de aplicación que espera una conexión RFCOMM o L2CAP tiene el mismo comportamiento que un servidor en espera de una conexión TCP o UDP, se puede extender el concepto de *socket* para la programación de aplicaciones para *Bluetooth*. Básicamente, desde el punto de vista de la aplicación todos los datos que pasen por el enlace deben entrar o salir del *socket*.

Un *socket* puede ser usado como cliente o como servidor. En la figura 22 se muestran los pasos para establecer una conexión por medio de *sockets*; los términos cliente y servidor se utilizan para indicar si el *socket* establece conexiones salientes o acepta conexiones entrantes.

Figura 22. **Pasos para establecer una conexión tipo *socket***



Fuente: HUAN, Albert; RUDOLPH, Larry. *Bluetooth essentials for programmers*. p. 22.

Al desarrollar una aplicación el primer paso que se debe dar es la creación de un *socket* en el cual se especifique el protocolo de transporte, generalmente para programación de aplicaciones para *Bluetooth* se crearán *sockets* L2CAP o RFCOMM.

3.7.1. Establecer conexión de un *socket* cliente

El *socket* cliente debe ser creado, posteriormente se debe establecer la conexión con el servidor. Para lograr establecer la conexión, la dirección y el puerto de destino deben ser especificados; el Sistema Operativo se ocupa de los detalles de las capas inferiores como el reservar los recursos en el adaptador *Bluetooth* local, la búsqueda de dispositivos remotos, crear una piconet y establecer una conexión. Una vez el *socket* se encuentre conectado puede ser usado para la transferencia de datos.

3.7.2. Establecer conexión de un *socket* de escucha

El establecer la conexión de un *socket* del lado del servidor requiere una serie de pasos a seguir. Primero, la aplicación debe enlazar el *socket* a los recursos locales, es decir, especificar el adaptador *Bluetooth* y el número de puerto a utilizar.

Posteriormente, se debe colocar el *socket* en modo de escucha para que acepte las peticiones de conexiones entrantes; finalmente se obtiene un *socket* conectado el que servirá para la transferencia de datos.

El *socket* de escucha que es creado en primera instancia por la aplicación nunca puede ser utilizado para la transferencia de datos. Al momento que el *socket* del servidor establece una conexión se crea un nuevo *socket* que representará esta nueva conexión y el *socket* de escucha regresa a esperar nuevas solicitudes de conexión. La aplicación usará el nuevo *socket* creado para comunicarse y transferir datos con el cliente.

4. DISEÑO DEL PUNTO DE ACCESO *BLUETOOTH*

4.1. Funcionamiento básico del punto de acceso

Se requiere que el punto de acceso sea capaz de distribuir información a dispositivos portátiles que posean la capacidad de almacenar dicha información y que estén equipados con un adaptador *Bluetooth* para realizar la conexión entre dispositivos.

El sistema debe ser capaz de entregar la información al dispositivo destino por medio de dos métodos diferentes; estos métodos de entrega se describen a continuación:

- Entrega bajo demanda: el usuario del dispositivo portátil debe enviar al sistema un código que le permita obtener la información deseada. Este código debe ser conocido con anterioridad por el usuario y debe cumplir con el formato que el sistema requiera poder entender la petición del usuario.

- Entrega por difusión selectiva: el sistema realizará una búsqueda de dispositivos que se encuentren en el área de cobertura y que posean la capacidad de recibir la información a enviar. Solicitará permiso para el envío de dicha información, si la solicitud es aceptada se realiza la transferencia de datos. Para evitar el envío repetitivo a los mismos usuarios se debe manejar una lista negra (*blacklist*) dinámica que permita identificar los usuarios que ya han recibido información en un período de tiempo determinado.

La información a transferir pueden ser archivos de audio comprimidos, imágenes, fotografías, vídeos, archivos PDF, etcétera. Es preciso tomar en cuenta que si se transfieren archivos multimedia a dispositivos móviles con ciertas características; los archivos deben cumplir con las especificaciones correctas de codificación, tamaño o formato para que puedan ser visualizados en él, de lo contrario es posible que solo puedan funcionar como dispositivos de almacenamiento.

4.2. *Hardware*

La aplicación que se encargue de ejecutar todos los procesos necesarios para que el punto de acceso sea funcional puede ser alojada en cualquier tipo de ordenador que soporte las especificaciones del *software* que se utilizará.

Sin embargo, debido a que el sistema *Bluetooth* es una tecnología de bajo consumo de energía y que permite crear redes *ad hoc* se debe considerar utilizar el *hardware* que permita cumplir con los objetivos del sistema y las funciones que este debe desempeñar.

A continuación se describen las especificaciones del *hardware* que se recomienda utilizar para esta aplicación ya que cumple con las condiciones de energía, desempeño y tamaño para su utilización como punto de acceso.

4.2.1. Microprocesador Intel® Atom™

Intel Atom™ es una de las líneas de microprocesadores x86 de la marca Intel. Estos procesadores ofrecen un menor consumo de energía, de 0,6 a 2,5 W, y gracias a su proceso de fabricación de 45 nm permite un diminuto tamaño de 26 mm^2.

La arquitectura del procesador permite ejecutar hasta dos instrucciones por ciclo e implementan el conjunto de instrucciones x86. Al igual que muchos otros microprocesadores de ese tipo, antes de ejecutar las instrucciones x86 las traduce a operaciones internas más simples denominadas micro-ops. La mayoría de las instrucciones que producen más de una micro-ops es significativamente menor que en otras arquitecturas.

Las micro-ops internas del procesador Atom pueden contener tanto carga como almacenamiento de memoria en relación con una operación de la unidad aritmética lógica del microprocesador (ALU, *Arithmetic Logic Unit* por sus siglas en ingles); permitiendo un rendimiento bueno con solo dos ALUs de enteros y sin reordenamiento de instrucciones.

El procesador Intel Atom Z530 es un procesador de la serie Z500, esta serie cuenta con circuitos del tamaño de una moneda y consumen muy poca energía.

La serie Z500 de procesadores Intel® tiene velocidades disponibles entre 0,8 GHz y 1,86 GHz. El modelo Z530 puede prescindir del uso de ventiladores durante su funcionamiento. Las especificaciones del procesador Z530 se describen en la tabla XII.

Tabla XII. **Especificaciones del procesador Z530 Intel® Atom™**

Característica	Descripción
Cantidad de núcleos	1
Cantidad de hilos	2
Velocidad de reloj	1,6 GHz
L2 Cache	512 KB
Relacion Bus/Core	12
Velocidad FSB	533 MHz
Velocidad de paridad	No
Set de instrucciones	32-bit
Litografía	45 nm
TDP máxima	2 W
Rango de voltaje VID	0,75 - 1,1 V
Tamaño del encapsulado	13 mm x 14 mm
Numero de transistores	47 millones
Estados disponibles	Si
Tecnologia de monitoreo de temperatura	Si
Tecnologia Hyper-Threading Intel®	Si
Tecnologia VT-x Intel®	Si
Tecnología mejorada Speedstep de Intel®	Si
Conmutación basada en demanda Intel®	Si
Intel® Execute Disable Bit	Si

Fuente: <http://ark.intel.com/products/35463/Intel-Atom-Processor-Z530-(512K-Cache-1_60-GHz-533-MHz-FSB)>. [Consulta: julio de 2010].

Estos procesadores son los más pequeños que Intel ha diseñado y utilizan los transistores más pequeños que puedan existir, los cuales son utilizados especialmente para dispositivos de Internet y computadores de bajo costo y bajo consumo de energía.

Estas características hacen del microprocesador Atom una de las mejores opciones para utilizarlo como el motor principal del sistema.

4.2.2. Placa base o tarjeta madre

La placa base es una tarjeta de circuito impreso en donde se conectan todas las partes de una computadora. La placa base posee una serie de circuitos integrados como el *chipset*, la memoria RAM, buses de expansión, entre otros.

La placa base posee un *software* llamado BIOS, el cual se encuentra instalado en una memoria de solo lectura. Este *software* permite realizar las funciones básicas, como pruebas de dispositivos, vídeo y manejo de teclado, reconocimiento de dispositivos y carga del Sistema Operativo. En una placa base estándar se podrán encontrar uno o varios conectores de alimentación, el *socket* para el CPU, conectores para la memoria RAM, el *chipset*, un reloj, la memoria CMOS, la pila del CMOS, la BIOS, varios tipos de bus, conectores de entrada y salida y los *slots* de expansión.

Generalmente, estas tarjetas están instaladas dentro de una caja o chasis ya sea de plástico o metal. Estas cajas cuentan con paneles para conectar dispositivos externos y para la interconexión de algunos elementos internos.

Para fines de desarrollar el sistema propuesto se escogió la placa base fabricada por la empresa CompuLab.

Esta placa base cuenta con el microprocesador propuesto en la sección anterior, Intel Atom Z530. La empresa ofrece un sistema completo en un chasis de metal que se ajusta perfectamente a las necesidades del proyecto. En la tabla XIII se describen las especificaciones del sistema que ofrece esta compañía.

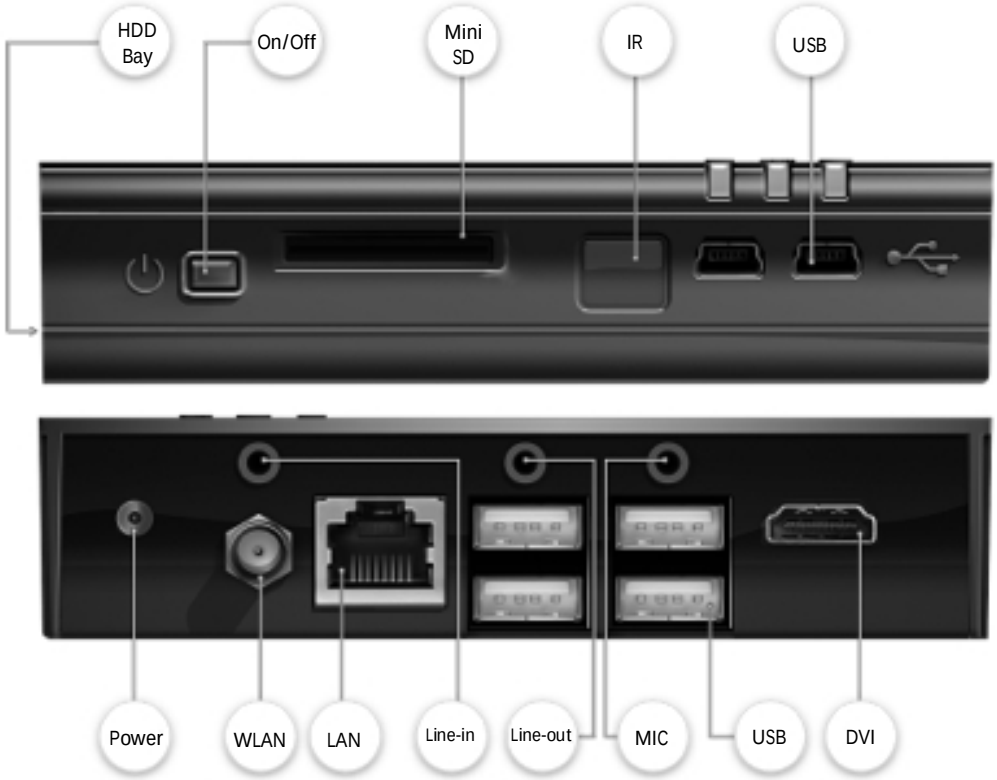
Tabla XIII. **Especificaciones de la placa base CompuLab**

Característica	Descripción
CPU	Intel Atom Z530 1,6GHz
Chipset	Intel US15W SCH
Memory	1GB DDR2-533 en-placa
Storage	Bahia interna para 2,5" SATA HDD miniSD socket
Display & Graphics	Aceleración gráfica Intel GMA500
Audio	Salida, entrada, micrófono
Networking	1000 BaseT Ethernet
USB	6 USB 2,0 puertos de alta velocidad
IR	Receptor IR programable
BIOS	Phoenix BIOS
Chasis	100% de aluminio
Dimensiones	101 x 115 x 27 mm
Peso	370 gramos
Temperatura de operación	0 - 45 °C con HD 0 - 70 °C con SSD
Alimentación	6W con baja carga en CPU 8W con CPU a toda capacidad 0,5W en standby

Fuente: <http://www.fit-pc.com/web/fit-pc/fit-pc2-specifications/>. [Consulta: julio de 2010].

En la figura 23 se muestran las imágenes del chasis que almacena la placa base y sus respectivos puertos para la conexión de elementos externos, sistema de alimentación, botón de encendido y *Leds* indicadores de estado.

Figura 23. **Chasis propuesto para el sistema *Bluetooth***



Fuente: <http://www.fit-pc.com/web/fit-pc/fit-pc2-specifications/>. [Consulta: agosto de 2010].

4.2.3. Adaptadores *Bluetooth*

Un adaptador puede ser un dispositivo físico o un componente de *software* que convierte los datos transmitidos de una forma a otra. Actualmente, muchas de las tarjetas madre para dispositivos portátiles traen incorporadas un adaptador *Bluetooth* que permite la transmisión de datos de forma inalámbrica. Mucho de estos sistemas incrustados no poseen las características físicas y lógicas que se desean para el sistema.

Por ejemplo, un adaptador *Bluetooth* incrustado en un computador portátil o en un teléfono móvil generalmente será de clase 2, es decir, que tendrá un rango aproximado de 10 metros.

Se quiere que el punto de acceso tenga un rango de alcance un tanto mayor, por lo que se propone utilizar un adaptador clase 1 de aproximadamente 100 metros de alcance, esto permitirá que todo dispositivo que tenga capacidad de establecer conexiones por este medio y que se encuentre en esta área de cobertura podrá comunicarse con el punto de acceso.

Para este fin se propone utilizar el adaptador BTA-6210 de la compañía Cirago. El adaptador cumple con muchos de los requisitos necesarios para el diseño y su tamaño es ideal para integrarlo al sistema de CompuLab. En la figura 24 se muestra la imagen del adaptador BTA-6210 y en la tabla XIV se listan las características de este dispositivo.

Figura 24. **Adaptador *Bluetooth* BTA-6210**

Fuente: <http://cirago.com/wordpress/products/bluetoothadapters/bta6210/>. [Consulta: agosto de 2010].

Tabla XIV. **Características del adaptador BTA-6210**

Característica	Descripción
Estándar	Bluetooth 2.1 EDR - Clase 1
EDR	Compatible con ambos modos, 2 Mbps y 3 Mbps
Sistemas Operativos que soporta	Vista, XP, 2000, Mac OS 10.1.4 o superiores, Linux
Rango de operación	100 metros
Frecuencia de operación	2,4GHz - 2,4835 GHz banda ISM
Potencia RF de salida	+15 dBm
Fuente de alimentación	Alimentación por USB, 5V
Sensibilidad del receptor	-90 dBm
Soporte de Piconet y Scatternet	Si
Dimensiones	0,5" x 0,95" x 0,125"

Fuente: <http://cirago.com/wordpress/products/bluetoothadapters/bta6210/>. [Consulta: agosto de 2010].

A continuación se listan algunos de los perfiles que este adaptador soporta:

- DUN
- FAX
- SPP
- HID
- HCRP
- FTP
- OPP
- HSP
- HFP
- PAN
- BIP
- A2DP

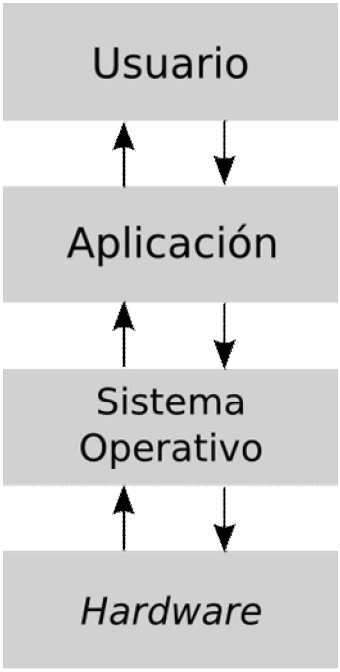
La descripción de cada uno de estos perfiles están descritos en la tabla IX. Los perfiles que interesan para las funciones básicas del sistema son el OPP y el SPP, los cuales se programan a través de comandos de alto nivel que se encuentran dentro de la interfaz de programación de aplicaciones o API por sus siglas en inglés.

4.3. Sistema Operativo

El Sistema Operativo (SO) es el *software* que permitirá que los dispositivos de *hardware* y el programa que ejecutará las tareas que se requieren interactúen entre sí; es decir, el Sistema Operativo es la interfaz de comunicación entre estos elementos. El SO actúa como estación para las aplicaciones que son ejecutadas en el sistema; también se encarga de la gestión y coordinación de actividades y de realizar intercambios de los recursos del sistema.

En la figura 25 se muestra un diagrama que representa la interacción del Sistema Operativo con las diferentes partes de un sistema; se muestra como el usuario interactúa con las diferentes aplicaciones que se alojan en el Sistema Operativo y este es el encargado de interactuar con el *hardware*.

Figura 25. **Interacción del Sistema Operativo**



Fuente: <http://en.wikipedia.org/wiki/Operating_system>. [Consulta: agosto de 2010].

Existen varios SO pero cabe mencionar que los más importantes y usados actualmente son, Windows®, OSX, Linux. A continuación se describirán a grandes rasgos las características del sistema Linux ya que es uno de los más prominentes ejemplos de SO y es el que se ha escogido como el sistema en el cuál correrán las aplicaciones.

4.3.1. Sistema Operativo Linux

Linux es el principal ejemplo de un Sistema Operativo libre. Es un programa de código abierto y su distribución es libre. Linux está licenciado bajo GNU. Esta licencia está orientada principalmente a proteger la libre distribución, modificación y uso de *software*.

Linux funciona bajo un núcleo monolítico donde los controladores de dispositivos y las extensiones del núcleo normalmente se ejecutan en un espacio reservado con libre acceso al *hardware*. Los controladores de dispositivos y las extensiones al núcleo se pueden cargar y descargar fácilmente de forma modular.

Existen diferentes distribuciones Linux que se pueden descargar gratuitamente de la red. Una distribución de *software* basada en el núcleo Linux y que incluye determinados paquetes de *software* para satisfacer las necesidades de un grupo de usuarios es denominada distribución Linux. Para fines del proyecto se ha escogido la distribución Linux con el nombre de Ubuntu, pero puede utilizarse cualquier distribución con las aplicaciones que se especificarán en la siguiente sección.

Ubuntu es un Sistema Operativo basado en una distribución Linux y es distribuido de forma gratuita. Ubuntu cuenta con versiones que reciben soporte durante cierto período de tiempo para actualizaciones de seguridad, parches y actualizaciones menores de programas.

Esta distribución posee una gran colección de aplicaciones funcionales para la configuración de todo el sistema. Ubuntu es conocido por su facilidad de uso y las aplicaciones orientadas al usuario final por lo que es una de las distribuciones que tiene un mayor ritmo de crecimiento que cualquier otra. Para abril de 2010 se estimó que habían 12 millones de usuarios para esta distribución.

4.4. APIs y lenguaje de programación

Ya teniendo el Sistema Operativo sobre el que se va a trabajar, se deben escoger varias interfaces de programación de aplicaciones (APIs) que faciliten la interacción entre la aplicación del Sistema Operativo y el adaptador *Bluetooth* por medio de un lenguaje de alto nivel. Así también, se debe elegir un lenguaje de programación robusto, fácil y efectivo con el cual se desarrollen las aplicaciones que se ejecutarán dentro del sistema y que puede interpretar las librerías de los APIs a utilizar.

A continuación se describirán las características del lenguaje de programación y de los APIs a utilizar; también se describirá el procedimiento para comprobar si están instalados en el sistema.

4.4.1. *Python*

Python es un lenguaje de programación que posee una sintaxis muy limpia y que favorece un código legible. Es un lenguaje interpretado o de script y entre alguna de sus características se pueden mencionar las siguientes: tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos.

4.4.1.1. Lenguaje interpretado

Un lenguaje interpretado se ejecuta utilizando un programa intermediario llamado generalmente intérprete. Un lenguaje interpretado no compila el código a lenguaje máquina para que ejecute directamente una computadora. Los lenguajes interpretados son más flexibles y portátiles que los lenguajes compilados.

Python comparte mucha de las características de los lenguajes compilados por lo que se podría decir que es un lenguaje semiinterpretado que genera un pseudocódigo que se ejecutará en sucesivas ocasiones.

4.4.1.2. Tipado dinámico

Esta característica se refiere a que no es necesario declarar el tipo de datos que una variable va a contener. El tipo de la variable será determinado durante la ejecución según el valor que se asigne a ella; el tipo de la variable puede cambiar si se asigna un valor de otro tipo.

4.4.1.3. Fuertemente tipado

Una variable no podrá ser tratada como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo antes de asignarle un nuevo valor. Esta característica permite que sea menos propenso a errores.

4.4.1.4. Multiplataforma

El intérprete para Python está disponible para varios Sistemas Operativos, entre ellos, se pueden mencionar Unix, Solaris, Linux, DOS, Windows, Mac OSX, entre otros. De esta forma, si no se utilizan librerías específicas, un programa puede ser ejecutado en todos estos sistemas sin grandes cambios.

4.4.1.5. Orientado a objetos

Los lenguajes de programación orientado a objetos, se basan en el paradigma de programación en donde los conceptos del mundo real son definidos dentro de un programa en clases y objetos. Cuando un programa de este tipo es ejecutado los diferentes objetos definidos dentro de él interactúan entre sí.

4.4.2. *PyBluez*

Bluez es un módulo del kernel de Linux que permite al Sistema Operativo soportar el *stack* de protocolos y capas principales de la tecnología *Bluetooth*. Su principal tarea es implementar las especificaciones *Bluetooth* para comunicación inalámbrica entre dispositivos para Sistemas Operativos Linux.

Adicionalmente al *stack* básico, existen utilidades de bajo nivel que permiten la interacción con los adaptadores *Bluetooth*. Para acceder a los recursos *Bluetooth* del sistema por medio de *Python* se debe contar con la extensión *PyBluez*. *PyBluez* soporta los protocolos de transporte RFCOMM, L2CAP, SCO y HCI en sistemas Linux.

4.4.3. *LightBlue*

LightBlue es un API multiplataforma para *Python* que provee un simple acceso para ejecutar operaciones a través del adaptador *Bluetooth*. El API se encuentra disponible para diferentes plataformas, como por ejemplo, Mac OS X, Linux, Python para Nokia Serie 60 entre otros. Algunas de las características de *LightBlue* son:

- Descubrimiento de dispositivos y servicios
- Interfaces estándar para *sockets* RFCOMM y L2CAP
- Publicación de servicios para RFCOMM y OBEX
- Información local de dispositivos

4.5. Recursos del sistema

Antes de desarrollar las aplicaciones se debe verificar que los recursos de *hardware* y *software* que se vayan a utilizar estén instalados y funcionando correctamente.

Para este fin se describen los pasos para verificar estos recursos y se explicará el proceso de instalación en dado caso no se encuentren instalados.

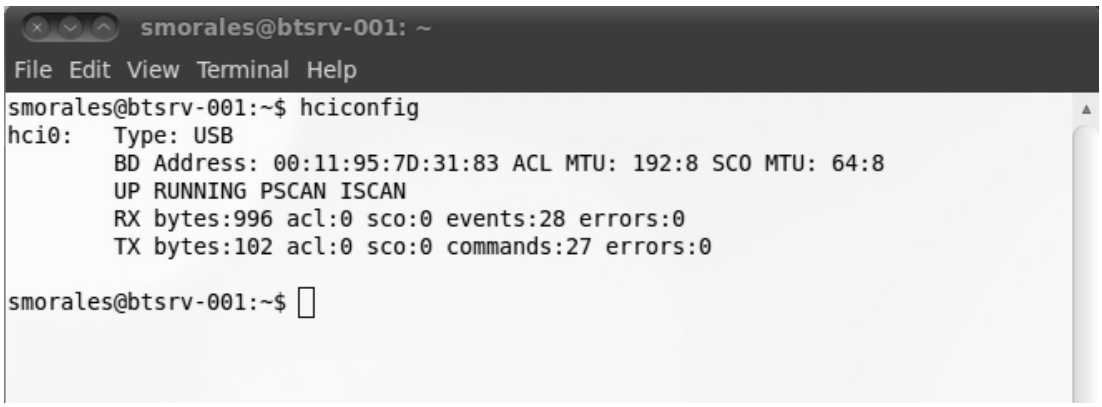
4.5.1. Verificar propiedades del adaptador *Bluetooth*

Antes de empezar a trabajar con el desarrollo de las aplicaciones se debe comprobar el estado del *hardware* a utilizar, es decir, se debe verificar si la configuración de los adaptadores *Bluetooth* que se encuentren instalados en el sistema.

Para este fin se usará uno de los comandos de línea para la terminal Linux que provee el kernel *Bluez*; la descripción de estos comandos se puede consultar en el apéndice A. En esta ocasión se utilizará el comando hciconfig.

En la figura 26 se muestra la forma de verificar el estado del adaptador. Desde una terminal del sistema se ejecuta el comando hciconfig el cuál desplegará información básica del *hardware* instalado.

Figura 26. **Verificación de adaptadores**



Fuente: elaboración propia.

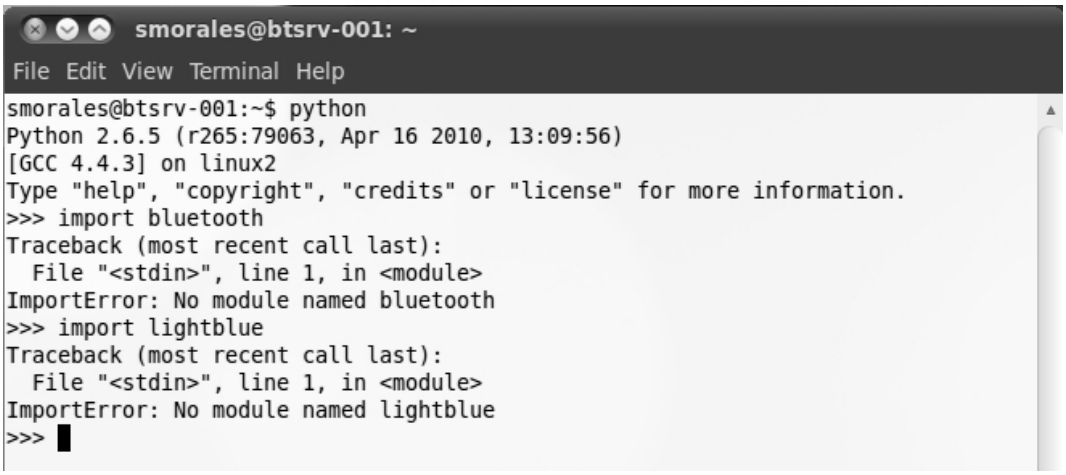
Cada adaptador reconocido por el módulo *Bluez* es mostrado después de ejecutar el comando. En este caso solo existe un adaptador, hci0, el cual tiene la dirección *Bluetooth* 00:11:95:7D:31:83. El texto UP RUNNING indica que el adaptador se encuentra habilitado. A la par se encuentran los textos PSCAN e ISCAN, estos indican que los subestados que controlan la conectividad y visibilidad del dispositivo están activos.

El sistema debe ser visible para que los dispositivos que se encuentren a su alrededor lo puedan localizar; del mismo modo deben ser capaces de establecer una comunicación entre ellos, por lo que es necesario que estos subestados se encuentren siempre activos. En el apéndice se describe como utilizar el comando hciconfig para habilitar y deshabilitar estos estados.

4.5.2. Instalación de herramientas de *software*

Después de verificar que los adaptadores instalados sean reconocidos por el sistema y que su configuración es la deseada se debe comprobar que las herramientas de software que se utilizarán estén disponibles. Para esto se necesita verificar que *Python*, *PyBluez* y *LightBlue* se encuentren instaladas dentro del Sistema Operativo.

Figura 27. **Pasos para verificar los módulos de *software***



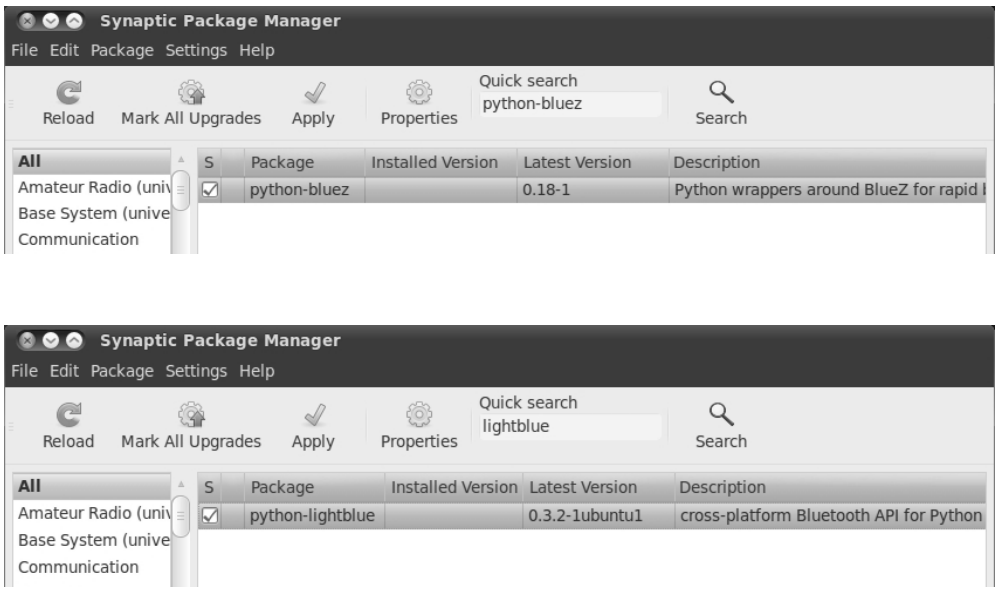
Fuente: elaboración propia.

En la figura 27 se muestran las instrucciones a ejecutar para realizar la verificación respectiva. Las instrucciones deben ser ejecutas desde una terminal del sistema.

Al ejecutar el comando python se ingresa a la aplicación y se puede empezar a trabajar dentro de ella; al ingresar se obtendrá la versión que se tiene instalada, en este caso la versión que se está utilizando es la 2.6.5. Para verificar los módulos de software *PyBluez* y *LightBlue*, dentro de la aplicación *Python* se ejecutan los comandos import *Bluetooth* e *import lightblue*, respectivamente. Se puede dar cuenta que el intérprete retornó un error de importación ya que los módulos que se desean utilizar no fueron encontrados en el sistema, es decir, deben ser instalados.

Para instalar los módulos faltantes se puede hacer por medio de la interfaz gráfica para instalación de paquetes de Ubuntu, *Synaptic*. Para ejectuar esta aplicación se deben ubicar en el menú principal e ir a *System/Administration/Synaptic Package Manager*. En la figura 28 se muestran los pasos a seguir dentro de esta aplicación para instalar correctamente los paquetes faltantes.

Figura 28. **Instalación de paquetes con *Synaptic***

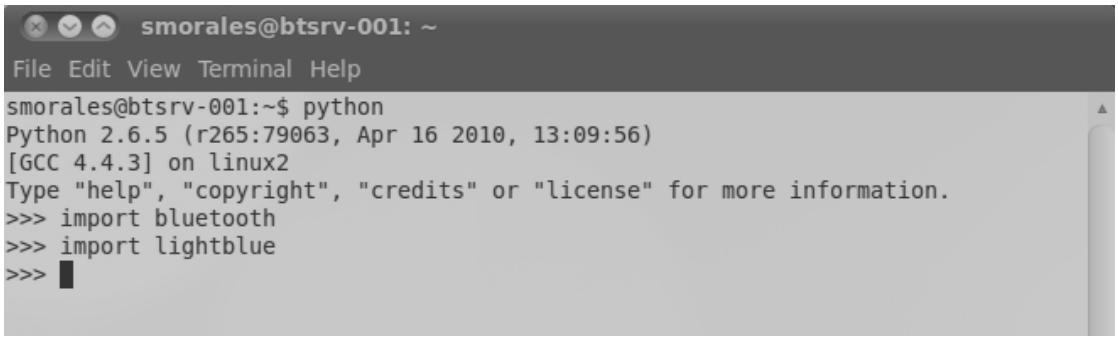


Fuente: elaboración propia.

Al ejecutar *Synaptic* se deben buscar los paquetes python-bluez y python-lightblue con al herramienta de búsqueda que este provee, tal y como se muestra en la figura anterior. Se selecciona la última versión de los paquetes que se deben instalar y se presiona el botón de *Apply*. A continuación se mostrará una ventana emergente con el resumen de los cambios que se aplicarán al sistema, se presiona nuevamente el botón *Apply* de la ventana emergente y se espera a que los paquetes sean instalados. Es necesario tener una conexión a Internet activa para poder descargar los paquetes.

Al finalizar la instalación se puede verificar nuevamente que los módulos estén instalados en el sistema. En la figura 29 se puede observar que al seguir los pasos de verificación descritos anteriormente no se visualizan errores de importación.

Figura 29. **Comprobación de instalación de módulos**



Fuente: elaboración propia.

4.6. Capa de aplicación

El diseño de la capa de aplicación es el que permitirá definir el comportamiento y las funciones que se ejecutarán. Se debe recordar que el punto de acceso debe ser un sistema autónomo capaz de realizar las tareas que se desean de forma eficiente y sin complicaciones.

En esta sección se describirán los diferentes pasos a seguir para desarrollar las funciones que se desea sean soportadas por el punto de acceso.

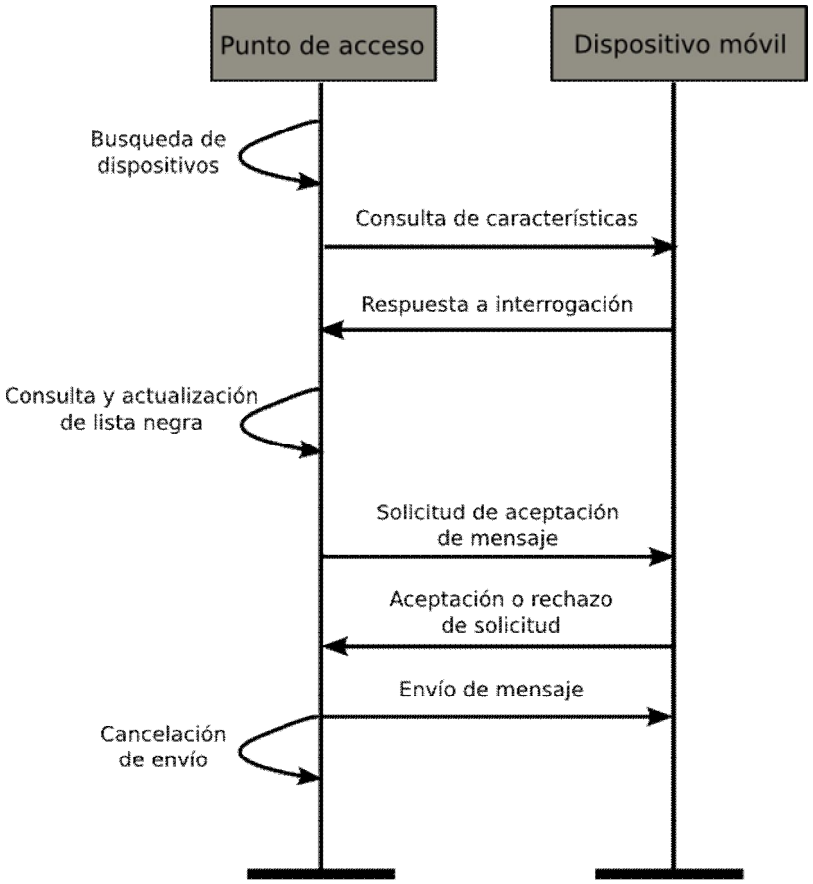
4.6.1. Entrega por difusión selectiva

La difusión selectiva de información permite enviar contenidos a dispositivos móviles que se encuentren dentro de un área de cobertura específica. A cada uno de los dispositivos encontrados por el punto de acceso se les podrá enviar material relacionado con algún tema de interés. Para este fin se supondrá que si el usuario se encuentra dentro del rango de la señal de radio del sistema es porque tiene interés en cierto tema; por ejemplo, si el punto de acceso es instalado dentro de una tienda de deportes y su cobertura es toda el área de la tienda se debe suponer que cada una de las personas con un dispositivo *Bluetooth* que sea detectado estará interesado en el tema.

Debido a que no se desea enviar repetidamente mensajes al mismo usuario es necesario crear una lista dinámica de dispositivos a los cuales ya se les ha enviado un mensaje, estos dispositivos permanecerán en la lista por un tiempo definido; serán eliminados después de este tiempo para que en futuras ocasiones puedan recibir mensajes.

En la figura 30 se muestra el flujo que se sugiere implementar para la entrega de mensajes por este método y poder llevar al mismo tiempo control de los dispositivos que ya han sido interrogados y han recibido mensajes.

Figura 30. **Flujo de entrega de contenido por difusión selectiva**



Fuente: elaboración propia.

El desarrollo de la aplicación dependerá de las funciones específicas que se desean para el sistema. A continuación se describirá brevemente la lógica y las funciones que podrían implementarse para poder cumplir con las funciones básicas bajo el concepto de entrega por difusión selectiva.

Como se muestra en el diagrama anterior, el primer paso es la búsqueda de dispositivos cercanos que se encuentran bajo el área de cobertura. Después de obtener el listado de las direcciones *Bluetooth* se debe asegurar que entre los dispositivos se anuncie el servicio de transferencia de datos OBEX. Este servicio es el que permite la transferencia de archivos entre dispositivos de una forma muy sencilla.

Posteriormente, de identificar los dispositivos que cuenten con los servicios necesarios, se realizará una consulta de los dispositivos que se encuentren almacenados en la lista negra y se compararán las direcciones encontradas contra las de la lista. Finalmente se enviará el mensaje multimedia únicamente a aquellas unidades que no se encuentren en la lista verificada; esta lista se actualiza con la nueva información.

De esta forma se logra transmitir información seleccionada por área de cobertura a los usuarios que se encuentren por esa zona y posean un dispositivo con las características indicadas.

A continuación se muestra un programa en código fuente realizado en *Python* y la explicación de sus funciones principales.

```
#! /usr/bin/python2.6

#Importar librerías

from bluetooth import *
from lightblue import *

import time
import random
import uuid

FTP_TARGET_UUID = uuid.UUID("{F9EC7BC4-953C-11D2-984E- ...
                        ... 525400DC9E09}").bytes
lista = [["","",""]]
nuevosdevs = [["","",""]]
cntservs = 0
cntcomp = 0

### Descubrir dispositivos cercanos ####

devs = discover_devices()
devqnt = len(devs)


#### Verificar dispositivos cercanos existentes ###

if devqnt != 0 :
   for i in range(0,devqnt) :
      devserv = find_service(name="OBEX File Transfer",address=devs[i])
```

```
        if len(devserv) != 0 :
            cntservs = cntservs + 1
            if cntservs == 1 :
              lista[0] = [time.strftime("%H:%M"),devserv[0]["host"],devserv[0]["port"]]
            else:
                lista.append([time.strftime("%H:%M"), ...
                    ... devserv[0]["host"],devserv[0]["port"]])
        else:
           devserv = find_service(name="OBEX FileTransfer",address=devs[i])
           if len(devserv) != 0 :
               cntservs = cntservs + 1
               if cntservs == 1 :
                 lista[0] = [time.strftime("%H:%M"), ...
                     ... devserv[0]["host"],devserv[0]["port"]]
               else:
                   lista.append([time.strftime("%H:%M"), ...
                       ... devserv[0]["host"],devserv[0]["port"]])
    if lista[0][0] != "" :
        listqnt = len(lista)

### Leer archivo de texto (blacklist actual) ###

        f = open("blacklist",'r')
        blklist = f.readlines()
        blkdevs = len(blklist)

### Construir lista en memoria ###

        for m in range(0,blkdevs) :
```

```
        ttime = blklist[m][0]+blklist[m][1]+blklist[m][2]+blklist[m][3]+blklist[m][4]
        tadds = blklist[m][6]+blklist[m][7]+blklist[m][8]+blklist[m][9]+...
                ...blklist[m][10]+blklist[m][11]+blklist[m][12]+blklist[m][13]+...
                ...blklist[m][14]+blklist[m][15]+blklist[m][16]+blklist[m][17]+...
                ...blklist[m][18]+blklist[m][19]+blklist[m][20]+blklist[m][21]+...
                ...blklist[m][22]
        if ord(blklist[m][25]) == 10 :
          tport = blklist[m][24]
        else:
          tport = blklist[m][24]+blklist[m][25]
        if m == 0 :
          memblkl = [{'time':ttime,'btadds':tadds,'port':tport}]
        else:
          memblkl.append({'time':ttime,'btadds':tadds,'port':tport})

### Comparar dispositivos encontrados con la blacklist ###

      for n in range(0,listqnt) :
        cntdevs = 0
        for z in range (0,blkdevs) :
          tlista = str(lista[n][1])
          tmemblkl = str(memblkl[z]["btadds"])
          if tlista != tmemblkl :
            cntdevs = cntdevs + 1
        if cntdevs == blkdevs :
          cntcomp = cntcomp + 1
          if cntcomp == 1 :
            nuevosdevs[0] = [lista[n][0],lista[n][1],lista[n][2]]
```

```
            else:
                nuevosdevs.append([lista[n][0],lista[n][1],lista[n][2]])
        if nuevosdevs[0][0] == "" :
            print "No hay nuevos dispositivos"
            blkupdate = False
        else:
            print "Nuevos Dispositivos"
            blkupdate = True
            print nuevosdevs

### Actualizar blacklist y enviar contenido ###

        if blkupdate == True :
            nuevosdevsqnt = len(nuevosdevs)
            f = open("blacklist", 'a')
            for y in range(0,nuevosdevsqnt) :
                f.write(str(nuevosdevs[y][0])+" ")
                f.write(str(nuevosdevs[y][1])+" ")
                f.write(str(nuevosdevs[y][2])+"\n")
            f.close()
            for p in range(0,nuevosdevsqnt) :
                client = obex.OBEXClient(nuevosdevs[p][1],nuevosdevs[p][2])
                con_resp = client.connect({'target': FTP_TARGET_UUID})
                if con_resp.reason == "OK" :
                    put_resp = client.put({"name": "nameindev"}, file("pathfile", "rb"))
                    timer = 0
                    while put_resp.reason != "OK" and timer < 60 :
                        time.sleep(1)
                        timer = timer + 1
```

```
            if put_resp.reason != "OK" :
                print "Time OUT"
        else: print "No se logro establecer comunicacion"
        dis_resp = client.disconnect()
    else: print "No hay servicios OBEX" + str(devs) + str(lista)
else: print "Fin de secuencia"
```

Las herramientas de *software* que se explicaron anteriormente son utilizadas en este programa. Como primer paso se importan las librerías *bluetooth* y *lightblue*, de esta forma se tendrá acceso a los métodos de éstos módulos.

Para descubrir dispositivos cercanos se utiliza el método *discover_devices()* y el resultado se almacena en una lista. Cada uno de los dispositivos encontrados son interrogados para determinar si poseen los servicios que se requieren, para esto se utiliza el comando *find_service(name="OBEX File Transfer",address=btaddress)*.

El análisis de las listas se realiza con los métodos y rutinas estándar de *Python*. La línea de código *client = obex.OBEXClient ("btaddress","puerto")* permite crear un objeto *OBEXClient*.

Para enviar el contenido a su destino se utilizan los métodos *connect()* y *put()*; el primero establece un enlace activo entre el servidor y el cliente mientras que el segundo se encarga de enviar el contenido. La sintaxis sería la siguiente; *client.connect({'target': "constante_de_UUID"})* y *client.put({"name": "NOMBRE"}, file("nombre del archivo", "rb"))*.

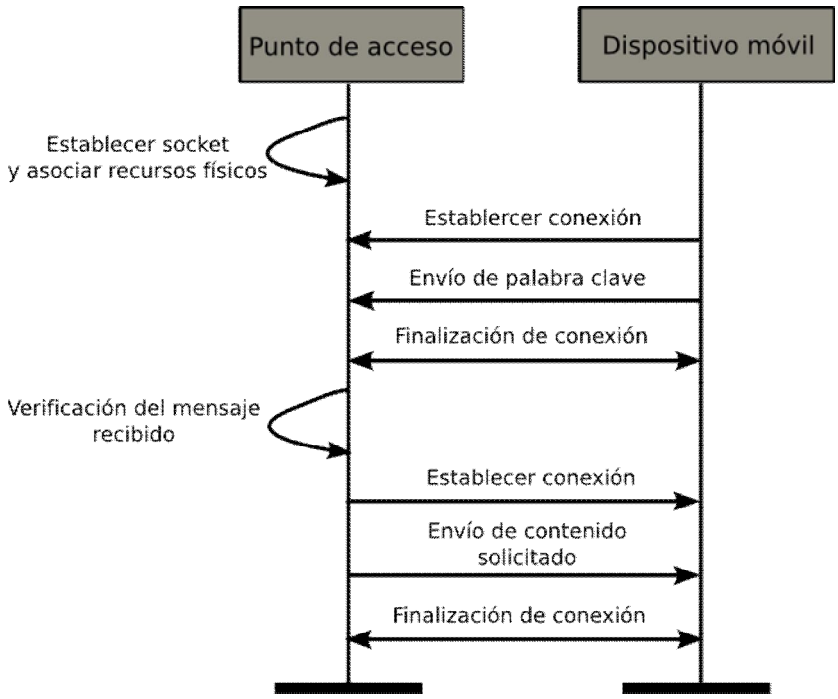
Esta rutina no maneja errores y únicamente permite enviar contenido a un solo dispositivo a la vez; por lo que se debe tomar únicamente como referencia y punto de partida para aplicaciones más complejas.

4.6.2. Envío bajo demanda

Este modelo de envío de información permite al usuario solicitar al servidor contenido específico. De esta forma se le presentará al usuario una lista de opciones que contiene diferentes contenidos multimedia. La lista podrá ser enviada por difusión selectiva o puede ser publicada en un medio físico.

El modelo permite que los usuarios sean los que decidan que tipo de información estarán recibiendo por lo que puede ser más efectivo para aplicaciones donde el contenido puede ser muy amplio por lo que puede ser más complicado concentrar la información a cierto público objetivo.

Figura 31. **Flujo de entrega de contenido bajo demanda**



Fuente: elaboración propia.

En la figura 31 se muestran los pasos que se sugieren implementar para el desarrollo de una aplicación que cumpla con este modelo de entrega. Para que un dispositivo móvil pueda enviar información al servidor, este debe tener habilitado un *socket* asociado al recurso físico que se desea utilizar. Con el *socket* habilitado cualquier dispositivo podrá enviar la información en el formato requerido, en este caso el archivo debe ser un archivo de texto, por lo que el usuario puede escribir la palabra clave en una aplicación para tomar notas o en una tarjeta de negocios virtual y enviarla al servidor. El servidor analizará la palabra clave y si es encontrada enviará el contenido que le corresponde; posteriormente se finaliza la conexión entre dispositivos.

El código fuente para esta aplicación se muestra a continuación, se explicarán la funciones principales que acceden a los recursos *Bluetooth* del sistema.

```
#! /usr/bin/python2.6

from bluetooth import *
from lightblue import *

import time
import uuid

FTP_TARGET_UUID = uuid.UUID("{F9EC7BC4-953C-11D2-984E-...
                        ...525400DC9E09}").bytes

#### Asignación de contenido a la palabra clave ###

keyasig = [{'promo1':"media/alone.jpg",'promo2':"media/sigue.gif"}]

### Crear socket, publicar servicio OBEX ####

s = socket()
s.bind(("", 0))
advertise("PromoServer", s, OBEX)
client = obex.recvfile(s,"reqinst1")
s.close()
```

```
### Leer archivo de texto recibido ###

f = open("reqinst1", 'r')
text = f.readlines()
textqnt = len(text)
f.close()
key = False

### Verificar que la palabra clave se encuentre dentro del texto ###

for n in range(0,textqnt-1):
    ltext = text[n].lower()
    pos = ltext.find("promo")
    if pos != -1 :
       keyword = ltext[pos:pos+6]
       key = True
       break
if key == True :
   time.sleep(15)

### Buscar servicios OBEX y puerto del dispositivo cliente ###

   devserv = find_service(name="OBEX File Transfer",address=client[0])
   cport = devserv[0]["port"]
   clientobx = obex.OBEXClient(client[0],cport)
   conresp = clientobx.connect({'target': FTP_TARGET_UUID})
   if conresp.reason == "OK" :
      path = keyasig[0][keyword]
```

```
### Enviar contenido al dispositivo solicitante ###

    putresp = clientobx.put({"name": keyword}, file(path, "rb"))
  else: print "No se logro establecer comunicacion"
  clientobx.disconnect()
else : print "No Keyword"
```

El código anterior crea un *socket* para poder recibir datos de un dispositivo cercano. Para fines de optimizar recursos, se deben crear diferentes instancias de esta rutina para que el sistema pueda aceptar solicitudes simultaneas utilizando la mayoría de los recursos de *hardware* que se posean.

Al igual que el código para difusión selectiva, se importan los módulos *Bluetooth* y *lightblue*. La línea de código *s = socket()* crea un *socket bluetooth*; y para asignarle un recurso del sistema se utliza el método *bind*, *s.bind((“”,0= 0))*. Posteriormente, se pública el servicio con el comando *advertise(“Nombre del servicio”, s, OBEX)*. Para esperar la solicitud del cliente y recibir el mensaje se utiliza la función *obex.recvfile(s,”Nombre_del_archivo”)*; al establecerse comunicación entre las unidades la función retorna la dirección *Bluetooth* del cliente.

Posteriormente, se lee el archivo y se verifica la palabra clave con rutinas básicas de programación. Para enviar el contenido multimedia que corresponde se utilizan los métodos *connect()* y *put()* que se explicaron anteriormente.

CONCLUSIONES

1. El sistema *Bluetooth* ofrece servicios que habilitan conexiones entre dispositivos y permiten el intercambio de datos entre ellos. El sistema principal consiste en diferentes partes básicas; un transceptor de radiofrecuencia para la transmisión y recepción de señales, un sistema de banda base que establece los procedimientos para establecer un enlace físico y una pila de protocolo que permite la interoperabilidad entre dispositivos.

2. El sistema *Bluetooth* se divide en cuatro capas de operación de la siguiente forma:

- Capa *core* o del sistema central se encarga de las funciones principales para los dispositivos que se encuentran en una piconet.

- Capa de reemplazo de cables que permite la creación de puertos seriales virtuales.

- Capa de control telefónico para poder configurar llamadas de datos, de voz y controlar los teléfonos móviles.

- Capa de protocolos adoptados para la interoperabilidad con otras tecnologías ya existentes.

3. En el modelo cliente/servidor el servidor es una aplicación que provee servicios a los usuarios de la red mediante recepción de peticiones. El cliente simplemente solicita el uso de esos servicios enviando las peticiones correspondientes al servidor usando un protocolo que ambos entiendan. Una aplicación que utilice este modelo consiste en ambas partes, un cliente y un servidor.

4. A cada chip *Bluetooth* se le asigna una dirección de 48-bit. Esta dirección cumple la misma función que las direcciones MAC para las interfaces *Ethernet* la cuál es proveer un número de identificación único para cada dispositivo. Además, de la dirección de 48-bit los dispositivos *Bluetooth* casi siempre tendrán un “nombre amigable”.

5. El punto final de un enlace de comunicación en programación es conocido como *socket*. Debido que un servidor de aplicación que espera una conexión RFCOMM o L2CAP tiene el mismo comportamiento que un servidor en espera de una conexión TCP o UDP, se puede extender el concepto de *socket* para la programación de aplicaciones para *Bluetooth*.

6. Utilizar *software* libre y *hardware* de bajo consumo de energía permiten desarrollar un sistema de difusión de contenido multimedia que cumpla con requerimientos específicos.

7. Las formas de envío que se describieron son unas de las muchas aplicaciones que se pueden implementar en un sistema como este. *Bluetooth* es una tecnología que día a día es implementada en muchos dispositivos de uso cotidiano; por lo que se puede esperar que vayan a surgir necesidades tecnológicas que puedan resolverse con un sistema similar a este.

8. Existen innumerables ventajas al utilizar este tipo de difusión. El sistema permite transmitir contenido continuamente; el mensaje a enviar puede ser cualquier tipo de formato multimedia. La conexión no es invasiva debido a que se solicita autorización para entregar la información. Se identifica a quien se le ha entregado antes de manera que no se le entregue 2 veces la misma información.

RECOMENDACIONES

1. Es necesario hacer énfasis en la seguridad y privacidad de la información colectada por el sistema. Se debe tener cierta responsabilidad al momento que los usuarios de dispositivos móviles acepten la conexión y permitan total acceso al mismo; por lo que solo se deben realizar operaciones que hayan sido notificadas y aceptadas por el usuario.

2. Es aconsejable que el *software* utilizado cumpla con las características de *software* libre. En Internet se podrán encontrar muchos recursos de aplicaciones ya desarrolladas o elementos que ayuden a su elaboración. Si en algún momento se utiliza *software* propietario se deben pagar las licencias que correspondan por su uso.

3. Se debe considerar adecuadamente el lugar de instalación de los equipos ya que deben cumplir con las normas que el fabricante haya establecido. Se requiere que la alimentación esté adecuadamente protegida, que los equipos estén alejados de cualquier tipo de interferencia y protegidos de cualquier evento climático que pueda afectar su funcionamiento.

4. Es importante optimizar los recursos de radio del sistema, es decir, se debe aprovechar cada uno de los puertos de comunicación que el sistema provee. Por lo que es importante mantener en observación estos recursos y dimensionar la capacidad para que siempre posea canales disponibles.

5. El implementar estadísticas permitirá llevar un mejor control de los recursos del sistema y se podrán verificar a quien se ha enviado información, si esta ha sido recibida o no por el usuario, etcétera.

6. Es conveniente optimizar las líneas de código propuestas con rutinas para el manejo de errores. Debido a que existen diferentes fabricantes de dispositivos con *Bluetooth* se deben considerar las características específicas de cada uno de ellos.

7. En integraciones más complejas, donde existan varios puntos de acceso para extender el área de cobertura, es necesaria la implementación de un servidor que funcione como administrador de todos los elementos en la red.

BIBLIOGRAFÍA

1. ANDERSON, Fritz. *Xcode 3 Unleashed*. Indianapolis, Indiana: Sams, 2008. 534 p. ISBN: 978-0-321-55263-1.

2. APPLEBY, Doris; VANDERKOPPLE, Julius. *Lenguajes de Programación: paradigma y práctica*. 2ª ed. México: McGraw-Hill, 1998. 492 p. ISBN: 970-10-1945-8.

3. BAKKER, D.M.; MCMICHAEL, Diane. *Bluetooth end to end.* Nueva York: Hungry Minds, 2002. 309 p. ISBN: 0-7645-4887-5.

4. CARRERAS, Cesar. *Claves de difusión de exposiciones. Narrowcasting (difusión selectiva) vs. broadcasting (divulgación)* [en línea]. España. <http://www.cervantesvirtual.com>. [Consulta: junio de 2009].

5. CARTER, James. *Bluetooth Bluez* [en línea]. Los Angeles, febrero 2007. <http://www.math.ucla.edu>. [Consulta: abril de 2010].

6. *Core V2.1 + EDR* [en línea]. Kirkland, (Wa): Bluetooth SIG, julio 2007. <http://www.bluetooth.com>. [Consulta: julio de 2009].

7. GONZÁLEZ, Raúl. *Phython para todos* [en línea]. España. <http://mundogeek.net/>. [Consulta: junio de 2010].

8. HUAN, Albert; RUDOLPH, Larry. *Bluetooth essentials for programmers*. Nueva York: Cambridge University Press, 2007. 198 p. ISBN: 978-0-511-35583-7.

9. LAM, Bea. *LightBlue API documentation* [en línea]: <http://lightblue.sourceforge.net/doc/index.html>. [Consulta: mayo de 2010].

10. LI, Sing; KNUDSEN, Jonathan. *Beginning J2ME Platform: from novice to professional*. 3ª ed. Estados Unidos: Apress, 2005. 456 p. ISBN: 978-1-59059-479-7.

11. PARZIALE, Lydia; BRITT, David. *TCP/IP tutorial and technical overview*. 8ª ed. Estados Unidos: Redbooks, 2006. 974 p. ISBN: 0738494682.

12. *Python v2.6.5 documentation* [en línea]. Phyton™, marzo 2010. <http://docs.python.org>. [Consulta: julio de 2010].

APÉNDICE

Apéndice 1. **Herramientas GNU/LINUX para *Bluetooth***

El *Kernel* de *bluetooth* para Linux posee herramientas que permiten gestionar el estado y los servicios de los enlaces creados entre el sistema y otros dispositivos. A continuación se describe la funciones y la sintaxis de algunos comandos que *Bluez* proporciona y que son indispensables para configurar el servicio *bluetooth* en una computadora.

- hciconfig

  Es utilizado para configurar las propiedades básicas de un adaptador *Bluetooth*. La sintaxis es la siguiente:

  # hciconfig <dispositivo> <comando> <argumentos…>

  Para configurar los subestados de búsqueda o paginación se debe ejecutar el comando con la siguiente sintaxis:

  # hciconfig hic0 noscan

  Para acceder al texto de ayuda, ejectuar # hciconfig -h

- hcitool

  Este comando es utilizado para detectar dispositivos cercanos y también para probar y mostrar información de las conexiones *Bluetooth* de bajo nivel. Para mostrar los comandos que se pueden utilizar ejecutar # hcitool.

- sdptool

  Se utiliza para consultar los SDP's de los dispositivos cercanos y para configurar los servicios SDP que ofrece el computador local. Para obtener el texto de ayuda y los comandos que se pueden utilizar, ejecutar # sdptool.

- hcidump

  Se utiliza para depurar las conexiones y la transferencia de datos que realiza el computador local. Permite interceptar los paquetes enviados y recibidos. El comando debe ejecutarse con permisos de administrador. Para mostrar el texto de ayuda, ejecutar # hcidump –h.