Universidad Nacional Autónoma de México

Tesis para obtener el título de Ingeniero en Computación

Implementación de sistema de comunicación Bluetooth con Java

Autor: Juan Carlos García Amaral

Las abejas no merecerían fama si no convirtieran lo que encontraron en algo mejor y nuevo.

Petrarca

Agradecimientos

El trabajo que he realizado, ha sido producto del apoyo de mucha gente e instituciones involucradas. Principalmente, agradezco a mi familia, quienes me han enseñado los principios que he debido seguir para llegar hasta aquí.

Agradezco a la Universidad Nacional Autónoma de México con su Facultad de Ingeniería, ya que, ahí fue donde he recibido la instrucción necesaria, entre otros muchos valores, que me han servido en el ámbito tanto personal y humano, como en el profesional. Además considero que me amplió la visión de muchos aspectos de la vida cotidiana.

Agradezco también, que con el apoyo del Instituto de Tecnología del Distrito Federal, del Instituto de Ingeniería de la UNAM y del Centro de Ciencias Aplicadas y Desarrollo Tecnológico, recibí la orientación y apoyo de diversas maneras para poder realizar el presente proyecto.

Agradezco a mis profesores que me compartieron sus conocimientos, me ayudaron para crecer profesionalmente y me ayudaron para poder concretar lo aprendido durante mis estudios en la Facultad. Principalmente, el Dr. Tupak Fernández García del CCADET, quien aportó las ideas para este proyecto y me fue orientado para la obtención de los objetivos del mismo. Me transmitió algo muy importante, la investigación de las cosas para que, con base en ello, formar mi propio conocimiento y a su vez la capacidad de poder transmitirlo.

También agradezco al Dr. José Luis Fernández Zayas y al Ing. Norberto Chargoy del Valle, quienes me brindaron una oportunidad en el Instituto de Ingeniería de la UNAM para la realización de mi servicio social y un foro para poder ir aprendiendo a formar mi conocimiento a través de mi trabajo de tesis, todo esto con la finalidad de desarrollarme profesionalmente.

Tabla de contenido

Introducción

Este documento se centra en la evaluación del rendimiento de la tecnología inalámbrica Bluetooth y su aplicación en un escenario real. La herramienta que he desarrollado sirve para el intercambio de información y datos en un área de trabajo.

El motivo para realizar este estudio es, utilizar dos tipos de tecnologías usadas comúnmente en la actualidad. Estas son, la tecnología de comunicación inalámbrica Bluetooth y dispositivos móviles. Gran parte de los dispositivos telefónicos móviles incluyen una manera de comunicación vía Bluetooth.

A partir de saber cómo funcionan estas tecnologías juntas, desarrollé un sistema de comunicación capaz de intercambiar información. En este caso particular, es texto y el envío de archivos por parte del servidor.

El desarrollo consiste en un sistema de comunicación Bluetooth en Java con la estructura cliente-servidor. La entidad cliente es una aplicación móvil capaz de conectarse e intercambiar información (texto) con servidor. Mientras que el servidor hace una búsqueda de dispositivos Bluetooth que se encuentren alrededor que les envíe archivos vía Bluetooth. A su vez que, reciba información proveniente de la aplicación del dispositivo móvil. La información recibida se almacena en una base de datos.

El estudio de los escenarios reales servirá para determinar sus características y poderlos extender a escenarios con estas características. Un ejemplo de ello es que este trabajo fue pensado para implementar un sistema de comunicación en un laboratorio. Un dispositivo que está por desarrollarse enviará información desde un microscopio electrónico a los celulares de quienes trabajen en determinado proyecto.

Otro objetivo de este trabajo es que sirva como base para proyectos que usen las tecnologías móviles en conjunto con Bluetooth. Los conceptos podrán aplicarse a otros lenguajes de comunicación, siempre y cuando cumplan con los conceptos que aquí se mencionan.

1.1 Objetivo

Conocer la tecnología Bluetooth, dado que es un tipo de tecnología que se puede tener acceso con relativa facilidad. En el caso de los dispositivos móviles, la mayoría de ellos tienen como característica de fábrica el uso de Bluetooth como medio para transmitir y recibir archivos, es por ello que pretendemos abarcar diversas variantes que existen con este tipo de tecnología.

1.2 Objetivos Particulares

- Desarrollar una aplicación que sea capaz de establecer una conexión Bluetooth con el esquema cliente-servidor y que intercambie información y datos.
- Analizar los modos de conexión de la tecnología Bluetooth.
- Analizar escenarios donde comúnmente se utiliza este medio de comunicación inalámbrica.

1.3 Motivación

Lo que motivó a hacer este trabajo fue la necesidad de comunicar vía Bluetooth desde una computadora con el esquema servidor a un dispositivo móvil en un laboratorio, cuya área de trabajo cuenta con características similares a las que se plantean en este trabajo. Con este trabajo, pretendo establecer una base teórica para este proyecto, pues falta el desarrollo de la parte electrónica y en aquellos proyectos donde intervengan tecnologías móviles combinadas con Bluetooth y que cumplan con las características tratadas en el trabajo. A su vez he querido dar un enfoque de la comunicación Bluetooth hacia los dispositivos móviles, pues me motivó la tendencia en cuanto a su utilización.

Es decir, en este trabajo se plantea un escenario de trabajo que es común, también por esa razón elegí la tecnología Java, pues esta plataforma puede ser montada en una gran gama de sistemas. A lo largo del documento serán detallados estos conceptos y a continuación mencionamos algunos proyectos que se han realizado bajo este marco de trabajo.

1.4 Antecedentes

Actualmente las comunicaciones han ido tomando mucha importancia en muchas de nuestras actividades cotidianas y el uso de la información se ha dirigido hacia puntos donde antes no imaginábamos. Tal es la importancia, que las comunicaciones se han vuelto indispensables en muchos escenarios de la vida cotidiana. En el ámbito socioeconómico sería casi imposible imaginar una realidad como la presente sin la presencia de las comunicaciones y el manejo de información.

Los métodos y dispositivos utilizados para las comunicaciones y el manejo de la información han evolucionado natural y notoriamente con el avance de la tecnología y el tiempo. Desde ser aparatos de gran tamaño conectados por cables en grandes instalaciones hasta aparatos que caben en la palma de la mano si ningún tipo de cable, que son capaces de almacenar grandes cantidades de información y comunicarse a cualquier parte del mundo en un solo instante.

La tendencia de las telecomunicaciones y las tecnologías de la información, hoy en día, van dirigidas hacia las conexiones inalámbricas. Los diferentes tipos de tecnologías inalámbricas nos brindan un espectro muy amplio de posibilidades para su utilización, siendo la más

utilizada la conexión WiFi, pero en nuestro caso, hemos decidido considerar otra tecnología también muy utilizada y conocida, es el caso de la tecnología inalámbrica Bluetooth.

El concepto de Bluetooth fue creado por Ericsson en 1997 y se acercó a varios fabricantes de dispositivos electrónicos portátiles para discutir el desarrollo de esta nueva tecnología inalámbrica de corto alcance, y en 1998 cinco empresas clave, Ericsson, IBM, Intel, Nokia y Toshiba formaron el *Special Interest Group (SIG),* para coordinar el desarrollo y promoción de la tecnología Bluetooth. Oficialmente se anunció en mayo de 1998, y el SIG lanzó la versión 1.0 de la especificación en julio de 1999 [Miller01].

Bluetooth SIG es una asociación sin fines de lucro fundada en 1998. Bluetooth SIG, Inc. No manufactura ni vende productos Bluetooth. Las funciones principales de Bluetooth SIG son publicar las especificaciones Bluetooth, administrar el programa de capacitación, proteger las marcas comerciales Bluetooth y promover la tecnología inalámbrica Bluetooth [BluetoothSIG].

Los fabricantes que integran el *SIG* se encargan de establecer las normas para la comunicación Bluetooth y publican la especificación de Bluetooth. El sitio oficial es *www.bluetooth.org*, es éste el sitio donde se encuentran las fuentes de información oficial correspondiente al tema de Bluetooth.

El SIG es un grupo de fabricantes integrado por:

- 3Com
- Ericsson
- Intel
- Agere
- Microsoft
- Motorola
- Nokia
- Toshiba

La versión 1.0 de Bluetooth fue lanzada como un estándar para la industria de comunicaciones inalámbricas de voz y datos de corto alcance. Algunas de las aplicaciones servían para la telefonía sin cables, acceso inalámbrico a impresoras, faxes o redes LAN, redes de área personal, entre otras [Schwingenschoegl00].

Bluetooth fue llamada de esa manera en honor al rey vikingo y danés Harald Blatand (Bluetooth en inglés) que vivió en el Siglo X. Harald Blatand unificó y controló los pueblos de Dinamarca y Noruega (de ahí el nombre, por el afán de unir dispositivos). El rey obtuvo su

nombre pues comía demasiados arándanos de color azul y sus dientes se teñían de ese color [Rodrin06].

Uno de los objetivos para el desarrollo de esta tecnología, fue eliminar el uso de cables en áreas limitadas de trabajo. En ocasiones resulta incómoda la presencia de cables, por ejemplo, en nuestro escritorio al usar la computadora, conectada con el monitor, el teclado, la impresora, etc. Cada uno de estos dispositivos poseen cables y en consecuencia cuando tenemos muchos aparatos, habrán muchos de ellos ocupando espacio que nos podría ser útil para otras cosas y de ahí que, Bluetooth surge por la necesidad de eliminar cables en área limitadas de trabajo, por ejemplo, en oficinas, laboratorios, etc.

Bluetooth es una de tecnología para comunicaciones inalámbricas de corto alcance. Es simple pues su manejo es transparente al usuario, es decir, éste ya no se debe preocupar por configuraciones adicionales, los dispositivos Bluetooth son capaces de conectarse entre sí. Es una tecnología segura. La seguridad en Bluetooth incluye funciones de autenticación y confidencialidad.

Las señales son omni-direccionales y pueden atravesar paredes y maletines. Dispositivos de comunicación no deben estar alineados y no necesita una línea de visión sin obstáculos. Bluetooth utiliza salto de frecuencia. Su enfoque de espectro ensanchado reduce en gran medida el riesgo de que la comunicación pueda ser interceptada.

Es una tecnología que actualmente es muy utilizada en muchos dispositivos alrededor de todo el mundo. Puede uno encontrar miles de millones de dispositivos que van desde teléfonos móviles y computadoras hasta dispositivos médicos y productos de entretenimiento en el hogar. Su objetivo es reemplazar los cables que conectan a los dispositivos, al tiempo que mantiene altos niveles de seguridad [BluetoothBasics11].

Bluetooth es una tecnología inalámbrica y automática. Uno no tiene que hacer un seguimiento de cables, conectores y conexiones, y no se necesita hacer nada especial para iniciar las comunicaciones. Los dispositivos pueden encontrarse unos a otros de forma automática y comenzar a comunicarse sin intervención del usuario, en principio se requiere autenticación. Bluetooth es barato. La banda *industrial, científica y médica (ISM)* que utiliza Bluetooth está regulada, pero sin licencia. Los gobiernos se han unido en una sola norma, por lo que es posible utilizar los mismos dispositivos prácticamente donde quiera que vaya, y no es necesario para obtener el permiso legal por adelantado para empezar a usar la tecnología [Mahmoud03].

La expansión de esta tecnología se ha extendido hacia diversos escenarios, esto se debe a que desde su desarrollo fue concebido como un estándar, entonces ahora su difusión es extensa. Un ejemplo es que actualmente la mayoría de los dispositivos móviles que hay en el mercado cuentan con esta tecnología de fábrica. Por esta difusión ha surgido la necesidad de crear

aplicaciones que exploten las características de esta tecnología. Existen muchas aplicaciones que ilustran la manera en que Bluetooth puede ser de utilidad. Las aplicaciones pueden ser muy variadas pero se basan sobre la misma estructura, es decir, el intercambio de información en áreas de trabajo limitadas a cierta distancia. Las aplicaciones son básicamente de comunicaciones y manejo de información en áreas pequeñas de trabajo y puede tener gran utilidad en muy diversos escenarios, por ejemplo en el ámbito académico, en algunas actividades financieras, en algunos deportes, etc. Más adelante mencionaremos algunos ejemplos de estas aplicaciones.

Bluetooth se encarga tanto de datos como de voz. Su habilidad para manejar los dos tipos de transmisiones al mismo tiempo hace posible las innovaciones tales como un teléfono móvil manos libres Bluetooth para voz con aplicaciones que imprimen en fax, y que se sincronizan las libretas de direcciones de su PDA, ordenador portátil y su teléfono celular.

En estos escenarios la tecnología inalámbrica Bluetooth puede resultar de mucha utilidad, por ejemplo imagine que es capaz de usar su teléfono habilitado con Bluetooth móvil para bloquear y desbloquear su automóvil, operar la puerta del garaje, y controlar su TV, VCR, reproductor de DVD y otros aparatos de consumo. Si usted quiere hacer ese tipo de control a disposición de sus usuarios, tendrá que ser capaz de escribir aplicaciones de Bluetooth que personalicen estos aparatos, y utilizarlos de una manera que permita a los usuarios a descargarlas a un teléfono celular, por ejemplo. Bluetooth y *Java 2 Micro Edition (J2ME)* pueden trabajar juntos para lograr esta visión unificada. Bluetooth permite a los dispositivos comunicarse de forma inalámbrica y J2ME permite escribir aplicaciones personalizadas y desplegar en dispositivos móviles [Mahmoud03].

La tecnología Bluetooth como modelo de negocio tiene un gran potencial asociado a ella, basado en dos puntos clave. En primer lugar, Bluetooth está diseñado para una gran variedad de aplicaciones inalámbricas *Personal Area Network (PAN)*. En segundo lugar, el costo y el tamaño de la tecnología se incluyen en dispositivos móviles por defecto. Este punto significa que habrá oportunidades para los dispositivos de red adicionales, incluyendo puntos de acceso, para aplicaciones Bluetooth [Swarnalatha10].

Existen muchas aplicaciones de la tecnología Bluetooth, a continuación mencionamos algunos ejemplos de ellas. En primer lugar, tenemos el ejemplo de aplicación en dispositivos Bluetooth que se usan comúnmente para el envío de archivos de alguna computadora a un teléfono, de teléfono a teléfono, o bien para sincronizar la agenda de algún dispositivo móvil con una computadora. Otro ejemplo de su utilidad es el uso de Bluetooth en los dispositivos de manos libres en celulares.

Como podemos apreciar la conectividad Bluetooth se puede llevar a cabo tomando en cuenta dos factores fundamentales:

- Que estén presentes dos dispositivos Bluetooth.
- Que estos dispositivos puedan encontrarse entre sí en un área determinada.

La tecnología Bluetooth por sus características, más adelante las detallaremos, provee una manera simple de conectar a los dispositivos que encuentren en la proximidad. Puede encontrarse presente en teléfonos móviles, PDAs, laptops, impresoras, consolas de video juegos y cámaras digitales [Boja10].

La aplicación de la tecnología Bluetooth está presente en consolas de video juegos y otros dispositivos electrónicos muy populares, tenemos el ejemplo del uso de Bluetooth en el control remoto de la consola de video juegos wii. Si recordamos bien, el objetivo de esta tecnología es eliminar el uso de cables. En una consola de video juegos esto resultó demasiado práctico e innovador para los usuarios. Otro ejemplo famoso de la utilización de la tecnología Bluetooth que es muy conocido es en los dispositivos de Apple, los periféricos que manejan las computadoras (teclado y mouse) más recientes son conectados vía Bluetooth. También las nuevas PC-tablets que empiezan a tomar auge, los de la marca Apple usan, después de Wi-Fi (802.11a/b/g/n), Bluetooth Clase 2.1+EDR, USB 2.0 como forma de conectividad.

Las aplicaciones sirven para algo más que el entretenimiento, la tecnología Bluetooth no sólo se limita a los ejemplos que mencionamos antes, sino que las aplicaciones pueden llevarse a cabo en escenarios que van desde lo académico, lo financiero, el entretenimiento, etc. Por ejemplo, en el ámbito académico hay trabajos que muestran cómo puede ser utilizada la tecnología. El estudio llamado “Evaluación del rendimiento de la herramienta Bluetooth en un salón de clases” [Davidrajuh09]. Esta aplicación consiste en que los estudiantes envían respuestas de algunos test mientras se está desarrollando la clase y un sistema informático hace las evaluaciones de forma automática, es decir, la respuesta se envía al momento que es escogida de un menú. El menú está contenido en una aplicación encuesta enviada al dispositivo móvil de cada alumno, el objetivo es utilizar la aplicación capaz de intercambiar información de esta manera y de forma automática, esto ayuda para agilizar las clases en términos del tiempo requerido para su desarrollo.

En este mismo escenario de la educación (*M- Learning*), M-learning se ha convertido en un objetivo como área de aplicación atractivo para aplicaciones de dispositivos móviles [Boja10].El aprendizaje móvil se ha visualizado como una auxiliar en los métodos educativos tradicionales [EuropeanCommission]. Otro ejemplo es una aplicación desarrollada en *Java 2 Micro Edition (J2ME)*, este desarrollo consiste en una herramienta para el control de las asistencias [Boja10]. La utilidad de esta herramienta radica en que se aprovecha el tiempo de la clase, pues no se requiere tomar tiempo de la presentación para el control de las asistencias.

La tecnología Bluetooth por ser inalámbrica entra en la categoría de *red oportunista*, una red oportunista es aquella donde sus nodos se conectan inalámbricamente. Existen aplicaciones

móviles que utilizan este esquema, por ejemplo, para el intercambio de archivos donde un nodo busca las partes de un archivo buscándolo en los nodos que se encuentren en la vecindad y lo recibe punto a punto *(P2P)* [Chavan09].

Otro escenario que mencionábamos anteriormente es el del ámbito financiero. En cualquiera de estos escenarios las aplicaciones Bluetooth pueden estar presentes, pues su fundamento es la conectividad, entonces puede ser usado en cualquier escenario que esté formado por dos o más entidades que cuenten con la tecnología Bluetooth. Tal es el caso de los modos de pago electrónicos (*M-payments*). La tecnología inalámbrica Bluetooth y la amplia difusión de dispositivos móviles permiten establecer nuevos paradigmas para realizar pagos, llamado pago local, permite a dispositivos conectarse fácilmente con máquinas previamente configuradas y realizar las transacciones [Me05].

De igual manera, se pueden hacer algunas transacciones vía Bluetooth, existen sistemas que pueden hacer algunas operaciones bancarias en [Swarnalatha10], existe un ejemplo de una aplicación desarrollada *Java 2 Micro Edition (J2ME)* de un sistema de comunicación Bluetooth para hacer algunas operaciones bancarias como depósitos, retiros, abrir cuentas y consultas de saldo. El objetivo principal de esta aplicación es eliminar las filas en los bancos, así como hacer más cómoda la visita al banco, pues mientras uno se ubique en el área del banco podrá hacer uso de este sistema.

Pero no solo en estos lugares puede utilizarse la tecnología Bluetooth, en el transporte público puede utilizarse. Un ejemplo es el que menciona [Yow10] en esta aplicación programada en *Java 2 Micro Edition (J2ME),* puede recibir información mientras viaja en un autobús del transporte público de Singapur y no sólo información, es posible revisar contenido multimedia y comunicarse en mediante un chat con quienes estén conectados en ese momento.

En los deportes como es el caso del Boxeo, existen en ocasiones dificultades para apreciar adecuadamente el puntaje. La función de los jueces es cuantificar el número de golpes que se ejecutan correctamente, pero esto sabemos es subjetivo. Por eso algunos comités en su preocupación por hacer este deporte más equitativo han buscado soluciones tecnológicas para que la puntación sea objetiva y en cierta forma justa. Se ha desarrollado equipamiento con componentes eléctricos, así como materiales que reciben las señales que indiquen que se está anotando un punto a favor. La manera que se sugiere para enviar la información es Bluetooth, pues es un medio de transmisión inalámbrico [Hahn10].

Con todas las aplicaciones que hemos mencionado nos hemos dado una idea de que esta tecnología puede estar presente en cualquier lugar, siempre y cuando se cumpla que dos dispositivos activos Bluetooth estén en su proximidad.

La idea fundamental en las comunicaciones Bluetooth es el intercambio de información en lugares específicos y limitados por el área en que se encuentran. Es capaz de convivir con otros

tipos de medios de transmisión inalámbricos como puede ser WiFi, considerando que la utilización de este medio de transmisión inalámbrico es de los más comunes que existen. Bluetooth tiene una característica muy particular que le permite hacer esto con gran eficacia *Adaptive Frequency Hopping* (*AFH*). Éste es un protocolo de saltos y propicia que no existan interferencias en la banda donde trabaja Bluetooth, más adelante explicaremos este protocolo con mayores detalles.

De las ventajas con que cuenta la tecnología Bluetooth, es que en la actualidad la mayoría de los dispositivos móviles cuentan con este tipo de tecnología por defecto. Bluetooth es la panacea para la interconexión de aparatos diferentes, eso es lo que mencionan los que están a favor de esta tecnología, sus defensores se esfuerzan por garantizar el 100% interoperabilidad. Otra ventaja del hardware de Bluetooth es el bajo consumo de este. El hardware incluye un chip CMOS que resulta un beneficio por el tamaño y el costo de este, esta es la razón por la que se incluye en los dispositivos móviles y otros de dimensiones similares. [Szweda00].

El boom de los teléfonos móviles demanda muchos tipos de componentes, en particular los dispositivos Bluetooth. La tecnología Bluetooth crece rápidamente. El número de dispositivos Bluetooth distribuidos en el mundo por año ha crecido exponencialmente. La mayoría de esos dispositivos Bluetooth son usados en teléfonos móviles [Boja10]. Bluetooth SIG reveló en 2005 que el mercado ha superado la impresionante marca de 9.5 millones de unidades por semana [IIIVsReview05]. Se estima que 35 millones de chips Bluetooth fueron producidos en 2002, con un aumento estimado de 510 millones en 2006 [Swarnalatha10]. En 2009 existió una proporción estimada del 90% entre el número de dispositivos móviles en todo el mundo y de la población mundial y que la tasa de crecimiento en el mercado de dispositivos móviles de la Unión Europea para 2009 es del 119% [Pocatilu09]. Finalmente, 2 mil millones de dispositivos Bluetooth se espera serán distribuidos alrededor del mundo en 2013 [eeherald11]. Este boom ha propiciado que su utilización haya permeado gran cantidad de aplicaciones que utilizan la tecnología Bluetooth. En la página principal del SIG, podemos observar las innovaciones existentes en el mercado [BluetoothSIG/Mahmoud03].

Todas estas características mencionadas anteriormente nos dan una idea de la utilidad de esta tecnología. Para comprender la tecnología Bluetooth hemos dividido funcionamiento en tres partes esenciales. La primera es el dispositivo físico. Sin la antena que es capaz de recibir, procesar y enviar información, es imposible lograr esta comunicación inalámbrica. Después le sigue el software pues es quien controlará las funciones del dispositivo físico. Finalmente para que se puedan comunicar dos o más dispositivos es necesario que hablen el mismo lenguaje. Esto se consigue con el protocolo de comunicación Bluetooth.

1.5 Esquema del documento

El documento está organizado de manera que se pueda crear una visión general de la tecnología Bluetooth y los alcances que ésta tiene. El documento está compuesto por una parte teórica donde se indica la manera en que funciona la tecnología, después está la parte enfocada al desarrollo de la aplicación y por último un capítulo dondes están las pruebas que hice para conocer su comportamiento en un escenario real.

En el capítulo 1, hice una revisión bibliográfica sobre las características y algunas aplicaciones de la tecnología inalámbrica Bluetooth que hay en la actualidad. En este capítulo están detallados los componentes que conforman a la tecnología Bluetooth. Para facilitar la compresión de estos conceptos, los dividí en tres partes: Hardware, Software y protocolo Bluetooth.

Presento a su vez, en el capítulo 2 en la sección de software, los aspectos para el diseño de la aplicación. Este documento trata de la programación de dispositivos Bluetooth con Java mediante la ***Application programming interface (API)*** desarrollada por el ***Java Community Process (*JCP*)*** y especificada en el ***Java Specification Request JSR-82***. El documento menciona las ***API s*** de Java necesarias para el desarrollo de las aplicaciones de comunicación Bluetooth, el envío de archivos y el almacenamiento de la información recibida por el servidor. Defino las entidades cliente-servidor y describo las herramientas de programación para la comunicación Bluetooth con el lenguaje Java.

En el capítulo 3 de este documento describo el sistema de comunicación Bluetooth programado en Java. Describo a detalle las entidades que lo componen y los elementos de programación que las integran.

En el capítulo 4 son mostrados los resultados de una serie de pruebas para evaluar la comunicación Bluetooth en determinados escenarios que definimos como característicos de un área personal de trabajo.

Al final del documento hay cinco apéndices donde se muestran las funciones en código Java más importantes para el desarrollo de aplicaciones que utilicen comunicaciones Bluetooth y los métodos utilizados para hacer los cálculos en las pruebas de conexión. Existe también al final del documento un glosario que contiene algunos de los conceptos más importantes utilizados en el documento. Estos conceptos son mencionados a lo largo del texto; están remarcados y escritos con letra cursiva.

CAPÍTULO 2. Visión general de la tecnología Bluetooth

En esta sección describimos la tecnología Bluetooth desde tres puntos de vista. Estos son Hardware, Software y Protocolos de comunicación. Los tres puntos de vista integrados entre sí, conforman la tecnología Bluetooth. Al final del capítulo detallamos la comunicaciones Bluetooth porque es el enfoque de este trabajo.

2.1 Hardware

En esta parte describimos los detalles a nivel físico que permiten llevar a cabo una comunicación inalámbrica a través de Bluetooth, tecnología de corto alcance 10m en promedio. El objetivo de la tecnología Bluetooth es reemplazar cables para dispositivos portables o adaptables conservando altos niveles de seguridad.

Las características clave de la tecnología Bluetooth son su robustez, pues posee un protocolo definido y probado; bajo consumo 2.5 mW en las versiones más comunes y bajo costo. La especificación Bluetooth define una estructura uniforme para una amplia gama de dispositivos para conectarse y comunicarse entre ellos.

La estructura y la aceptación global de la tecnología Bluetooth, implica que al menos un dispositivo se puede conectar en todo el mundo con otro dispositivo habilitado localizado en la proximidad del otro.

Estructura de red Bluetooth Topografía. Una red conformada por varios dispositivos Bluetooth se le clasifica como una red de área personal (PAN), al mismo tiempo, a nivel mismo de dispositivos Bluetooth es llamada *Piconet* consiste en que uno o más dispositivos conectados a algún dispositivo master. El master es, en consecuencia, responsable del enrutamiento de los mensajes en la *Piconet*.

Las *Piconets* son establecidas dinámica y automáticamente en tanto los dispositivos Bluetooth que estén habilitados y entren en un radio de proximidad que se puedan conectar fácilmente cuando y donde sea conveniente.

Cada dispositivo en una piconet puede, también comunicarse con hasta otros 7 dispositivos contenidos en una piconet simple y cada dispositivo puede pertenecer a muchas piconets simultáneamente.

Una ventaja fundamental con la tecnología inalámbrica Bluetooth es la capacidad de manejar simultáneamente transmisiones de voz y datos. Proveen a los usuarios una gran cantidad de soluciones innovadores, tales como manos libres para llamadas telefónicas, impresoras y faxes inalámbricos y sincronización entre la PC y teléfonos móviles, sólo por nombrar algunos.

El rango de la tecnología Bluetooth es dependiente de la aplicación. La especificación principal de esta tecnología indica un rango de 10 m pero no hay límite establecido y los fabricantes

pueden ajustar sus implementaciones para proveer el rango capaz de soportar sus mismas aplicaciones.

2.1.1 Especificación Bluetooth

La especificación Bluetooth provee a los desarrolladores del producto las definiciones de la tecnología. En los siguientes puntos mencionamos las características físicas de la tecnología Bluetooth. En los siguientes subtemas mencionamos la información correspondiente al Special Interest Group [BluetoothSIG].

2.1.1.1 Espectro radioeléctrico

La tecnología opera en la banda ISM (industrial, científica y médica) sin licencia de 2.4 a 2.485 GHz.

Esta banda es usada también por: teléfonos inalámbricos de 2.4 GHz, redes inalámbricas del estándar IEEE 802.11 (por ejemplo WiFi), redes inalámbricas del estándar HomeRF para interconexión de dispositivos caseros, algunos monitores para bebés, puertas automáticas de garaje, sistemas de comunicaciones inalámbricas de transporte urbano y suburbano, incluyendo muchos radios de emergencia, algunos tipos de comunicaciones en gobiernos locales de España, Francia y Japón y hornos de microondas [Miller01].

Usa un *espectro ensanchado por salto de frecuencia (FHSS)*, con una señal full-dúplex a una velocidad nominal de 1600 saltos /s. La banda de 2.4 GHz ISM está disponible y sin licencia en la mayoría de los países.

Para minimizar la complejidad del transmisor, se utiliza una modulación de frecuencia binaria. La tasa de transferencia de símbolos es de 1 MS/s véase ***MS/s MSs***, que admite una velocidad de transmisión de 1 Megabit por segundo (Mbps) en el modo de transferencia básica y una velocidad de transmisión aérea total de 2 a 3 Mbps en el modo de transferencia de datos mejorada. Estos modos se conocen como transferencia básica y transferencia de datos mejorada, respectivamente.

2.1.1.2 Interferencia

Una de las características de la tecnología Bluetooth es su *Adaptive Frequency Hopping (**AFH**)* que fue diseñada para reducir la interferencia entre tecnologías inalámbricas que comparten la banda de 2.4 GHz. AFH funciona dentro del espectro para tomar ventaja de la frecuencia disponible. Está hecho por la tecnología para detectar otros dispositivos en el espectro y evitar las frecuencias que se están utilizando. El sistema emplea un transmisor de salto de frecuencia para contrarrestar las interferencias y la pérdida de intensidad, y cuenta con gran número de portadoras de espectro ensanchado por salto de frecuencia (*FHSS*). Estos saltos adaptativos entre 79 frecuencias de intervalos a 1 MHz brindan un alto grado de inmunidad ante interferencias y también permiten una más eficiente transmisión dentro del espectro. Para los

usuarios de la tecnología Bluetooth estos saltos proveen un mejor rendimiento, aun cuando otras tecnologías estas siendo utilizadas junto con la tecnología Bluetooth.

Los saltos son elegidos al azar dentro de un rango de frecuencias asignadas. Los dispositivos habilitados para Bluetooth usan este método "salto", el cambio de frecuencias 1.600 veces por segundo. Como resultado, más dispositivos pueden utilizar una porción del espectro de radio. El riesgo de que un dispositivo como un teléfono celular o algún dispositivo para monitorear bebés pueda interferir con los dispositivos Bluetooth se minimiza, ya que cualquier interferencia en una frecuencia específica tendrá una duración de sólo una fracción de segundo. Bluetooth versión 2.0 + *EDR* , lo último de las versiones de la especificación Bluetooth, usa la tecnología salto adaptable de frecuencia (AFH). AFH permite que los dispositivos Bluetooth midan la calidad de la señal inalámbrica y luego determinar si hay canales mal presente en frecuencias específicas debido a la interferencia de otros dispositivos inalámbricos. Si los canales que no nos son útiles están presentes en una frecuencia específica, el dispositivo Bluetooth ajustará su secuencia de salto de evitarlos. Como resultado, la conexión Bluetooth es más fuerte, más rápida y más fiable.

Para minimizar la complejidad del transmisor, se utiliza una modulación de frecuencia binaria. La tasa de transferencia de símbolos es de 1 MS/s, que admite una velocidad de transmisión de 1 Megabit por segundo (Mbps) en el modo de transferencia básica y una velocidad de transmisión aérea total de 2 a 3 Mbps en el modo de transferencia de datos mejorada. Estos modos se conocen como transferencia básica y transferencia de datos mejorada, respectivamente.

2.1.1.3 Alcance

El rango de distancias depende de cada aplicación y a pesar de que hay un mínimo rango de 10m establecido por la especificación, no hay límite y los fabricantes pueden ajustar sus implementaciones para que soporten en el caso de que estén habilitados. El rango puede variar dependiendo de la clase de radio utilizado en la implementación.

- Clase 3 radios – tienen un rango de hasta 1 m
- Clase 2 radios – comúnmente encontrado en dispositivos móviles – tienen un rango de 10 m
- Clase 1 radio – Utilizado principalmente en casos de uso industrial – tienen un rango de 100 m

2.1.1.4 Alimentación.

Los clase más utilizada es la 2 y consume 2.5 mW de energía. La tecnología Bluetooth está diseñada para tener muy bajo consumo de energía. Esto está indicado en la especificación que permite disminuir el consumo cuando está inactivo. La tecnología Bluetooth en la versión 3.0

de bajo consumo, optimiza el requerimiento de energía ante transferencias de datos altas, consume entre la mitad y una centésima parte del consumo en las versiones anteriores.

La especificación Bluetooth crece a medida que se añaden nuevos perfiles de programación. La intención de la especificación de diseño es tal que los nuevos perfiles de Bluetooth pueden ser construidos en la parte superior de esta API de Java utilizando el lenguaje de programación Java, siempre y cuando la especificación de la capa central no cambia [Jsr82Spec].

2.1.2 Pila de Bluetooth

A nivel físico la pila de Bluetooth es responsable de controlar el dispositivo Bluetooth, es el ***Firmware*** del dispositivo Bluetooth, por lo que necesita inicializar la pila de Bluetooth antes de poder hacer otra cosa. El proceso de inicialización consta de una serie de medidas cuyo objetivo es conseguir que el dispositivo se prepare para la comunicación inalámbrica. La entidad que se encarga de controlar la pila Bluetooth es el *Bluetooth Control Center (BCC)*. La especificación Bluetooth permite la aplicación del *BCC* y el diseño de la de la pila Bluetooth a los distribuidores y proveedores. Por ejemplo, un dispositivo puede ser una aplicación con una interfaz *Graphic User Interface (GUI)*, y otra puede ser una serie de ajustes que no se pueden cambiar por el usuario.

El protocolo de la pila Bluetooth puede ser dividido en dos componentes: El host Bluetooth y el controlador Bluetooth (o módulo de radio Bluetooth). El controlador de interface del host (HCI) provee una interface estándar entre el host Bluetooth y el controlador Bluetooth (módulo de radio). En la Figura 2.1 [Jsr82SpecJsr82Spec] se ilustran las partes que componen el protocolo de la pila Bluetooth. La pila del protocolo Bluetooth puede ser dividida en 4 capas de acuerdo a lo mostrado en la Tabla 2.1 [Jsr82Spec].

Grupos de protocolo	Protocolos en la pila
Protocolos básicos Bluetooth	Banda base, Protocolo gestor de enlaces, L2CAP (Logical Link Control and Adaptation Protocol)* y SDP (Service discovery protocol)*
Protocolo de reemplazamiento de cables	RFCOMM (Radio frequency communication)*
Protocolo de control telefónico	Binario TCS
Protocolos adoptados	***PPP***, UDP/TCP/IP, OBEX (Object exchange)*,

Tabla 2.1 Protocolos y capas en la pila Bluetooth

La capa de banda base habilita el enlace físico de radiofrecuencia entre unidades Bluetooth haciendo una conexión. El protocolo controlador de enlace (LMP Link Manager Protocol) es responsable de llevar a cabo el enlace de dispositivos Bluetooth y manejar los aspectos de seguridad, tales como autenticación y la encriptación de información transmitida. L2CAP (Logical Link Control and Adaptation Protocol, se mencionará más adelante en este mismo capítulo) adapta los protocolos superiores a la capa de banda base. Éste multiplexa las diferentes conexiones lógicas hechas por las capas superiores. Los datos de audio son enrutados directamente hacia y desde la capa de banda base y no a través de L2CAP. SDP (Service discovery protocol) es usado para brindar información del dispositivo, servicios y características de los servicios. RFCOMM emula señales de control y datos de manera serial RS-232 sobre la capa banda base de Bluetooth, proporcionando capacidades de transporte para los servicios de capas superiores que utilizan una interface serial como mecanismo de transporte. Finalmente el binario TCS define la señalización de la llamada de control para el establecimiento de llamadas de voz y datos entre dispositivos Bluetooth.

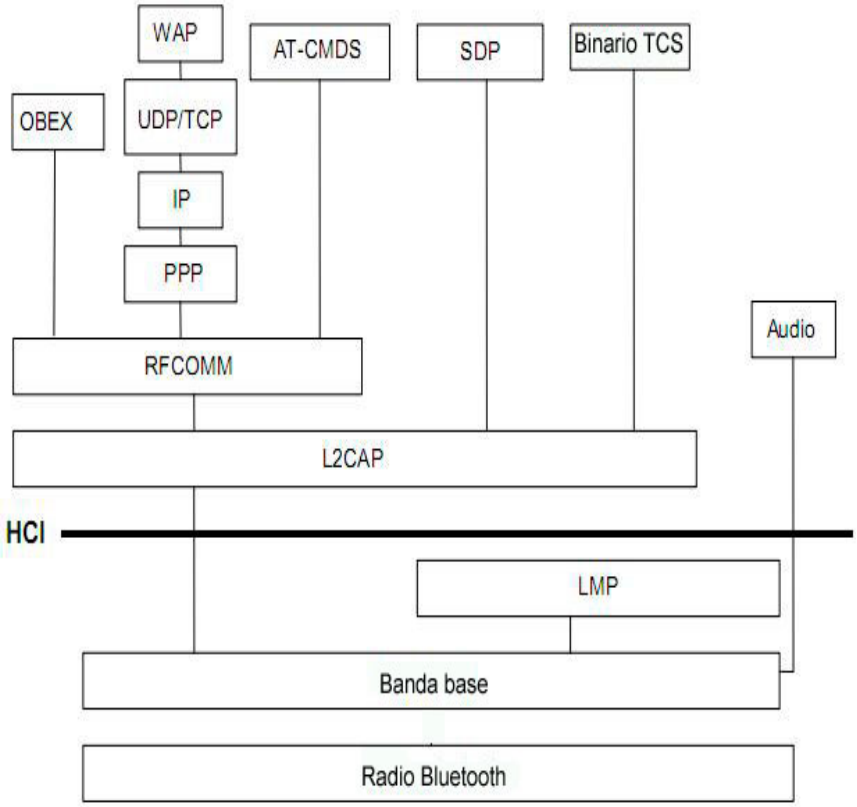


Figura 2.1 Protocolo de Pila de Bluetooth

2.2 Software

Para que el hardware del dispositivo Bluetooth pueda funcionar requiere de un programa que se encargue de controlarlo. En el caso del dispositivo Bluetooth lo que se debe controlar es la pila Bluetooth, hay que inicializarla para que pueda realizar las funciones de transmisión y comunicación a nivel físico. En esta sección mencionaremos cómo se controla un dispositivo Bluetooth.

2.2.1 Controladores de la pila Bluetooth

Para controlar las operaciones de la pila de Bluetooth existen varios tipos de software controladores a nivel de sistema operativo. A continuación mostramos los controladores más comunes de la pila Bluetooth:

Pilas de Bluetooth, implementaciones de propósito general [Wiki10]

Widcomm Es la primera pila que salió para el sistema operativo Windows. Inicialmente desarrollada por Widcomm Inc. Ésta fue adquirida por Broadcom Corporation en abril de 2004, que continúa vendiendo licencias para su uso en multitud de dispositivos Bluetooth.

Existe una API para interactuar con la pila a través de una aplicación genérica.

Pila de Windows. Windows XP incluye una pila integrada a partir del Service Pack 2. Previamente, Microsoft lanzó un ***QFE*** de su pila en el Service Pack 1, con la referencia QFE323183. No obstante, sólo fue distribuida a compañías y no directamente al público. Estas compañías podían utilizar el QFE como parte de los instaladores de sus dispositivos Bluetooth, aunque ya no está soportado por Microsoft.

Windows Vista incluye también una pila integrada que es una expansión de la pila presente en Windows XP. Además del soporte para más perfiles Bluetooth, también soporta el desarrollo de drivers que permite que los desarrolladores externos den soporte a perfiles adicionales, un aspecto que no estaba presente en la pila de Windows XP y sólo permitía el desarrollo por encima de la pila de Microsoft (lo que según algunos retrasó su adopción).

Microsoft no ha lanzado una pila oficial para versiones anteriores de su sistema operativo como Windows 2000 o Windows ME.

Windows Vista y Windows 7 ofrecen una renovación y mejora de la pila presente en sus anteriores sistemas operativos.

Pila Toshiba. Toshiba ha desarrollado su propia pila para Microsoft Windows. Ésta es una pila distinta a la integrada en el sistema operativo. Toshiba vende licencias a OEM´s y se ha

utilizado en algunos portátiles de Dell y Sony. Debe firmarse un acuerdo de confidencialidad para tener acceso a la API.

BlueSoleil. Está desarrollada por IVT Corporation, que se dedica al desarrollo de pilas para dispositivos embebidos y sistemas de escritorio. Está disponible en versiones estándar y VoIP para Microsoft Windows. La API se puede obtener sin coste alguno, y se provee una interfaz de usuario que monitoriza la actividad de la API en tiempo real, lo que ofrece apoyo al desarrollo de software. Actualmente la aplicación es de tipo shareware, por lo que está limitada. Si se desea funcionalidad completa se ha de pagar licencia.

Para las distribuciones **Linux** actualmente hay dos implementaciones:

BlueZ. Incluida oficialmente en el núcleo y desarrollada inicialmente por Qualcomm. Es la pila Bluetooth oficial de Linux. Su meta es lograr una implementación de los estándares inalámbricos Bluetooth para Linux. En 2006, la pila soporta todos los protocolos y niveles de la especificación de base. Está disponible a partir de la versión 2.4.6 del núcleo.

Affix. Desarrollada por Nokia Research Center.

Al momento en que se realizó este trabajo la tecnología Java era la que llevaba la batuta en las tecnologías móviles. En los últimos años, debido a la adquisión de Sun microsystems por parte de Oracle, la tecnología Java tuvo un rezago considerable frente al sistema operativo Android de Google y el iOS de Mac. Éstos sistemas se han convertido en los más populares y en consecuencia su presencia ha aumentado considerablemente en el desarrollo de aplicaciones móviles. Java es un sistema multiplataforma y de fácil aprendizaje y manejo para desarrolladores [Benjuya]. Por esta causa se ha elegido para este trabajo esta plataforma.

Si bien la presencia de Java en iOS y Android ha sido restringida, por cuestiones legales, no deja de ser una buena opción por dos razones esenciales. Es multiplataforma, puede utilizarse sobre Android y iOS y la programación es muy amigable y de fácil instalación.

Para beneplácito de muchos desarrolladores, ha sido en el transcurso del JavaOne de Oracle, que estos días celebra su OpenWorld (2012) en San Francisco, cuando se ha podido presenciar una demo en la que se ha ejecutado JavaFX tanto en un iPad como en un Samsung Galaxy sobre Android. Además, delante de la audiencia tradicional de desarrolladores de Oracle, se ha presentado Avatar, un proyecto que utiliza HTML5 para llevar Java a los dispositivos sobre plataforma iOS. [Ticbeat12]

Además, Oracle ha aprovechado su evento mundial para anunciar también su intención de consolidar su plataforma Java ME (Micro Edition), utilizada para la integración de lenguaje Java en móviles, con Java SE (Stardard Edition).[Ticbeat12]

Por estas razones considero que la plataforma Java, a pesar de su rezago, no debe ser descartada para el desarrollo de tecnología móviles y mucho menos para aplicaciones de escritorio.

2.2.2 Plataforma Java

Para el desarrollo de este trabajo se ha tomado en consideración el uso de la plataforma Java. Java es el lenguaje ideal para aprender informática moderna, porque incorpora todos estos conceptos de un modo estándar, mucho más sencillo y claro que con las citadas extensiones de otros lenguajes. Esto es consecuencia de haber sido diseñado más recientemente y por un único equipo. [García00]

Un factor que consideramos, es que el lenguaje de programación Java es uno de los más utilizados en la actualidad por el paradigma mismo de programación (Orientada a objetos) y que es libre para su implementación. Además hay suficiente documentación disponible para el desarrollo de diversas aplicaciones. Se pueden ir creando librerías y programas que manejen diversas funciones e integrarlos en una sola aplicación con el solo hecho de hacer una llamada a algún objeto de cierta librería. Esto implica que el lenguaje está en constante crecimiento, así como la facilidad para integrarlo y aumentar significativamente las aplicaciones.

Todos estos controladores a nivel de sistema operativo permiten la interacción con los dispositivos Bluetooth. ¿Ahora qué hacer siendo desarrollador, si debemos tener acceso a la pila? Hemos elegido Bluecove, pues esta librería de Java para Bluetooth (JSR-82) de java es la que hay mayor documentación en la comunidad Java. Interactúa con los controladores de MAC OS X, WIDCOMM, BlueSoleil, la pila Bluetooth de Microsoft que se encuentra en Windows XP SP2 o en Windows Vista, WIDCOMM y la pila Bluetooth de Windows Mobile, también puede manejarse junto con diferentes sistemas operativos.

Java es un sistema multiplataforma, tiene la característica muy particular de montar una máquina virtual sobre cualquier sistema operativo, entonces la gama de dispositivos electrónicos en los que puede funcionar es muy amplia. En la Figura 2.2 [Benjuya] es posible observar algunos de los dispositivos electrónicos que pueden utilizarse con la plataforma Java. En nuestro caso utilizamos computadoras y teléfonos móviles.

Figura 2.2 Pila de Protocolos para Bluetooth en Java

2.2.3 Especificación Java (JSR-82)

Cualquier mejora que sea aceptada como estándar en el lenguaje de programación Java debe incorporarse en la comunidad **JCP** *Java Community Process* . El documento para un módulo específico en la plataforma Java es llamado *Java Specification Request JSR.* Este documento incluye todos los métodos que ya han sido programados y que sirven para alguna tarea específica. En el caso de la transmisión Bluetooth, la especificación que se adapta al estándar IEEE 802.15 de transmisiones Bluetooth es la JSR82.

2.2.3.1 JSR82

El documento JSR82, que está en la Java Community Process ***JCP***, contiene todas las adaptaciones hechas al estándar IEEE 802.15 de transmisiones Bluetooth para su implementación en la plataforma Java. Java Specification Request 82 (JSR-82) [Jsr82Spec] define los detalles referentes y sus paquetes para la tecnología inalámbrica Bluetooth para Java 2 Platform (J2SE), Micro Edition (J2ME). El objetivo de este documento es establecer las condiciones para definir la arquitectura y las *API s* asociadas necesarias para permitir un entorno de desarrollo.

Esta API está diseñada para funcionar en la parte superior de la Conexión, Configuración Limitada de Dispositivos (CLDC), descrita en la JSR-30. JSR-30 es un paquete opcional que se puede utilizar para ampliar la capacidad de un perfil de J2ME, como son los dispositivos móviles.

2.2.3.2 Capacidades de JSR 82

La API está destinada a proporcionar las siguientes capacidades [Mahmoud03]:

- Registro de los servicios.
- Detección de los dispositivos y servicios.
- Establecer RFCOMM, L2CAP, y las conexiones entre los dispositivos OBEX.
- El uso de estas conexiones, enviar y recibir datos (comunicación por voz no es compatible en esta especificación).
- Gestionar y controlar las conexiones de comunicación.
- Proporcionar seguridad para estas actividades.

2.2.4 Librería Bluecove

La librería Bluecove es la que cuenta con métodos para el manejo de Bluetooth, contenidos en la especificación JSR-82. BlueCove es de licencia pública *GPL*. BlueCove BlueZ. En la Figura 2.3 [Bluecove] siguiente se muestra el funcionamiento de la librería:

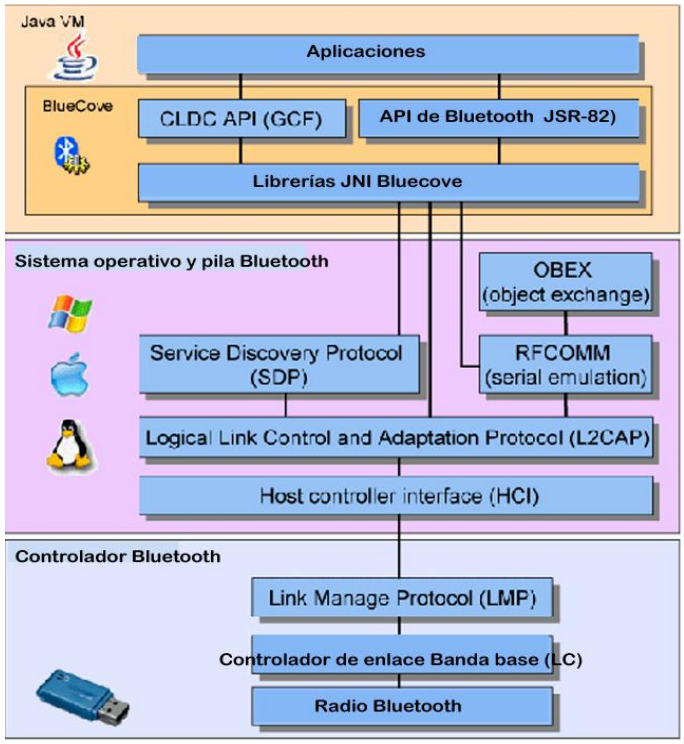


Figura 2.3 Bluecove

El emulador BlueCove JSR-82 es el módulo adicional para BlueCove para simular la pila de Bluetooth. Se puede utilizar en Java Standard Edition (J2SE) 1.1 o posterior. A su vez provee la interface Java JSR82 para los siguientes perfiles Bluetooth:

SDAP – Perfil de aplicación para detección de servicios

RFCOMM – Protocolo de emulación de cable serial

L2CAP – Control de enlace lógico y protocolo de adaptación

OBEX – Perfil de intercambio de objetos genéricos (GOEP), perfil en la parte superior de RFCOMM y TCP

2.3 Protocolos de comunicación Bluetooth

Una vez que ya hemos explicado el funcionamiento del hardware y software, analizaremos en esta sección la manera en que interactúa el software con el hardware, es decir, las características que tienen para utilizar y controlar el dispositivo Bluetooth. Es aquí donde introducimos el concepto de protocolo de comunicación Bluetooth, el protocolo es quien dicta las normas para la comunicación con dispositivos Bluetooth.

El protocolo Bluetooth permite utilizar varios métodos, incluidos RFCOMM y OBEX (Object Exchange) para enviar y recibir archivos entre dispositivos. La dirección del dispositivo remoto al que se desea conectar. Los métodos del protocolo más utilizados son: L2CAP, RFCOMM y OBEX.

2.3.1 Descripción general del funcionamiento del protocolo

Los protocolos son organizados desde el control en la parte física, después se efectúan los enlaces físicos y por último se llevan a cabo los enlaces de manera lógica, a éstos se les denomina protocolos de gestión de enlaces y canales [Bluetooth.com].

Capas de control. Sobre el canal físico hay una capa de enlaces y canales con sus respectivos protocolos de control. La jerarquía de abajo a arriba de los canales y enlaces es la siguiente: canal físico, enlace físico, comunicación lógica, enlace lógico y canal L2CAP.

Enlaces físicos. En el canal físico, se forma un enlace físico entre dos dispositivos que se intercambian paquetes, sea cual sea la dirección. Pero no todos los dispositivos pueden conectarse mediante un enlace físico dentro de la piconet. Existe un enlace físico entre cada dispositivo esclavo y el maestro. En una piconet, no se forman enlaces físicos directos entre los dispositivos esclavos.

Enlaces lógicos. El enlace físico se utiliza como medio de comunicación entre uno o dos enlaces lógicos que admiten tráfico síncrono, asíncrono e isócrono de unidifusión, y tráfico de difusión.

El tráfico de los enlaces lógicos se multiplexa en el enlace físico ocupando las ranuras asignadas por el programador del gestor de recursos.

En la mayoría de dispositivos convencionales, computadoras y teléfonos móviles con Bluetooth el modo OBEX es el más utilizado pues sirve para el intercambio de archivos. El método OBEX, como mencionamos, se utiliza convencionalmente, aunque se puede hacer uso de otros métodos del protocolo Bluetooth para funciones específicas. En la figura 2.4 [Hopkins06] se muestran los protocolos de comunicación Bluetooth y después se explican cada uno de los componentes que lo integran.

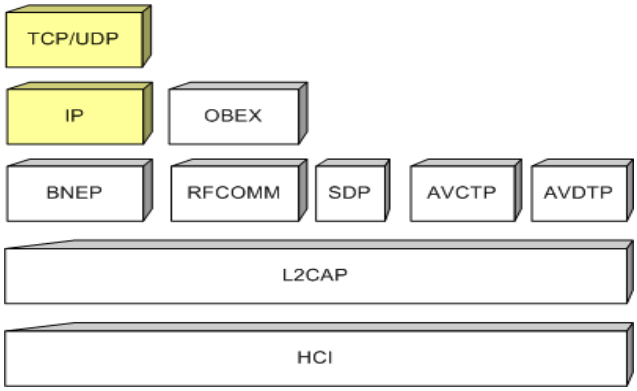


Figura 2.4 Protocolos de comunicación Bluetooth

- ***Host Controller Interface (HCI).*** Es la capa inferior de la pila. Esta capa es, literalmente, la interfaz entre el host (el procesador) y el controlador (el dispositivo Bluetooth).
- ***Logical Link Control and Adaptation Protocol (L2CAP).*** La capa por encima de la HCI es L2CAP, el enlace lógico del controlador del protocolo. Esta capa actúa como el multiplexor de datos para todas las otras capas.
- ***Bluetooth network encapsulation protocol (BNEP).*** La siguiente capa es BNEP, la encapsulación de red Bluetooth Protocolo. Usando BNEP, puede ejecutar otros protocolos de red, como IP, TCP y UDP, a través de Bluetooth.
- ***Radio frequency communication (RFCOMM).*** Es conocido como el *protocolo* de *puerto serie virtual,* ya que permite que un dispositivo Bluetooth pueda simular las funciones de un puerto serie.
- ***Object exchange (OBEX).*** La capa del protocolo OBEX se aplica sobre la capa RFCOMM y es útil cuando se desea transferir los datos como un objeto, por ejemplo, archivos.

- ***Service discovery protocol (SDP).*** Es el descubrimiento de servicios de Protocolo de la capa, que se utiliza cada vez que desea encontrar servicios en un dispositivo Bluetooth remoto.

- ***Audio/video control transport protocol (AVCTP) y Audio/video data transport protocol (AVDTP).*** Las dos últimas capas, y AVCTP AVDTP se utilizan para el *control* y *distribución* de audio y vídeo a través de Bluetooth. AVCTP y AVDTP son relativamente nuevas adiciones al protocolo Bluetooth, sino que se utilizan cuando se desea controlar las funciones de un reproductor de medios o si desea transmitir audio en estéreo.

El protocolo Bluetooth permite utilizar varios métodos, incluidos RFCOMM y Object Exchange (OBEX), para enviar y recibir archivos entre dispositivos. RFCOMM es mejor cuando se quiere enviar y recibir datos de forma típica serial. Por otro lado, tenemos OBEX es excelente cuando se quiere enviar datos como objetos, comúnmente son archivos.

2.4 Conexiones Bluetooth

En esta sección nos ocupamos de las herramientas de programación que hemos utilizado para la realización de nuestro sistema de comunicación vía Bluetooth.

El objetivo de esta aplicación de software es establecer un canal de comunicación entre dos dispositivos Bluetooth, para nuestro objeto de estudio, se cuenta con una aplicación servidor montada bajo el lenguaje de programación Java y un dispositivo cliente móvil montado en J2ME.

De acuerdo a las características de la tecnología inalámbrica Bluetooth, mencionadas en el capítulo anterior, consideramos algunos de estos conceptos para el diseño de un sistema que permita controlar y establecer comunicaciones con la tecnología inalámbrica Bluetooth.

A su vez, como hemos dicho ya, la plataforma java tiene gran flexibilidad, pues es un lenguaje en constante evolución. Desarrolladores van creando aplicaciones de acuerdo a necesidades particulares y estas pueden ser compartidas y utilizadas por otros desarrolladores para hacer otras aplicaciones. La programación orientada a objetos tiene esta funcionalidad.

El sistema consta de dos entidades, es decir, la estructura de la comunicación es cliente-servidor, la parte del servidor sirve para controlar la comunicación que se desee hacer desde un dispositivo móvil, el servidor es quien define los parámetros de la comunicación, tanto los protocolos de comunicación, el orden de conexión con la cual podrán acceder al servicio. El cliente a su vez, está diseñado para realizar una búsqueda de los servicios adecuados para establecer la comunicación y el envío de información.

El desarrollo se llevó a cabo en el entorno integrado de desarrollo (*IDE*) Netbeans v.7.0 desarrollado principalmente en el lenguaje de programación Java, y que permite el desarrollo

de aplicaciones con diferentes lenguajes de programación. En nuestro caso nos sirvió para el desarrollo de nuestra aplicación, pues incluye los paquetes para desarrollo de software (*SDK*) tanto de J2SE, como J2ME que incluyen las librerías que necesitamos para la programación de las dos entidades de nuestro sistema.

El diseño de la aplicación consta de dos entidades. Una que es el servidor, está desarrollado con la versión estándar de Java J2SE, de ahí tomamos todas las librerías, con que cuenta está edición desde las que incluyen las interfaces gráficas y las que se encargan del manejo de la información de entrada y saliente, "java.io"

La otra entidad del sistema es el cliente, programado con el SDK J2ME y este funcionará en los teléfonos móviles que cuenten con estas APIs en su constitución de fábrica y en sus sistemas operativos correspondientes.

Tanto J2SE, como J2ME no cuentan con el software controlador de la pila Bluetooth, aquí es donde usamos la librería Bluecove que permite establecer y controlar las conexiones Bluetooth con Java. A su vez java contiene una librería para establecer canales de comunicación y poder manejarlos, la librería es llamada "java.io". En nuestro caso combinaremos estas aplicaciones previamente establecidas, la primera, Bluecove, recientemente desarrollada y java.io que viene normalmente incluída en el Java Development Kit (*JDK*) por default.

2.4.1.Establecimiento de una conexión

La cuestión fundamental para lograr la comunicación entre dispositivos Bluetooth es llevar a cabo el establecimiento de la conexión. Dos extremos van a acordar con base en un protocolo, las características de la información que será enviada [Alonso09].

La especificación JSR-82 encargada de las conexiones Bluetooth, establece el método:

```
StreamConnection conn = (StreamConnection) Connector.open(address);
```

Con este método logramos realizar una conexión automática a la dirección descrita por una cadena de la clase String de acuerdo con el protocolo indicado. En nuestro caso hemos escogido dos protocolos diferentes para el establecimiento de la conexión.

De acuerdo al protocolo Bluetooth, todas las conexiones se hacen sobre la capa L2CAP que está por encima de la de banda base y se encarga de ofrecer una abstracción de canales de comunicación a las aplicaciones y los servicios. Realiza la segmentación y la unificación de los datos de las aplicaciones, así como la multiplexación y demultiplexación de varios canales a través de un enlace lógico compartido. La capa L2CAP dispone de un canal de control de protocolos a través de la comunicación lógica *ACL* predeterminada. Los datos de la aplicación enviados al protocolo L2CAP pueden transferirse a través de un enlace lógico compatible con el protocolo L2CAP.

2.4.2 Conexión RFCOMM

RFCOMM es conocido como el protocolo de puerto serie virtual, ya que permite que un dispositivo Bluetooth pueda simular las funciones de un puerto serie.

RFCOMM tiene reservado el valor de protocolo y servicio multiplexor (PSM=0x0003) usado por L2CAP para identificar el tráfico de RFCOMM. El PSM tiene una función similar al número de un puerto en una red IP. PSM es usado por el cliente para conectarse con el servidor.

La especificación menciona la sintaxis utilizada para el protocolo de comunicación en el caso RFCOMM, una dirección Bluetooth válida se describe como:

“btspp://008003DD8901:1”

El protocolo utilizado es descrito en la cadena de caracteres btspp, la dirección del dispositivo está descrita en una cadena HEX que identifican el hardware de ese dispositivo en particular, “:1” es el canal utilizado físicamente por el dispositivo.

El protocolo RFCOMM, que se ubica sobre el protocolo L2CAP, emula una conexión serie RS-232. El perfil de puerto serial (SPP) facilita la comunicación entre dispositivos Bluetooth mediante una interfaz de flujo basado en el protocolo RFCOMM. Algunas capacidades y limitaciones a la nota:

- Dos dispositivos pueden compartir una sola sesión RFCOMM a la vez.
- Hasta 60 conexiones lógicas de serie puede ser multiplexados este período de sesiones.
- Un único dispositivo Bluetooth puede tener como máximo 30 servicios RFCOMM activa.
- Un dispositivo puede admitir sólo una conexión de cliente para un servicio determinado a la vez.

Para que el servidor y el cliente puedan comunicarse con el perfil de puerto serial, cada uno debe realizar unos sencillos pasos:

1. Construir una dirección URL que indica cómo conectar con el servicio, y almacenarlo en la hoja de servicios
2. Hacer la hoja de servicios a disposición del cliente
3. Aceptar una conexión desde el cliente
4. Enviar y recibir datos desde y hacia el cliente

La dirección URL coloca en la hoja de servicios será algo como:

btspp://102030405060740A1B1C1D1E100:5

Esto nos dice que un cliente debe utilizar el perfil Bluetooth Serial Port para establecer una conexión a este servicio, que se identifica con el canal 5 en un servidor de dispositivo cuya dirección es 102030405060740A1B1C1D1E100.

```
String serviceURL  = "btspp://localhost:"+serviceUID.toString());
```

2.4.3 Conexión OBEX

El protocolo Bluetooth indica que la capa del protocolo OBEX se aplica sobre la capa RFCOMM y es útil cuando se desea transferir los datos como un objeto, por ejemplo, archivos. Una de las funciones con que cuenta nuestro sistema es el envío de archivos es por ello que hemos utilizado el protocolo OBEX para la transferencia de éstos. Cabe aclarar que, esto es sólo para el establecimiento de la conexión, los detalles sobre la transferencia de archivos son tratados en el Apéndice B.La sintaxis utilizada para el protocolo OBEX, se indica en la especificación como la dirección Bluetooth descrita: “btgoep://1886AC67D45F:9”.

La sintaxis para el protocolo OBEX es btgoep y 1886AC67D45F:9 representan la dirección física del dispositivo y “:9” el canal utilizado para la conexión en ese dispositivo en específico.

Las conexiones se han hecho de acuerdo con los protocolos mencionados anteriormente. Pero para hacer que las conexiones se lleven a cabo de manera de una forma más eficiente en términos de tiempo de establecimiento de la conexión, hemos de llevar a cabo el emparejamiento entre los dos dispositivos Bluetooth. Llevar a cabo el emparejamiento entre los dispositivos tiene como finalidad, que las conexiones que se realicen a posteriori se hagan de manera automática. Las conexiones Bluetooth normalmente no se llevan a cabo de manera automática, sino que por cuestiones de seguridad, se requiere de autorización para establecer la conexión. La autorización se lleva a cabo estableciendo una clave, misma que ha sido establecida por los administradores de los dispositivos correspondientes. El acuerdo sobre las claves se hace de manera presencial entre los dos administradores, quienes se ponen de acuerdo para establecer una contraseña y una vez hecho esto, se pueden conectar. Hecho esto las conexiones podrán establecerse automáticamente.

2.4.4 Emparejamiento

Cuando dos dispositivos Bluetooth se conectan entre sí se le llama emparejamiento. La estructura de la tecnología Bluetooth implica que cualquier dispositivo compatible con Bluetooth, en casi todo el mundo, se puede conectar a otros dispositivos Bluetooth situados en la proximidad de uno al otro [BluetoothBasics11].

Los dispositivos Bluetooth poseen una base de datos interna (*SDDB)*, en ella encontramos una serie de datos relevantes para establecer conexiones. Entre estos datos están las direcciones

de los dispositivos con los cuales nos hemos conectado anteriormente. Para poder establecer una conexión Bluetooth mencionamos al inicio de este capítulo debe haber un protocolo de comunicación y una dirección hacia dónde conectarse. La dirección física es una característica que viene de fábrica en toda antena Bluetooth Es de tal importancia que se considera como fundamental en la JSR82.

Por ejemplo, si una persona pierde su dispositivo Bluetooth la manera de recuperarlo sería, utilizando ObjectFinder una función de AnonySense, una aplicación que busca la dirección física específica del dispositivo Bluetooth [Shin10].

Para evitar realizar todo el proceso cada vez que intentemos una conexión, existe la base datos contenida en el dispositivo que guarda la información de las conexiones previas.

A continuación enlistamos los pasos a seguir para realizar un emparejamiento:

- Mostrar el dispositivo a los demás, es decir, encender la antena del dispositivo. De esta manera otros dispositivos serán capaces de encontrarnos.
- Realizar una búsqueda de los dispositivos que se encuentran alrededor. Nuestra antena hace un mapeo de acuerdo al alcance que tenga y si encuentra dispositivos alrededor, los enlista para elegir al dispositivo que se conectará.
- Una vez elegido el dispositivo, establecer el protocolo de comunicación, puede ser una transmisión serial con RFCOMM u OBEX para transferencia de archivos y la dirección del dispositivo remoto.
- Hacer una petición de conexión. Otra característica que por cuestiones de seguridad, la especificación establece que un dispositivo Bluetooth es que, cuando uno de ellos intenta conectarse a otro, el dispositivo remoto pregunta si desea conectarse con quien está haciendo la petición y para poder concretar la operación igualmente, de manera segura, establece una contraseña arbitraria que el otro dispositivo deberá enviar como respuesta.
- Una vez puesto de acuerdo y si coinciden las contraseñas, los dispositivos se conectan y podemos decir que se ha realizado el emparejamiento.

CAPÍTULO 3. Aplicación de comunicación Bluetooth

En este capítulo es decrito el sistema, sus entidades y las funciones que utiliza la aplicación que desarrollada y los métodos de Java utilizados. El sistema cumple con el esquema de comunicación cliente-servidor. El servidor es un programa en J2SE que correrá en una computadora que cuente con la plataforma Java. Mientras que la entidad cliente será un programa desarrollado en J2ME para un dispositivo móvil. Más adelante en este capítulo serán descritos detalladamente los componentes de estas dos entidades. En la figura 3.1 es mostrados el esquema de nuestra aplicación.

Para lograr la comunicación con el esquema cliente-servidor fue desarrollada una aplicación en Java tanto para el cliente, como para el servidor. A lo largo de esta sección se explicarán a detalle las características de estos dos módulos (cliente y servidor), fundamentales para la integración del sistema, así como las características que utilizan para comunicarse entre sí.

A continuación se muestra un esquema que ilustra las entidades que conforman el sistema. También la imagen contiene los métodos más importantes en el lenguaje de programación Java utilizados, éstos serán descritos con mayor detalle en el capítulo.

Figura 3.1 Esquema Cliente-Servidor

3.1 Servidor

El servidor es un programa desarrollado en la versión estándar de java J2SE, Su función principal es permitir la comunicación con teléfonos móviles. Un servidor SPP (Serial Port Profile) crea un objeto ***StreamConnectionNotifier*** con la dirección física de cada dispositivo (url).

El objeto StreamConnectionNotifier servirá para poder posteriormente crear la conexión con los dispositivos que deseen acceder a nuestro servidor. El objeto ***StreamConnection*** se crea cuando algún dispositivo se ha conectado. El bloque de código queda de la siguiente manera:

```
notifier = (StreamConnectionNotifier) Connector.open(url);
jTextArea1.append("\n"+"Esperando Conexión...");//Mensaje del servidor, a la espera del
cliente.

        try {
        StreamConnection con = notifier.acceptAndOpen();
        } catch (ServiceRegistrationException sre) {
        System.out.println("Error creando el service record");
        }
        jTextArea5.append("Conectado");
```

Una vez que un dispositivo cliente ha establecido la conexión con nuestro servidor, será posible establecer la comunicación entre estas dos entidades. Las funciones que desarrollamos para nuestro programa son: el envío de texto mediante un chat y la recepción de información específica proveniente de una encuesta. La encuesta se ubica en la aplicación cliente por lo que la describiremos más adelante en este capítulo. En la figura 3.2 observamos la aplicación servidor y sus componentes.

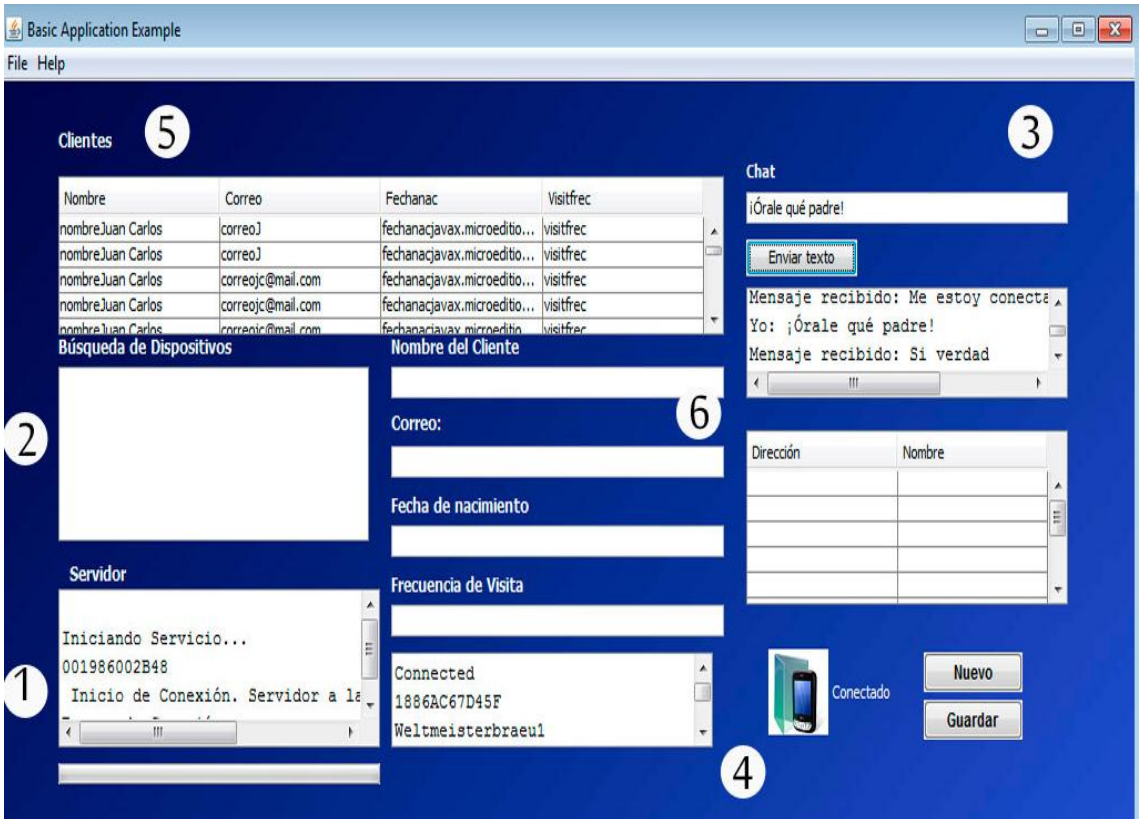


Figura 3.2 Esquema del Servidor, programa en J2SE

Lista de Funciones del servidor J2SE:

1. Establecimiento de los parámetros del servidor para llevar a cabo la comunicación.
2. Búsqueda de los dispositivos conectados alrededor para el envío de archivos. En nuestra aplicación enviamos una imagen .jpg, la imagen puede servir para fines publicitarios. Y enviamos el archivo .jar desarrollado en J2ME, archivo ejecutable de java, al dispositivo móvil para comunicarse, ya sea por medio del chat o la encuesta.
3. Campo para el chat y envío de texto.
4. Campo donde aparece el dispositivo conectado.
5. Tabla donde se muestran los datos recibidos de la encuesta.
6. Campos donde aparecen uno a uno la información de la base de datos.

En el punto número uno es donde establecemos los parámetros para que los clientes puedan acceder al servicio de comunicación. Véase la sección servidor del Apéndice A. El método de Java que hemos utilizado es el llamado ***bluetoothConnection().*** Lo que realiza fundamentalmente este método es hacer que la antena Bluetooth del servidor sea visible a los

dispositivos que estén a su alrededor, establece los parámetros para permitir la conexión con el objeto ***StreamConnectionNotifier*** (línea 13) que contiene la información de nuestra dirección **url** y con el método ***acceptAndOpen();*** (línea 16) se crea otro objeto llamado ***StreamConnection*** y sirve para manipular las conexiones. Finalmente utilizamos los métodos ***refresh(DataInputStream in, DataOutputStream out)*** y ***send(final byte[] data, DataOutputStream ot)*** (línea 39) para el envío y recepción de información (punto 3 en el esquema del servidor). Véase la sección comunicación del Apéndice A.

Para el envío de archivos hacia los dispositivos que se encuentren alrededor, punto 2 del servidor, está el método **run();** del hilo de ejecución (Thread t) de nuestra aplicación. Véase Apéndice A en la sección búsqueda. Este método hace uso de la interface de programación **DiscoveryListener**, véase Apéndice C para conocer su funcionamiento y métodos que integran esta interface de Java. Una vez que haya encontrado el servidor algún dispositivo Bluetooth intentará enviar un archivo con el método ***sendFile(url)*** (líneas 27, 30, 30 y/o 36, pues el envío depende de canales libres para la recepción en los dispositivos remotos) mismo que utiliza la clase ***SendFileTask.*** Véase Apéndice B. La clase ***SendFileTask*** contiene el método **Connection connection = Connector.open(btConnectionURL);** (línea 13) cuando se ejecuta este método el dispositivo Bluetooth accede a la Pila, en este caso la de Windows y es cuando se lleva a cabo el proceso del emparejamiento. Ver capítulo 2. Si se hace la conexión correctamente, se envía el archivo, los dispositivos quedan acoplados y cuando el cliente use su programa la conexión se hará en menor tiempo con el servidor.

Los restantes puntos del esquema del servidor son funciones de una base de datos, éstos explican en la siguiente sección.

3.1.1 Base de datos

Esta parte consiste en recopilar la información que es recibida por el servidor de la aplicación encuesta del cliente. La información es almacenada en una base de datos. La base de datos que utilizamos está controlada por el gestor de base de datos MySQL. La base de datos MySQL se ha convertido en la base de datos de código abierto más popular debido a su alto rendimiento, alta fiabilidad y facilidad de uso. También es la base de datos de elección para una nueva generación de aplicaciones basadas en la pila LAMP (Linux, Apache, MySQL, PHP / Perl / Python). Muchas de las organizaciones más grandes y de más rápido crecimiento del mundo, incluyendo Facebook, Google, Adobe, Alcatel Lucent y Zappos se basan en MySQL para ahorrar tiempo y dinero en sus grandes volúmenes de sitios Web, sistemas críticos de negocio y software empaquetado. MySQL se ejecuta en más de 20 plataformas, incluyendo Linux, Windows, Mac OS, Solaris, AIX de IBM, que le da el tipo de flexibilidad que le da el control. Si eres nuevo en la tecnología de base de datos o un desarrollador con experiencia o DBA, MySQL ofrece una amplia gama de herramientas de base de datos, servicios de apoyo, de capacitación y consultoría que usted tenga éxito [***MySQL***].

Si bien existen otros sistemas gestores de bases de datos ***DBMS*** de software libre, hemos elegido el anterior porque es una aplicación lidera y rápida en bases de datos pequeñas, mientras que, por ejemplo**, *PostgreSQL*, *MaxDB*, *Ingres*** y ***Firebird*** están orientadas hacia bases de datos más grandes, un ejemplo es el ámbito empresarial, mencionamos la relación ***SAP*** con MaxDB [Horstmann05]. Tomando en cuenta nuestro caso particular, nuestra base de datos contiene una sola tabla, consideramos conveniente el uso de **MySQL**. En la Figura 3.3 podemos apreciar que la base de datos consta de una tabla con los datos que son enviados desde el programa encuesta del dispositivo móvil. VéaseApéndice AApéndice A en la sección base de datos de servidor.

Cuando el servidor utiliza el método ***refresh(DataInputStream in, DataOutputStream out)*** y recibe ciertas cadenas de texto especificadas en el código, las cadenas son desplegadas en la pantalla y al recibir la última cadena de la encuesta del programa cliente, se guardan en la base de datos utilizando el método ***save();*** Véase Apéndice B. En la Figura 3.3 podemos observar que la información se ha almacenado correctamente en la base de datos. En el programa servidor podemos saber que se ha almacenado la información, pues el programa da un aviso y además se muestra la información en una tabla, punto 5 del esquema del servidor. En el punto 6 del esquema aparecen los últimos datos recibidos por el servidor.

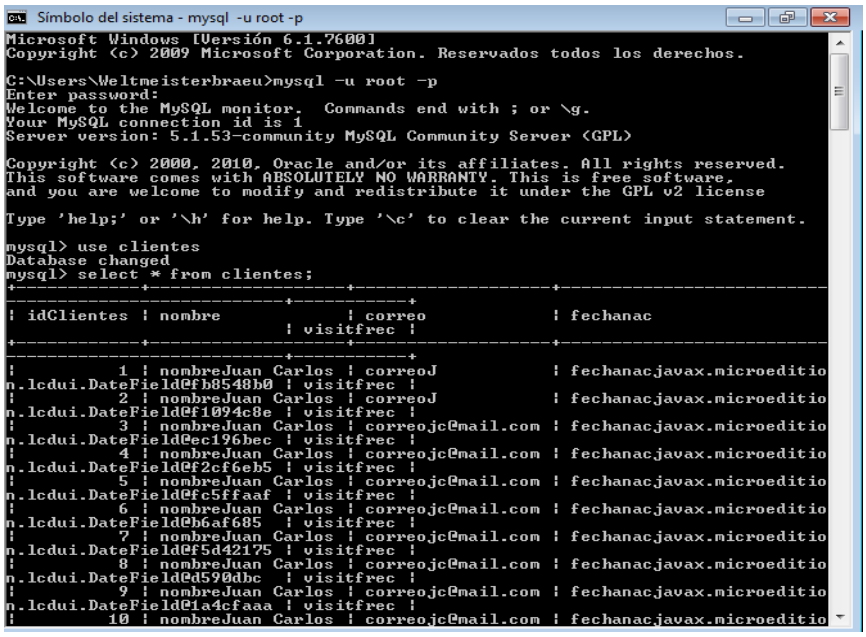


Figura 3.3 Pantalla del sistema gestor de base de datos MySQL

3.2 Cliente

En esta sección son descritos los componentes que integran la aplicación cliente. Esta aplicación está desarrollada en la plataforma Java Microedition J2ME para dispositivos móviles. Aquellos dispositivos que cuenten con la plataforma Java podrán utilizar este programa. El programa es un archivo .jar, archivo ejecutable para la plataforma Java y es capaz de conectarse e intercambiar información (cadenas de texto) con el servidor. La figura 3.4 muestra el esquema de la aplicación cliente en el ***IDE*** Netbeans:

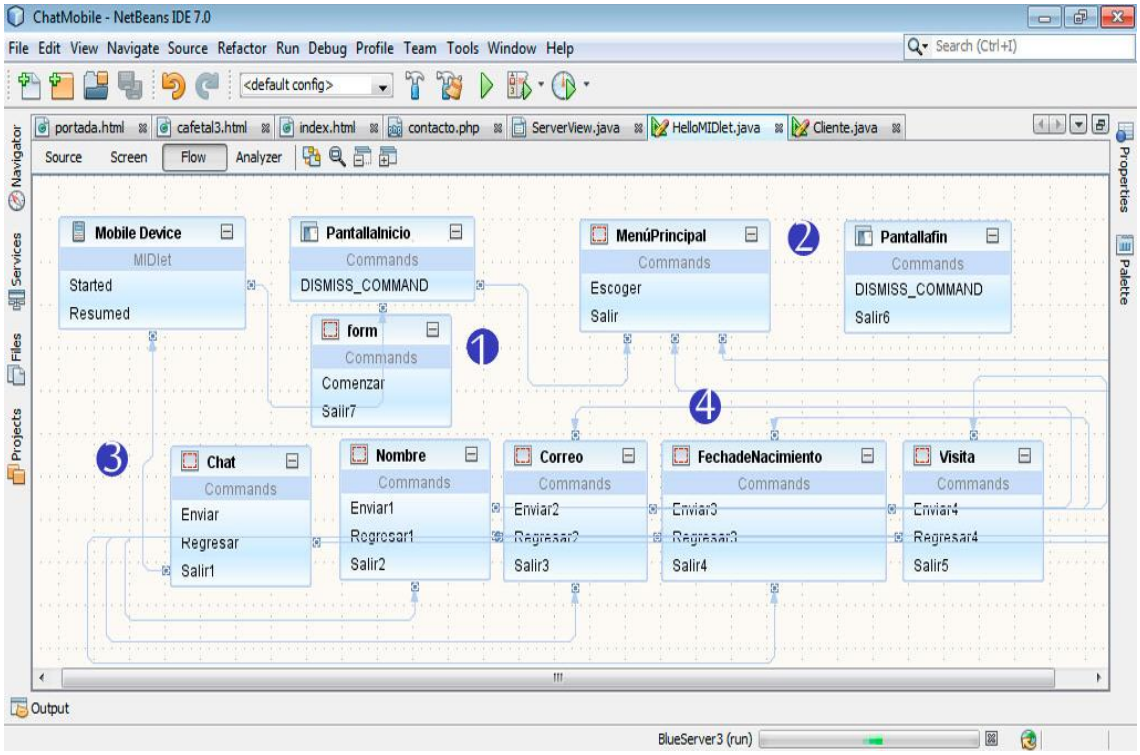


Figura 3.4 Esquema de funciones del cliente, programa en J2ME

Lista de Funciones del cliente J2ME:

1. Conexión con el servidor (método connect();)
2. Encontrado el servidor y realizada la conexión, se despliega en el celular una pantalla donde se deberá elegir la manera de comunicarse (Chat o envío información por la encuesta).
3. Pantalla de Chat, intercambio de texto.
4. La aplicación pide al usuario llenar campos de información y los envía al servidor, quien los recopilará y almacenará en una base de datos.

Cuando se ejecuta el programa .jar en el dispositivo móvil automáticamente intentará conectarse con el servidor con el método ***connect()*** (Véase Apéndice A sección conexión, punto 1)***.*** A diferencia con el servidor, en este caso se busca los parámetros específicos del servidor su ***url.*** Mientras que el servidor sólo busca dispositivos que se en encuentran a su alrededor. Cuando se ha podido establecer la conexión (punto 2), el programa despliega un menú de opciones de comunicación. Estas son, comunicarse por medio de un chat o enviar información al servidor a través de una encuesta (puntos 3 y 4). La comunicación se realiza de igual manera en ambos casos. Para el envío y recepción de información usamos los métodos: ***public void refresh(DataInputStream in, DataOutputStream out)*** y ***synchronized void send(final byte[] data).*** Véase Apéndice AApéndice A en la sección de comunicación.

CAPÍTULO 4. Pruebas de conexión Bluetooth

Este capítulo trata sobre las pruebas realizadas para el análisis de conexiones vía Bluetooth. El análisis de la conexión se hizo considerando algunas variables características de un área personal de trabajo son, la distancia, si es un lugar abierto ó cerrado y el número de dispositivos que se encuentran en ella. Este análisis permitió obtener una serie de datos que indican las condiciones donde es conveniente llevar a cabo una conexión con Bluetooth. Lo anterior, se hace con el fin de adaptar dichas condiciones a las diversas aplicaciones que se podrían desarrollar haciendo uso de java y una conexión Bluetooth. En el Apéndice C de este documento describimos la forma de calcular los tiempos de conexión con la aplicación Java, el ***Valor Medio*** y la desviación estándar (***Dstd***) en las mediciones.

Bluetooth permite establecer la conexión (véase capítulo 2) de dos diferentes maneras. Denominados modos de establecimiento de conexión. El primero*, modo de conexión sin emparejamiento* y el otro *modo de conexión con emparejamiento*.

El modo de conexión sin emparejamiento funciona haciendo una búsqueda de dispositivos Bluetooth, éstos son reconocidos junto con sus direcciones y después se hace la petición de conexión. En la petición debe haber un acuerdo directo entre los dos usuarios del dispositivo Bluetooth con una clave previamente establecida entre ellos. En algunas ocasiones, como es el caso de la pila Bluetooth de Windows, la contraseña se establece de manera aleatoria y automática. Pero en los teléfonos móviles normalmente el administrador del dispositivo se encarga de establecer la contraseña. En la petición de conexión, la contraseña establecida se escribe en el otro dispositivo, entonces se realiza un tipo de *Handshaking*. Los dispositivos entonces, quedan acoplados y por tanto, conectados. Esto permite almacenar la dirección del dispositivo remoto en la pila del dispositivo Bluetooth y cuando se quiera establecer una nueva conexión, éste la aceptará sin restricción alguna, pues es un dispositivo confiable, por tanto el tiempo de establecimiento de conexión disminuye, además se automatiza el proceso de conexión, las conexiones que se lleven a cabo de esta manera se les llama con emparejamiento.

El objetivo de las pruebas es hacer una comparación de los tiempos en que se realiza la conexión, fueron tomadas alrededor de 10 mediciones del tiempo por cada variable. Las variables sirven para conocer las características de la conexión, es decir, con o sin emparejamiento, distancias, número de dispositivos y si el lugar donde se hacen las conexiones está al aire libre o es un lugar cerrado. Fue calculado el promedio para tener un valor representativo por cada variable y el valor de la desviación estándar para conocer el error de las mediciones y darle certidumbre a los valores. La manera en que hicimos las mediciones están descritas de manera más detallada en el Apéndice C.

.

4.1 Pruebas de Conexión

En esta sección son mostrados los resultados de las pruebas realizadas. Éstas permitieron conocer y determinar ciertas características adaptables a un entorno real de conexión, así como sus variables y condiciones. Con los resultados es posible diseñar y adaptar las aplicaciones de software de acuerdo a las características que arrojen los resultados. Con las pruebas logramos recopilar información para el diseño, adaptación y mejora de aplicaciones que usen Bluetooth.

La estructura de comunicación Bluetooth es cliente-servidor. El servidor, programa desarrollado en J2SE montado en PC con sistema operativo Windows 7, se encarga de controlar las operaciones y procesos de comunicación. El cliente es un programa J2ME que se conecta al servidor. Los teléfonos móviles utilizados están descritos en el Apéndice D, en éste se indican las características, similitudes y diferencias de dichos dispositivos.

1.	**Nokia 2330**
2.	**Alcatel ice3**
3	**Nokia 5310**
4	**Samsung gt-m2310**

Tabla 4.1 Teléfonos móviles utilizados para las pruebas de conexión

4.2 Modos de Conexión

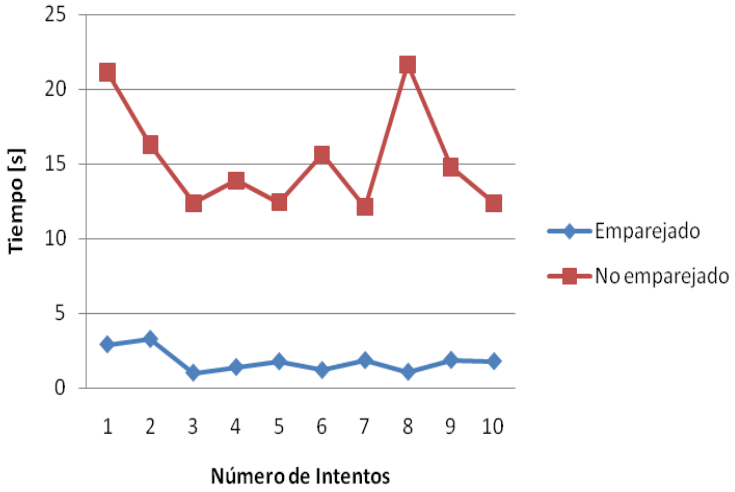
Para el estudio se han considerado 2 modos de conexión para dispositivos Bluetooth*, modo de conexión sin emparejamiento* y *modo de conexión con emparejamiento.* El proceso para llevar a cabo el emparejamiento se describe en capítulo 4. En primer lugar, se hicieron las mediciones con una conexión sin emparejamiento y determinamos el tiempo en el cual se establece la conexión. Posteriormente fueron hechas las pruebas con emparejamiento y también determinamos los tiempos de conexión. A partir de estos resultados serán consideradas otras variables para determinar otras particularidades que existen en la conexión Bluetooth.

4.2.1Pruebas de los modos de conexión. (Dispositivo 1, Nokia 2330)

Los primeros resultados obtenidos, fueron haciendo la comparación entre los modos de conexión. Las pruebas se hicieron solamente con el dispositivo (1). Fue medido el tiempo de conexión con el dispositivo no emparejado. Después se midió el tiempo con el mismo dispositivo ya emparejado. La distancia en que fueron realizadas las mediciones fue de 1m en un entorno cerrado. Se le llama entorno al espacio donde trabajamos. Es entorno cerrado

cuando nos encontramos en un lugar con paredes y techo, un entorno abierto es aquel que esta al aire libre.

A continuación se muestran en la Gráfica 4.1 Comparación de modos de conexión los tiempos de conexión con el dispositivo (1) no emparejado y con el emparejamiento previamente realizado:



Gráfica 4.1 Comparación de modos de conexión

La disminución del tiempo de conexión es notable. El tiempo promedio de conexión en dispositivo no emparejado es 15.25s y en el dispositivo ya emparejado es 1.8s. El cálculo del error en conexiones sin emparejamiento es ± 3.36s, mientras que con emparejamiento es ± 0.71s. Vemos que también disminuye el error.

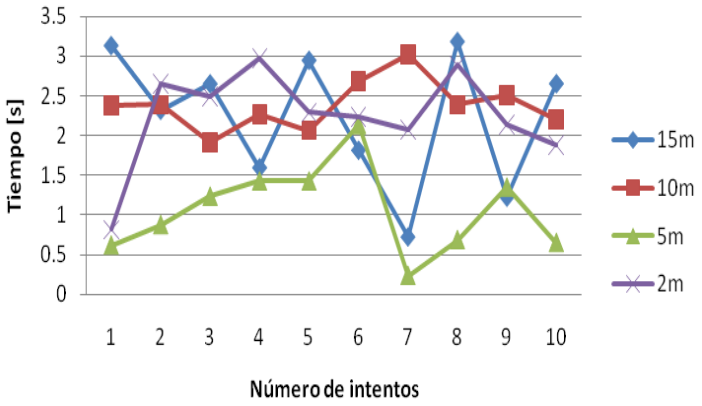
Lo anterior significa que la conexión con emparejamiento reduce el tiempo de conexión a 11.8% del tiempo de conexión sin emparejamiento. En el caso del error, las conexiones con emparejamiento representan un 21.13% del error que hubo en las conexiones sin emparejamiento. Este resultado indica que será conveniente hacer el emparejamiento. Es por ello que, de ahora adelante las pruebas se realizaron con dispositivos emparejados.

4.3 Pruebas de distancia

Ahora que se ha determinado que es conveniente hacer el emparejamiento siempre que se quiera establecer una conexión con un dispositivo reconocido, el estudio se enfocará en este tipo de conexiones con el fin de determinar ciertas características que permitan llevar a cabo conexiones más eficientes y rápidas. Para poder determinar estas características establecimos como variables que influyen en las conexiones, la distancia, número de dispositivos que se estén conectando y entorno (abierto o cerrado).

4.3.1 Prueba de distancia con dispositivo (Dispositivo 1, Nokia 2330) en un entorno cerrado

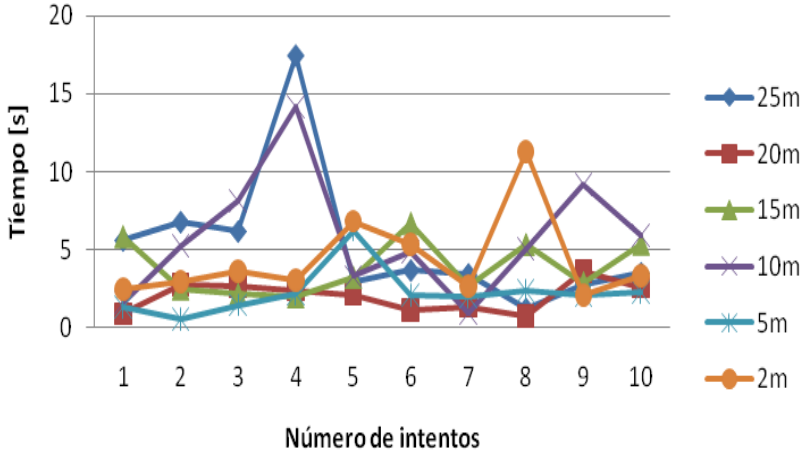
La primera prueba se hizo considerando distancia y tiempo de conexión con el dispositivo 1, previamente emparejado. Los resultados se muestran en la Gráfica 4.2:



Gráfica 4.2 Conexión con dispositivo (1) emparejado en entorno cerrado.

4.3.2 Prueba distancia con dispositivo (Dispositivo 1, Nokia 2330) en un entorno abierto

Ahora fue repetido el experimento anterior pero ahora en un entorno abierto. En la Gráfica 4.3 es posible observar los tiempos de conexión.



Gráfica 4.3 Conexión con dispositivo (1) emparejado en entorno abierto.

Las conexiones con el dispositivo (1) en un entorno *cerrado* se hicieron considerando distancias de 2, 5, 10 y 15m. El resultado, los tiempos promedio están en el rango de 1.06 a 2.8s.

Las conexiones hechas con emparejamiento con el dispositivo (1) en un entorno *abierto*, resultaron que los tiempos promedio están en el rango de 3.1-4.37s. Se pueden apreciar los

resultados en la Gráfica 4.2 Conexión con dispositivo (1) emparejado en entorno cerrado.y en la Gráfica 4.3. En la Tabla 4.2 Comparativo de tiempo de conexión con dispositivo (1) ya emparejado en entorno abierto y entorno cerrado.se puede apreciar que las conexiones son más estables en un entorno cerrado, pues el error en las conexiones en un entorno cerrado está en el rango ±0.31-±0.84s. Mientras que en un entorno abierto está en el rango ±0.95-±2.8s

Distancia [m]	2[m]	5[m]	**10**[m]	**15**[m]
Tiempo[s] Entorno cerrado	2.24 ± 0.61	1.06 ± 0.55	2.38 ± 0.31	2.22 ± 0.84
Tiempo[s] Entorno abierto	4.37 ± 2.8	3.04 ± 1.21	3.1 ± 0.95	3.17 ± 1.65

Tabla 4.2 Comparativo de tiempo de conexión con dispositivo (1) ya emparejado en entorno abierto y entorno cerrado.

Haciendo una comparación con los resultados anteriores se aprecia que el tiempo de conexión aumentó aproximadamente al doble en un lugar abierto, de ahí que influye el entorno en donde se lleve a cabo la conexión. También los resultados muestran que el tiempo no aumenta conforme la distancia lo hace, por ejemplo, en la primera prueba los tiempos promedio en 2m y 15m son iguales o en la prueba 2 en 25m y 10m también son iguales y así, hay otros casos donde se muestra que no hay una relación proporcional entre tiempo y distancia. Otra particularidad es que a 5m están los tiempos más bajos de las mediciones.

4.3.3 Prueba distancia con 4 dispositivos en un entorno abierto

Las pruebas anteriores (Ver Tabla 4.2 Comparativo de tiempo de conexión con dispositivo (1) ya emparejado en entorno abierto y entorno cerrado.) muestran que las conexiones en un entorno se realizan en un tiempo menor y con mayor eficacia (menor error), es decir, conviene hacer conexiones en este tipo de entorno. Pero ahora fue requerido conocer si también es conveniente hacerlo en un entono abierto. Para ello, agregamos una variable más a las pruebas. Esta es que más dispositivos se estén conectando al servidor, cuatro fue el número de dispositivos que conectamos al servidor y las conexiones se realizaron en el orden que aparece en la Tabla 4.1 Teléfonos móviles utilizados para las pruebas de conexióny sus características son descritas en el Apéndice D.-

Las pruebas realizadas consisten en conectar los 4 dispositivos móviles secuencialmente con el servidor, en el orden que aparece en la Tabla 4.1 Teléfonos móviles utilizados para las pruebas de conexión4.1. El objetivo es saber qué tanto afecta al tiempo de conexión y si la conexión se mantiene en tanto entra una nueva conexión en un entorno con mayores dificultades para

realizar las conexiones. En la Tabla 4.3 se muestran los resultados de conexiones realizadas con 4 dispositivos emparejados a diferentes distancias en un entorno abierto.

Dispositivo	5[m]	10[m]	15[m]	20[m]
	Tiempo [s]	Tiempo [s]	Tiempo [s]	Tiempo [s]
1	3.04 ± 1.21	3.1 ± 0.95	3.17 ± 1.65	3.77 ± 1.76
2	3.17 ± 1.22	3.49 ± 2.34	2.87 ± 0.89	4.7 ± 3.23
3	2.55 ± 1.22	3.87 ± 1.27	5.18 ± 5.3	7.65 ± 4.38
4	2.7 ± 1.63	3.99 ± 5.35	4.56 ± 2.3	4.38 ± 1.39

Tabla 4.1 Resultados de prueba de conexión a 5, 10, 15 y 20 m en un entorno abierto

En estas pruebas se tomaron en cuenta las condiciones de entorno *abierto y mayor número de dispositivos (4)* ver Tabla 4.1 Resultados de prueba de conexión a 5, 10, 15 y 20 m en un entorno abierto. Medimos en 5m, 10m, 15m y 20m, los tiempos promedio fueron 2.8s, 3.6s, 3.9s y 5.1s y el error promedio ±1.3s, ±2.4s, ±2.5s y ±2.6s respectivamente y corresponden en ese orden a las distancias de 5m, 10m, 15m y 20m. En este caso influye la distancia pues se incrementa tanto el tiempo promedio de conexión, como el error.

Conclusiones

Fue desarrollado un sistema de comunicación (cliente-servidor) Bluetooth en Java. El servidor es capaz de hacer búsqueda de dispositivos y enviarles archivos (en nuestro caso imagen y el archivo J2ME para el cliente), intercambiar información con el dispositivo móvil y guardar la información recibida en una base de datos. Mientras que el cliente es capaz de conectarse de manera automática debido al uso del emparejamiento en las conexiones e intercambiar texto con el servidor.

Cuando se establezcan con frecuencia conexiones es conveniente utilizar el emparejamiento. La conexión con emparejamiento reduce el tiempo de conexión a 11.8% del tiempo de conexión sin emparejamiento.

Es el caso de nuestro sistema cliente-servidor. El tiempo de conexión resulta más rápido en entornos cerrados que en abiertos. Con el dispositivo 1 se tiene que en un entorno cerrado los tiempos promedio están en el rango de 1.06-2.8s. Mientras que en un entorno abierto los tiempos promedio están en el rango de 3.1-4.37s. De acuerdo a lo anterior es conveniente conectarse en un entorno cerrado.

En ocasiones las pruebas de conexión mostraron casos donde el valor la desviación estándar se elevaba. Esto significa que en ocasiones la conexión tenía dificultades para llevarse a cabo. Esto puede ser por dos razones, una de software, es decir, la manera en que el sistema operativo ejecuta el programa. O bien, por hardware que la señal sufra de interferencias. Se sugiere un análisis del hardware y como se ha mencionado este trabajo, en un principio, pretendía adaptarse a un desarrollo de hardware, independiente de este estudio.

Por último, al momento en que comenzó este proyecto, la tecnología Java tenía presencia en muchos dispositivos móviles disponibles en el mercado. De un año ó dos al momento, los teléfonos con sistemas Android y iOS han aumenado su presencia en el mercado. Convendría hacer una adapatación referente a los lenguajes de programación de dichos sistemas para abarcar un mayor rango de dispositivos móviles y de esa manera, poder desarrollar aplicaciones que cumplan con los conceptos aquí tratados sobre Bluetooth y dispositivos móviles.

Apéndice A. Aplicación Java

Búsqueda

```
//Método Principal
1. Thread t = new Thread(this)
2. public void run() {
3. for(int a=1; a<100; a++){ //Loop para que la búsqueda se ejecute constantemente
4. try {
5. LocalDevice localDevice = LocalDevice.getLocalDevice();//Inicio dispositivo
6. DiscoveryAgent agent = localDevice.getDiscoveryAgent();//agente de búsqueda
7. jTextArea3.append("\nIniciando Búsqueda...");//Mensaje de confirmación de inicio del Bluetooth
8. agent.startInquiry(DiscoveryAgent.GIAC, this);//Iniciar la búsqueda de dispositivos
9. try {
10. synchronized (lock) {
11. lock.wait();
12. }
13. } catch (InterruptedException e) {
14. }
15. //los dispositivos encontrados los transformo en un valor entero, para poder manipularlo
16. int deviceCount = vecDevices.size();
17. if (deviceCount <= 0) {
18. } else {
19. jTextArea3.append("\n Dispositivos: \n ");
20. for (int i = 0; i < deviceCount; i++) {//Búsqueda de todos los dispositivos alrededor
21. RemoteDevice remoteDevice = (RemoteDevice) vecDevices.elementAt(i);
22. jTextArea3.append("\n " + remoteDevice.getFriendlyName(true) + " " + remoteDevice.getBluetoothAddress());
23. TableModel tm = jTable2.getModel();
24. jTable2.setValueAt(remoteDevice.getBluetoothAddress(), i, 0);
```

```
25. jTable2.setValueAt(remoteDevice.getFriendlyName(true), i, 1);
26.  String url =    "btgoep://" + remoteDevice.getBluetoothAddress() +":1" ;
27.  String url2 =    "btgoep://" + remoteDevice.getBluetoothAddress() +":3" ;
28.  String url3 =    "btgoep://" + remoteDevice.getBluetoothAddress() +":6" ;
29.  String url4 =    "btgoep://" + remoteDevice.getBluetoothAddress() +":9" ;
30. if(jTable2.getValueAt(i, 0)!= remoteDevice.getBluetoothAddress()) { //Si ya se ha
    conectado, esta condición evita que lo haga constantemente
31.   try{
32.   sendFile(url);
33.   }catch(Exception e){System.out.println("Error en envío: "+ e);}
34.  try{
35. sendFile(url2);
36. }catch(Exception e){System.out.println("Error en envío: "+ e);}
37. try{
38. sendFile(url3);
39. }catch(Exception e){System.out.println("Error en envío: "+ e);}
40. try{
41. sendFile(url4);
42. }catch(Exception e){System.out.println("Error en envío: "+ e);}
43. } }
44. }
45. jTextArea3.append("\n Service Search Completed. \n");
46. jTextArea3.append("\n Ya acabé \n");
47. } }catch (Exception e1) { } } }
```

Conexión

```
//Método para establecer conexión con el Servidor----------------------------------------
1.  void connect() {
2.  t4 = new Thread() {
```

```
3. public void run() {
4. try {
5. //Establecimiento de Conexión con el Servidor
6. conn = (StreamConnection) Connector.open("btspp://001986002B48:1");//Dirección
   física del servidor
7. out = conn.openDataOutputStream();//to open output stream
8. in = conn.openDataInputStream();
9. int c;
10. for (c = 0; c < 30; c++) {
11. refresh(in, out);
12. }
13. //termino conexión
14. conn.close();
15. } catch (IOException ioe3) { }
16. } else {
17. form.append("Servidor no encontrado");
18. } }
19. };
20. t4.start();}
```

Emparejamiento

Este es el método que utilizamos para hacer las mediciones del tiempo de conexión con emparejamiento. Cuando se intenta establecer la conexión (línea 10) se ejecuta el programa controlador de la pila Bluetooth, en este caso Windows, y es cuando se lleva a cabo el proceso de emparejamiento.

```
1. synchronized void emparejamiento() {
2. Thread t3 = new Thread() {
```

```
3.  public void run() {
4.  long tiempoInicio = System.currentTimeMillis();
5.  System.out.println("inicio");
6.  try {
7.  LocalDevice localDevice = LocalDevice.getLocalDevice();//Inicio dispositivo
8.  DiscoveryAgent agent = localDevice.getDiscoveryAgent();//agente de búsqueda
9.  text.append("\nIniciando conexión...");
10. conn = (StreamConnection) Connector.open(address);
11. long totalTiempo = System.currentTimeMillis() - tiempoInicio;
12. System.out.println("El tiempo de demora es :" + totalTiempo + " miliseg");
13. text.append("\n El tiempo de demora es :" + totalTiempo + " miliseg");
14. text.append("Connected");
15. } catch (IOException ioe3) {
16.  text.append("\nNot connected");
17. } }
18. };
19.  t3.start();  }
```

Servidor

Establecimiento de Servicios:

```
UUID uuid = new UUID("100001", true);
String url = "btspp://localhost:" + uuid + ";name=bitiserial" + ";master=true";
```

```
1.  void bluetoothConnection() {
2.  Thread tbt = new Thread() {
3.  public void run() {
4.  try{
```

```
5.  localdevice = LocalDevice.getLocalDevice();//Acceder al Hardware
6.  String dir = localdevice.getBluetoothAddress();//y obtener su dirección
7.   jTextArea1.append("\nIniciando Servicio...");//Mensaje de confirmación de inicio del
    Bluetooth
8.  localdevice.setDiscoverable(DiscoveryAgent.GIAC);//  Permite que sea visible a los demás
    dispositivos Bluetooth
9.  System.out.println(localdevice.getBluetoothAddress());
10.  jTextArea1.append("\n"+localdevice.getBluetoothAddress());
11.  jTextArea1.append("\n Inicio de Conexión. Servidor a la Espera de
    conexiones");//Mensaje de inicio de la conexión
12.   /*Un servidor SPP crea un objeto StreamConnectionNotifier del siguiente modo:
    *Usando el apropiado string ("url") para un servidor SPP como argumento de
    *Connector.open()  *Haciendo un  casting del resultado de  Connector.open() al interfaz
    StreamConnectionNotifier.  */
13. notifier = (StreamConnectionNotifier) Connector.open(url);//Parámetros de conexión,
    creo objeto conexión
14. jTextArea1.append("\n"+"Esperando Conexión...");//Mensaje del servidor, a la espera del
    cliente.
15. try {
16. con = notifier.acceptAndOpen();
17. } catch (ServiceRegistrationException sre) {
18. System.out.println("Error creando el service record");
19. }
20. jTextArea5.append("Connected");
21. jLabel6.setVisible(true);
22. RemoteDevice dev = null;
23. String name, address;
24. boolean authorized = false;
```

```
25. dev = RemoteDevice.getRemoteDevice(con);
26. address = dev.getBluetoothAddress();
27. try{
28. name = dev.getFriendlyName(false);
29. //authorized = dev.isAuthorized(con);
30. } catch (IOException e) {
31. name = "Unknown";}
32. jProgressBar1.setIndeterminate(false);
33. jTextArea5.append("\n"+address);
34. jTextArea5.append("\n"+name);
35. in = con.openDataInputStream();
36. out = con.openDataOutputStream();
37. int d;
38. for (d = 0; d < 100; d++) {
39. refresh(in, out);
40. }
41. String tres = "Conectado";
42. byte data[] = tres.getBytes();
43. send(data, out);
44. synchronized (lock)//Solamente un subproceso puede acceder a dicho método a la vez.
    Cuando un subproceso intenta acceder al método sincronizado mirará a ver si la llave
    está echada, en cuyo caso no podrá accederlo. Si método no tiene puesta la llave
    entonces el subproceso puede acceder a dicho código sincronizado.
45. {
46. try {
```

47. lock.wait();// Espera a que otro hilo de Ejecución (Thread) interrumpa la operación, es decir, llame al objeto
48. } catch (InterruptedException oe) {//Estructura Catch para capta una interrupción al proceso iniciado por "try"
49. oe.printStackTrace(); //método asociado a la interrupción; mensaje de error
50. } }
51. } catch (IOException e1) {
52. } }
53. };
54. tbt.start(); }

Base de datos

Código SQL que representa la tabla que contiene los datos que recibe el servidor de la aplicación cliente.

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO'
*/;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

CREATE DATABASE /*!32312 IF NOT EXISTS*/ `clientes` /*!40100 DEFAULT CHARACTER SET
latin1 */;

USE `clientes`;

DROP TABLE IF EXISTS `clientes`;

/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
```

```
CREATE TABLE `clientes` (

`idClientes` int(10) unsigned NOT NULL AUTO_INCREMENT,
`nombre` varchar(255) NOT NULL,
`correo` varchar(255) NOT NULL,
`fechanac` varchar(255) NOT NULL,
`visitfrec` varchar(255) NOT NULL,

  PRIMARY KEY (`idAntecedentes`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*!40101 SET character_set_client = @saved_cs_client */;

LOCK TABLES `clientes` WRITE;

/*!40000 ALTER TABLE `clientes` DISABLE KEYS */;
/*!40000 ALTER TABLE `clientes` ENABLE KEYS */;

UNLOCK TABLES;

/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

Métodos para manejar la base de datos

```
1. private void save(){
2. if(jComboBox1.getSelectedItem().toString().isEmpty() ||
   jComboBox2.getSelectedItem().toString().isEmpty() || jTextField1.getText().isEmpty() ||
   jTextArea1.getText().isEmpty()){
3. JOptionPane.showMessageDialog(null, "Hay campos nulos \n Verifique nuevamente");
4. }else{
5. try {
6. entityManager1.getTransaction().commit();
```

```
7.  entityManager1.getTransaction().begin();
8.  JOptionPane.showMessageDialog(null, "Datos Almacenados");
9.  } catch (RollbackException rex) {
10. rex.printStackTrace();
11. entityManager1.getTransaction().begin();
12. List<blueserver3.Clientes> merged = new ArrayList<blueserver3.Clientes>(list1.size());
13. for (blueserver3.Clientes c : list1) {
14. merged.add(entityManager1.merge(c));
15. }
16. list1.clear();
17. list1.addAll(merged);
18. } } }
```

```
1.  private void nuevo(){
2.  blueserver3.Clientes c = new blueserver3.Clientes();
3.  entityManager1.persist(c);
4.  list1.add(c);
5.  int row = list1.size()-1;
6.  jTable1.setRowSelectionInterval(row, row);
7.  jTable1.scrollRectToVisible(jTable1.getCellRect(row, 0, true));   }
```

Interface DiscoveryListener

// 4 Métodos siguientes son los que genera DiscoveryListener, necesarios para hacer la búsqueda de dispositivos y servicios

```
1.  public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) {
2.   if (!vecDevices.contains(btDevice)) {
3.  vecDevices.addElement(btDevice);
```

```
4.  }  }
5.  public void inquiryCompleted(int discType) {
6.  String inqStatus = null;
7.  if (discType == DiscoveryListener.INQUIRY_COMPLETED) {
8.  inqStatus = "[client:] Inquiry completed\n";
9.  } else if (discType == DiscoveryListener.INQUIRY_TERMINATED) {
10. inqStatus = "[client:] Inquiry terminated\n";
11. } else if (discType == DiscoveryListener.INQUIRY_ERROR) {
12. inqStatus = "[client:] Inquiry error\n";
13. }
14. synchronized (lock) {
15. lock.notify();
16. } }
17. public void servicesDiscovered(int transID, ServiceRecord[] servRecord) {
18. for (int i = 0; i < servRecord.length; i++) {
19. //vecServices=servRecord[i].getConnectionURL(0,false);

20.    DataElement serviceNameElement = servRecord[i].getAttributeValue(0x0100);
21. String _serviceName = (String) serviceNameElement.getValue();
22. String serviceName = _serviceName.trim();
23. if (serviceName.equals("bitiserial")) {
24. try {
25. connectionURL = servRecord[i].getConnectionURL(0, false);
26. active = true;
27. } catch (Exception e) {
```

```
28. } } } }
29. public void serviceSearchCompleted(int transID, int respCode) {
30. String searchStatus = null;
31. if (respCode == DiscoveryListener.SERVICE_SEARCH_DEVICE_NOT_REACHABLE) {
32. searchStatus = "Device not reachable\n";
33. } else if (respCode == DiscoveryListener.SERVICE_SEARCH_NO_RECORDS) {
34. searchStatus = "Service not available\n";
35. } else if (respCode == DiscoveryListener.SERVICE_SEARCH_COMPLETED) {
36. searchStatus = "Service search completed\n";
37. } else if (respCode == DiscoveryListener.SERVICE_SEARCH_TERMINATED) {
38. searchStatus = "Service search terminated\n";
39. } else if (respCode == DiscoveryListener.SERVICE_SEARCH_ERROR) {
40. searchStatus = "Service search error\n";
41. }
42. synchronized (lock) {
43. lock.notify();
44. }}
```

Comunicación

Recepción de información

```
1. public void refresh(DataInputStream in, DataOutputStream out){//Recepción de
   Mensajes
2. try {
3. Integer in3 = in.readInt();//Recibo la longitud de Cadena de entrada
4. String int4 = "";//El ciclo siguiente sirve para concatenar cada símbolo y así formar el
   mensaje que quiero recibir
```

```
5.  for (int i = 0; i < in3; i++) {
6.  int4 += (char) in.read();//función para concatenar e integrar el mensaje    }
7.  /*El siguiente bloque de código significa recibir una cadena específica si
     * es esta igual a la que esperamos recibir, envío un mensaje definido en
     * una cadena específica.   */
8.  jTextArea2.append("\nMensaje recibido: "+int4);
9.  if (int4.equals("Drei 3")) {
10. String tres = "Tres 3";
11. byte data[] = tres.getBytes();
12. send(data, out);
13. } else if (int4.equals("001")) {
14. jTextArea1.setText(int4 +"Respuesta Correcta");
15. String uno = "Respuesta Correcta";
16. byte data[] = uno.getBytes();
17. send(data, out);
18. save();
19. } else if (int4.contains("nombre")) {
20. nuevo();
21. jTextField1.setText(int4);
22. } else if (int4.contains("correo")) {
23. jTextField2.setText(int4);
24. } else if (int4.contains("fechanac")) {
25. jTextField3.setText(int4);
26. } else if (int4.contains("Frecuente ")||int4.contains("Casual ") ) {
27. jTextField4.setText(int4);
```

28. save();}
29. } catch (Exception e)//Excepción al proceso ejecutado { }

Envío de información

1. synchronized void send(final byte[] data, DataOutputStream ot){//Función para enviar una trama de información
2. t4 = new Thread(){//Inicio de un nuevo hilo de ejecución, proceso independiente a ejecutar
3. public void run(){//función para el inicio del hilo de ejecución
4. try {
5. System.out.println(out);//Con esta instrucción verifico que el objeto }Connection no sea nulo y contenga información, lo que hay en DataOutputStream
6. out.writeInt(data.length);//Asigno un valor entero, longitud de la cadena
7. out.flush();//libero el buffer del stream de salida
8. out.write(data);//Escribo la trama, "byte[]" en el OutputStream
9. out.flush();//libero el buffer del stream de salida
10. synchronized (lock) {
11. lock.notify();//subproceso a la espera }
12. } catch (IOException ioe2)//Interrupción de entrada o salida, si no se cumple la condición del try inicial{ } }
13. };
14. t4.start(); }

Apéndice B. Transferencia de archivos con el protocolo OBEX

Envío de archivos

```
1.  void sendFile(final String url) {
2.  Thread tbt = new Thread() {
3.  public void run() {
4.  jTextArea1.append("\n Enviando archivo");
5.  try{
6.  // Obtenemos los bytes del archivo para enviarlo de esa manera
7.  FileInputStream stream = new FileInputStream("logo2.jpg");
8.  File f = new File("logo2.jpg");
9.  int size = (int) f.length();
10. byte[] file = new byte[size];
11. stream.read(file);
12. String filename = f.getName();
13. SendFileTask task =  new SendFileTask(url, file, filename);
14. Thread thread = new Thread(task);
15. task.run();
16. } catch (Exception e){ }
17. try{
18. FileInputStream stream = new FileInputStream("Clientebt.jar");
19. File f = new File("Clientebt.jar");
20. int size = (int) f.length();
21. byte[] file = new byte[size];
22. stream.read(file);
```

```
23. String filename = f.getName();
24. SendFileTask task =  new SendFileTask(url, file, filename);
25. Thread thread = new Thread(task);
26. task.run(); }
27.  catch (Exception e){  System.out.println(e);  }  }
28. };
29. tbt.start(); }
```

Clase SendFileTask

Es la clase de Java utilizada para hacer la conexión con el dispositivo remoto y poder enviar el archivo a ese dispositivo.

```
1. public class SendFileTask implements Runnable {
2. private String btConnectionURL;
3. private byte[] file;
4. private String filename;
5. public SendFileTask(
6. String url, byte[] file, String filename) {
7. this.btConnectionURL = url;
8. this.file = file;
9. this.filename = filename;
10. }
11. public void run() {
12. try {
13. Connection connection =   Connector.open( btConnectionURL );
14. ClientSession cs =  (ClientSession) connection;
15. HeaderSet hs = cs.createHeaderSet();
```

```
16. cs.connect(hs);
17. hs.setHeader(HeaderSet.NAME, filename);
18. hs.setHeader(HeaderSet.TYPE, "image/jpeg");
19. hs.setHeader(
20. HeaderSet.LENGTH,
21. new Long(file.length));
22. Operation putOperation = cs.put(hs);
23. OutputStream outputStream = putOperation.openOutputStream();
24. outputStream.write(file);
25. outputStream.close();
26. outputStream.close();
27. cs.disconnect(null);
28. connection.close();
29. } catch (Exception e) {
30. e.printStackTrace();
31. } } }
```

Apéndice C. Método de Java para medición de tiempos y cálculo del valor medio y la desviación estándar.

Método de Java para establecer la conexión y medir el tiempo que tarda en hacerlo.

Para la medición de los tiempos de conexión, desarrollamos un método en Java para llevarlo a cabo.

A continuación mostramos el código fuente documentado:

```
1. private void BotónConexión (java.awt.event.ActionEvent evt) {
2. long tiempoInicio = System.currentTimeMillis();// Establecemos una objeto inicial que dice la hora presente en el sistema.
3. System.out.println("inicio");
4. try {
5. LocalDevice localDevice = LocalDevice.getLocalDevice();//Inicio dispositivo Bluetooth
6. DiscoveryAgent agent = localDevice.getDiscoveryAgent();//agente de búsqueda de dispositivos.
7. text.append("\nIniciando conexión...");
8. conn = (StreamConnection) Connector.open(address);//Intentamos conectarnos al dispositivo definido en el objeto address.
9. long totalTiempo = System.currentTimeMillis() - tiempoInicio; //Volvemos a obtener el tiempo presente y ese dato numérico (long) se le restará el primer valor, así determinamos el tiempo que tardó en hacerse la conexión correctamente.
10. System.out.println("El tiempo de demora es :" + totalTiempo + " miliseg");
11. text.append("El tiempo de demora es :" + totalTiempo + " miliseg");
12. text.append("Connected");
13. } catch (IOException ioe3) {
14. text.append("\nNot connected");
15. } }
```

Cálculo de valor medio y desviación estándar

El valor medio es el valor característico de una serie de datos cuantitativos objeto de estudio que parte del principio de la esperanza matemática o valor esperado, se obtiene a partir de la suma de todos sus valores dividida entre el número de sumandos. Cuando el conjunto es una muestra aleatoriarecibe el nombre de media muestral siendo uno de los principales estadísticos muestrales.

Valor Medio = $\frac{\sum_{i=1}^{n}(Valor_i)}{n}$

La desviación estándar es la medida de la dispersión de los valores respecto a la media (valor promedio).

Desviación estándar = $\sqrt{\frac{\sum(x-\bar{x})^2}{(n-1)}}$

Apéndice D. Teléfonos móviles utilizados en las pruebas de conexión y sus características.

1 Nokia 2330 [Nokia2330Spec08**]**.

De acuerdo con el manual de usuario; la tecnología Bluetooth permite conectar el teléfono mediante ondas de radio, a un dispositivo Bluetooth compatible. Cumple con la especificación **2.0 + *EDR*** de Bluetooth que admite los siguientes perfiles: acceso genérico, intercambio de objeto genérico, manos libres, auricular, objeto push, transferencia de archivos, acceso a la red de área personal, acceso telefónico a redes, aplicación de detección de servicios, acceso a SIM y puerto serial.

Además en el sitio web de nokia para estos modelos encontramos [Nokia2330Spec08]:

Conectividad: Bluetooth 2.0 +EDR.

Perfiles Bluetooth: BIP, DUN, FTP, GAP, GAVDP, GOEP, HFP, HSP, OPP, SAP, SDAP

Plataforma de desarrollo: Series 40 5th Edition, Feature Pack 1 Lite

Sistema Operativo: Nokia OS

Resolución de la pantalla: 128 x 160 px

Tecnología y API´s aceptadas de Java

JSR 139 Connected, Limited Device Configuration (CLDC) 1.1
JSR 118 MIDP 2.1
JSR 185 Java™ Technology for Wireless Industry
JSR 75 FileConnection and PIM API 1.0
JSR 82 Java™ APIs for Bluetooth 1.1
JSR 135 Mobile Media API 1.1
JSR 177 Security and Trust Services API for J2ME™ 1.0 (SATSA-APDU package)
JSR 177 Security and Trust Services API for J2ME™ 1.0 (SATSA-CRYPTO package)
JSR 205 Wireless Messaging API 2.0
Nokia UI API 1.1

2 Alcatel ice3 [AlcatelIce3Spec**]**

Conectividad Bluetooth: Bluetooth A2DP estéreo, V2.0 +EDR.

Software Java MIDP 2.0

Pantalla 2,4 pulgadas, TFT de 65K colores. Resolución 1600 X 1200 píxeles

Perfiles Bluetooth:

Headset (HSP), Handsfree (HFP), Dial-up networking (DUN), Object Push (OPP), Generic Access (GAP), Serial Port (SPP), Service Discovery Protocol (SDP), Generic Audio/Video Distribution (GAVDP), Advanced Audio Distribution (A2DP), Audio/Visual Remote Control Profile (AVRCP), Logical Link Control and Adaptation Protocol, Audio/Video Distribution Transport Protocol (AVDTP), Audio/Video Control Transport Protocol (AVCTP), Phone Book Access (PBAP)

3 Nokia 5310. [Nokia5310Spec07**]**

Conectividad: Bluetooth 2.0 +EDR. Bluetooth Stereo Audio

Perfiles Bluetooth: A2DP, AVRCP, DUN, FTP, GAP, GAVDP, GOEP, HFP, HSP, OPP, PAN, SAP, SDAP, SPP

Plataforma de desarrollo: Series 40 5th Edition, Feature Pack 1

Sistema Operativo: Nokia OS

Resolución de la pantalla: 240 x 320 px

Tecnología y API´s aceptadas de Java

JSR 139 Connected, Limited Device Configuration (CLDC) 1.1
JSR 118 MIDP 2.1
JSR 248 Mobile Service Architecture Subset for CLDC
JSR 75 FileConnection and PIM API 1.0
JSR 82 Java™ APIs for Bluetooth 1.1
JSR 135 Mobile Media API 1.1
JSR 172 J2ME™ Web Services Specification 1.0 (RPC package)
JSR 172 J2ME™ Web Services Specification 1.0 (XML Parser package)
JSR 177 Security and Trust Services API for J2ME™ 1.0 (SATSA-APDU package)
JSR 177 Security and Trust Services API for J2ME™ 1.0 (SATSA-CRYPTO package)
JSR 184 Mobile 3D Graphics API for J2ME™ 1.1
JSR 205 Wireless Messaging API 2.0
JSR 211 Content Handler API 1.0
JSR 226 Scalable 2D Vector Graphics API
JSR 234 Advanced Multimedia Supplements 1.0 (audio3d)

JSR 234 Advanced Multimedia Supplements 1.0 (music)
Nokia UI API 1.1

4 Samsung gt-m2310 [Samsunggtm2310Spec**,**Samsunggtm2310Spec2**]**

Conectividad: Bluetooth 2.0 +EDR. A2DP

Software Java MIDP 2.0

MIDP: 2.0,CLDC: 1.1

Tamaño de la pantalla Java: 128 x 142 px

Tamaño de la pantalla completa de Java: 128 x 160 px

Tecnología y API´s aceptadas de Java

JSR-75 - File System

JSR-75 - PIM

JSR-82 - Bluetooth

JSR-82 - OBEX

JSR-120 - Wireless Messaging

JSR-135 - Mobile Media

JSR-185 – JTWI

Apéndice E. Funciones particulares para la programación de Bluetooth en Java

En este capítulo describimos las funciones particulares de Java que utilizan las entidades cliente y servidor. La API Bluetooth *(javax.bluetooth)* contiene los métodos necesarios para la programación de aplicaciones con Bluetooth, en esta sección describimos algunas métodos que contiene la API y que integramos a nuestro sistema.

Perfiles Bluetooth

Los perfiles Bluetooth proporcionan un conjunto bien definido de procedimientos de la capa superior y la manera de usar uniformemente las capas inferiores de la tecnología Bluetooth. Los perfiles sirven de guía a los desarrolladores sobre cómo implementar una determinada funcionalidad para el usuario final mediante el sistema Bluetooth. Los perfiles que se incluyen con la versión de la especificación Bluetooth 1.1 se denominan perfiles de fundación. Tabla 1 [Bluetooth.com] ofrece una visión general y una breve descripción de estos perfiles.

Perfil	Descripción
Generic Access Profile (GAP)	Es la base de todos los perfiles en el sistema Bluetooth. GAP define las funcionalidades básicas de Bluetooth como establecer los links L2CAP, manejando los modos de seguridad y los modos de descubrimiento de dispositivos.
Serial Port Profile (SPP)	Realiza la emulación del puerto serial (RS-232) basado en el protocolo RFCOMM de la pila Bluetooth.
Dial Up Networking Profile (DUNP)	Define las funciones de un dispositivo Bluetooth para que pueda ser utilizado como puerta de un acceso telefónico a redes.
FAX Profile	Define las funciones de un dispositivo Bluetooth para que pueda ser utilizado como puerta de Fax.
Headset Profile	Define las funciones requeridas para hacer transferencias de audio en auriculares Bluetooth.
LAN Access Point Profile	Define las funciones para habilitar un dispositivo Bluetooth como LAN Access point.
Generic Obex Exchange Profile (GOEP)	Proporciona soporte para el protocolo Obex Exchange (OBEX) sobre enlaces Bluetooth.

Object Push Profile	Define las funciones necesarias para para el intercambio de objetos vCalendars y vCards sobre GOEP.
File Transfer Profile	Define las funciones para navegar a través de carpetas y copiar/crear/ eliminar un archivo o carpeta en el dispositivo Bluetooth, basado en el perfil GOEP.
Synchronization Profile	Define las funciones necesarias para sincronizar Object Stores contiene objetos IrMC (vCards, vMessaging y vNotes) entre dispositivos Bluetooth, basado en el perfil GOEP.
Intercom Profile	Permite a los dispositivos Bluetooth establecer un enlace de comunicación directa similar la comunicación Intercom.
The Cordless Telephony Profile	Permite a los dispositivos Bluetooth actuar como teléfonos comunicados sin cables. Por ejemplo, una entrada ISDN.

Tabla.1 Perfiles base de Bluetooth

Servicio Remoto (*ServiceRecord*)

Para que un dispositivo local utilice un servicio de un dispositivo remoto, deberán tener el mismo protocolo de comunicación. Con el protocolo se puede acceder a una amplia variedad de servicios Bluetooth, los API´S de Java para Bluetooth nos proporcionan mecanismos que permitan todas las comunicaciones, usando RFCOMM, L2CAP u OBEX y demás protocolos para proveer servicios determinados [Mahmoud03].

El API Java para Bluetooth establece mecanismos que permite conexiones a cualquier servicio que utilice RFCOMM, L2CAP, o como su protocolo OBEX. Si utiliza un servicio de otro protocolo (como TCP / IP) en capas por encima de uno de estos protocolos, la aplicación puede acceder al servicio, pero sólo si se implementa el protocolo adicional en la solicitud, utilizando el marco CLDC genéricos de conexión.

Dado que el protocolo OBEX se puede utilizar en varios medios de transmisión diferentes - con cable, infrarrojo, radio Bluetooth, y otros JSR 82 implementa el API OBEX (javax.obex) de forma independiente del núcleo de Bluetooth API (javax.bluetooth). La API OBEX es un paquete separado opcional que puede utilizar con el paquete básico de Bluetooth o de forma independiente.

Los perfiles y los servicios que existen en las APIs de Bluetooth se configuraran a través de un Identificador único universal (UUID). La Norma Internacional especifica el formato y las reglas de generación que permiten a los usuarios producir identificadores de 128 bits que tienen la

garantía de ser únicos globalmente o una gran probabilidad de serlo. Es un identificador estándar usado en el desarrollo de software, es estandarizado por la Open Software Foundation (OSF) como parte del Distributed Computing Environment (DCE). El intento de los UUIDs para habilitar sistemas distribuidos con un único identificador de información. Ahora, cualquiera puede crear un UUID y usarlo para identificar algo con confidencialidad que el identificador nunca será sin intensión por alguien más.

El UUID es definido por el usuario, en nuestro caso es así, nosotros definimos un canal de comunicación entre el cliente el servidor único, por cada conexión que se quiera hacer, pero en el caso de servicios esto cambia, es decir, es decir, hay un organismo encargado de definir lo anterior, un ejemplo es el establecimiento de un servicio FTP.

La clase UUID define los identificadores únicos universales. Estos enteros sin signo de 128-bits están garantizados para ser únicos en todo tiempo y espacio. En consecuencia, una instancia de esta clase es inmutable. La especificación Bluetooth ofrece un algoritmo que describe cómo un UUID de 16-bits o 32-bits podría ser promovido a un UUID de 128-bits. En consecuencia, esta clase proporciona una interfaz que ayuda en la creación de aplicaciones 16-bits, 32-bits, y UUID largo de 128-bits. Los métodos con el apoyo de esta clase permiten probar la igualdad de dos objetos UUID.

El documento de Asignación de Números de Bluetooth que se encuentra en el sitio web oficial de Bluetooth, se definen un gran número de UUID para los protocolos y las clases de servicio. La siguiente tabla proporciona una lista corta de los UUID más comunes definidos en el documento Números Asignados Bluetooth. A continuación son descritos algunos perfiles y sus valores asignados [AsignedNums].

Nombre	Valor	Tamaño
Valor Base de UUID	0x00000000000010008 00000805F9B34FB	128-bits
SDP	0X0001	16-bits
RFCOMM	0X0003	16-bits
OBEX	0X0008	16-bits
HTTP	0X000C	16-bits
L2CAP	0X0100	16-bits

BNEP	0X000F	16-bits
Puerto serial	0x1101	16-bits
ServiceDiscoveryServerServiceClassID	0x1000	16-bits
BrowseGroupDescriptorServiceClassID	0x1001	16-bits
PublicBrowseGroup	0x1002	16-bits
OBEX Object Push Profile	0x1105	16-bits
OBEX File Transfer Profile	0x1106	16-bits
Personal Area Networking User	0x1115	16-bits
NetworkAccessPoint	0x1116	16-bits
GroupNetwork	0x1117	16-bits

Tabla 2 Números asignados para perfiles y protocolos Bluetooth

Cada uno de los perfiles y protocolos tienen funciones particulares. Por ejemplo, en la siguiente tabla aparecen algunos ejemplos de identificadores para determinados servicios con el protocolo OBEX. Ver tabla 3 [AsignedNums]

Perfiles OBEX	UUID
Object Push File OPP	0x1105
File Transfer Profile FTP	0x1106
Synchronizer Profile SYP	0x1104
Basic Image Profile BIP	0x111A
Phone Book Access Profile PBAP	0x1130
Basic Printing Profile	0x1122

Tabla 3 Perfiles OBEX con su identificador.

Ejemplo de UUID para RFCOMM

url = "btspp://localhost:"+ ***0X0003*** +";name=bitiserial"; //Servicio de perfil RFCOMM.

La siguiente url es válida para la localización de un servicio, de acuerdo a la tabla 5.2 será posible establecer el servicio que más convenga a nuestras aplicaciones.

En el siguiente ejemplo establecemos un identificador arbitrario para un servicio particular.

```
UUID uuidSet = new UUID("100001",true);
```

url = "btspp://localhost:"+uuidSet+";name=bitiserial"; //El servicio a utilizar, simple transferencia de bits, perfil RFCOMM. El UUID que hemos utilizado es arbitrario.

Detección de dispositivos Bluetooth (*DeviceDiscovery*)

Para llevar a cabo una conexión a cualquier dispositivo Bluetooth el protocolo indica que, primero demos hacer una búsqueda de los dispositivos que se encuentran en los alrededores, elegirlo e intentar la conexión con éste.

El API de Bluetooth para nos ofrece un método para realizar esta búsqueda, este es llamado ***DeviceDiscovery***. Una aplicación puede obtener una lista de dispositivos utilizando ***startInquiry()*** *o* ***retrieveDevices()*** (este método obtiene la información de los dispositivos que se han conectado del BCC y forman parte de la clase ***DiscoveryAgent***). El método ***startInquiry()*** requiere que la aplicación especifique un objeto ***Listener*** que consiste en habilitar al dispositivo Bluetooth para que detecte a otros dispositivos Bluetooth. El objeto ***Listener*** proviene de la interface ***DiscoveryListener***. Esta interfaz permite a una aplicación especificar un detector de eventos que responda a los eventos relacionados con la indagación. Esta interfaz se utiliza también para la búsqueda de servicios. El método ***deviceDiscovered()*** es llamado cada vez que se encuentra un dispositivo durante una investigación. Cuando la investigación se haya completado o cancelado, la método de ***inquiryCompleted()*** será llamado. Este método recibe como un argumento ya sea el INQUIRY_COMPLETED, INQUIRY_ERROR o INQUIRY_TERMINATED constante para diferenciar entre completado, error o consultas canceladas. El objeto ***listener*** es notificado cuando nuevos dispositivos se encuentran a partir de una búsqueda. Si una aplicación no quiere esperar a una búsqueda para comenzar, el API ofrece la ***retrieveDevices()*** que devuelve la lista de dispositivos que se encontraron ya a través de una búsqueda previa o dispositivos que están clasificados como previamente conectados y conocidos como *"pre-known"*. Los dispositivos *pre-known o* previamente conectados son aquellos dispositivos que se definen en el *Bluetooth Control Center (BCC)* del dispositivo local

como contactos frecuentes. Este método no realiza una ninguna búsqueda, pero proporciona una forma rápida de obtener una lista de dispositivos que pudieran estar en la zona. Una vez que un dispositivo es descubierto, la búsqueda de algún servicio se inicia generalmente. De la búsqueda obtenemos su dirección física, su ***Friendlyname,*** este es escrito arbitrariamente por quien maneja el dispositivo.

Detección de servicios (*ServiceDiscovery*)

La clase ***DiscoveryAgent*** también incluye la funcionalidad del descubrimiento de servicios. La clase proporciona una interfaz para buscar y recuperar los atributos de un determinado servicio. Hay dos modos de búsqueda de servicios. Para buscar un servicio en un único dispositivo, debe ser utilizado el método ***searchServices()***. Por otro lado, si no le importa qué dispositivo cuenta con determinado servicio, el método ***selectService()*** hace la búsqueda de un servicio específico en un conjunto de dispositivos remotos [JSpecs].

Estructura Cliente-Servidor

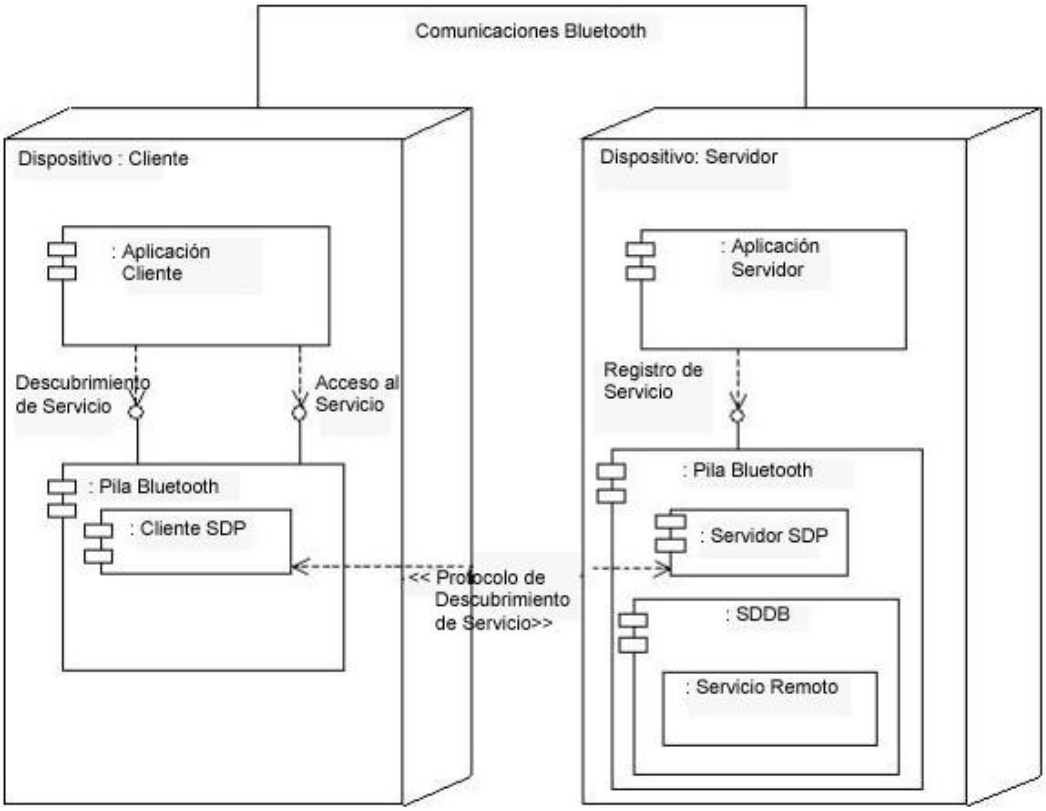
El sistema cuenta con dos entidades de comunicación, éstas forman una estructura cliente-servidor, donde el cliente es capaz de conectarse, comunicarse e intercambiar información con el servidor. El servidor, por su parte, se encarga de controlar las conexiones que se realicen con los clientes. Explicaremos más adelante las funciones de cada una de estas dos entidades involucradas en el sistema.

Un servicio de Bluetooth es una aplicación que actúa como servidor que proporciona algún tipo de asistencia a los dispositivos cliente a través de comunicaciones Bluetooth. Esta ayuda suele darse en forma de una capacidad o función que no está disponible localmente en el dispositivo cliente. Un servicio de impresión es un ejemplo de una aplicación de servidor Bluetooth. Otros ejemplos de aplicaciones de servidor Bluetooth se encuentran en los perfiles de Bluetooth: Los servidores de acceso a LAN, servidores de archivos y objetos, servicios de sincronización, etc. Los programadores pueden definir sus propias aplicaciones de servidor Bluetooth además de los especificados en los perfiles de Bluetooth y hacer que estos servicios disponibles para los clientes remotos. Lo hacen mediante la definición de una hoja de servicios que describe el servicio y la adición de que el historial de servicio a la base de datos de descubrimiento de servicios (SDDB) del dispositivo local [JSpecs].

Después de registrar un servicio en el SDDB, la aplicación de servidor espera a que una aplicación cliente para iniciar el contacto con el servidor y acceder al servicio. En la figura se muestra el esquema cliente-servidor de nuestra estructura de comunicación.

Esquema Cliente-Servidor

El servidor habilita un determinado servicio con *ServiceRecord();* y con el método *AcceptAndOpen();* queda a la espera de conexiones por parte de cliente. Del lado del cliente, éste hace una búsqueda de determinado servicio con el método *ServiceDiscover()*; y cuando ya lo ha encontrado se conecta con el servidor proveedor del servicio con el método *connect();* y finalmente se conecta con el servidor proveedor del servicio con el método *connect();* cuando ya lo ha encontrado. En la figura siguiente[Jsr82Spec] se muestra el proceso que se debe seguir en la estructura cliente-servidor para el establecimiento de servicios.



El servidor proporciona un servicio remoto que permite al cliente conectarse

Servidor Bluetooth

El servidor está listo para establecer la conexión con un dispositivo remoto. También el servidor está previamente configurado para proveer de servicios. Si el dispositivo remoto lo sabe y busca esos servicios procederá a conectarse con él para establecer la comunicación entre ellos.

El servidor está desarrollado en J2SE, es una interfaz gráfica, que sirve para desplegar y realizar alguna de la funciones de éste.

Las responsabilidades típicas de una aplicación de servidor Bluetooth son los siguientes:

• Crear una hoja de servicios que describe el servicio que ofrece la aplicación.

• Agregar un registro de servicio a SDDB del servidor para hacer que los clientes potenciales conozcan este servicio.

• Registro de la seguridad de Bluetooth medidas relacionadas con este servicio que deben aplicarse para las conexiones con los clientes.

• Aceptar conexiones de clientes que solicitan el servicio ofrecido por la aplicación.

• Actualizar la hoja de servicios en SDDB del servidor si las características del cambio de servicio.

• Eliminar o deshabilitar el registro de servicio en SDDB del servidor cuando el servicio ya no está disponible.

La pila de Bluetooth proporciona las siguientes capacidades locales para aplicaciones de servidor Bluetooth:

• Un repositorio para los registros de servicio que permite a los servidores para agregar, actualizar y eliminar sus registros de servicio propia.

• Registro de Asignación de servicio único.

• El establecimiento de conexiones lógicas a las aplicaciones cliente.

Descripción del Servidor

Los dispositivos remotos que se encuentran en los alrededores y su búsqueda forman parte fundamental en las comunicaciones con Bluetooth, pues son necesarias de acuerdo al protocolo de comunicación, para poder localizar algún dispositivo que se tenga que conectar al servidor y viceversa.

El servidor puede establecer los servicios y la manera en que ellos están disponibles a los clientes. Los servicios son llamados *Connect-AnyTime Services* y *Run Before Connect*. A continuación las describimos de acuerdo con la especificación [Jsr82Spec].

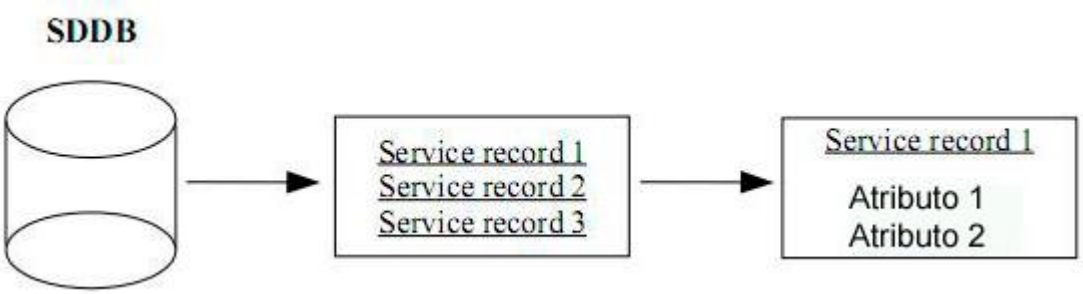
Connect-AnyTime Services. Algunos dispositivos tienen la capacidad de iniciar aplicaciones específicas conforme éstas son requeridas por el cliente en el momento en que trate de conectarse a un servidor que no este activo. Para conseguir realizar esta operación es necesario tener acceso al dispositivo Bluetooth, esto se realiza cuando ya se han almacenado las direcciones de los dispositivos que se han conectado previamente en el *BCC*. Para poder actualizar los datos del dispositivo se tiene que realizar el emparejamiento nuevamente.

Run Before Connect. En este modo de conexión, la aplicación servidor debe estar lista para aceptar conexiones antes de que el cliente intente llevar a cabo la conexión. Las aplicaciones que cumplen con estas características son llamadas *run-before-connect services* de acuerdo

con la especificación de la interfaz de programación de aplicaciones java de Bluetooth [Jsr82Spec].

En el caso de un servicio de ejecución antes de conexión, el ***ServiceRecord*** se añade a la *SDDB* la primera vez que el aplicación de servidor de llamadas ***acceptAndOpen()*** en el notificador asociados. Cualquier modificación en el servidor solicitud que presentó en su ***ServiceRecord*** antes de llamar a ***acceptAndOpen()*** se verá reflejado en la hoja de servicios añadidos a la *SDDB*.

Los dispositivos Bluetooth mantienen la información sobre sus servicios de Bluetooth en la SDDB como se muestra en la Figura [Jsr82Spec]. El servicio contiene SDDB las entradas de registro, donde cada registro contiene los atributos que describen un servicio en particular. Cada servicio tiene su propia entrada en la SDDB.



La base de datos Discovery Service (SDDB)

Múltiple Conexión

El servicio SPP puede aceptar múltiples conexiones de diferentes clientes llamando ***acceptAndOpen()*** repetidamente. Un objeto ***StreamConnection*** se crea para cada conexión aceptada. Cada cliente accede al registro de un mismo servicio y se conecta al servicio utilizando el mismo canal RFCOMM servidor [JSpecs].

Si el sistema Bluetooth de fondo no es compatible con conexiones múltiples, entonces la aplicación de la ***acceptAndOpen()*** lanza una ***BluetoothStateException*.**

Clientes Bluetooth

La aplicación cliente está desarrolla con J2ME, es una interfaz gráfica para aquellos dispositivos que cuenten con las APIs de Java para dispositivos móviles. Esta entidad se encarga de establecer una conexión con el servidor, comunicarse con él y acceder a los servicios que este provee. Las funciones de un cliente Bluetooth son:

- Búsqueda de dispositivos
- Búsqueda de servicios
- Establecimiento de la conexión
- Comunicación

Las responsabilidades típicas de una aplicación cliente Bluetooth son los siguientes:

• Utilice SDP para consultar a distancia los servicios deseados.

• Registro de la seguridad de Bluetooth y las medidas relacionadas con este servicio que deben aplicarse para las conexiones con los servidores.

• Iniciar las conexiones con los servidores que ofrecen los servicios deseados.

• Opcionalmente, acceder a la SDDB para determinar si el servicio ha cambiado o ha de estar disponible.

La pila de Bluetooth proporciona las siguientes capacidades para los clientes remotos de descubrimiento de servicios:

• Búsqueda y recuperación de archivos de servicios almacenados en SDDB del servidor (es decir, actuando como un servidor SDP).

• El establecimiento de conexiones lógicas de aplicaciones de servidor.

La API de Java para dispositivos Bluetooth con las siguientes características:

- Mínimo 512 KB de memoria total disponible (ROM y RAM) (requisitos de las aplicaciones de memoria adicional).
- Conexión Bluetooth.
- Cumplimiento de la aplicación de la configuración de J2ME Limitada de Dispositivos Conectados (CLDC).

El sistema Bluetooth sobre los que la API de Java se construirá también debe cumplir con ciertos requisitos:

- El sistema subyacente debe ser "calificado" de acuerdo con el Programa de Calificación Bluetooth, por lo menos los perfiles, Generic Access Profile, Service Discovery Application Profile y el Serial Port Profile.

- El sistema debe ser compatible con tres capas de comunicación o protocolos, definidos en la especificación Bluetooth 1.1, y la aplicación de esta API debe tener acceso a ellos, Service Discovery Protocol (SDP), Radio Frecuencia Protocolo de Comunicaciones (RFCOMM), y Control de enlace lógico y Protocolo de adaptación (L2CAP).

- El sistema debe proveer un control de Bluetooth Center (BCC), un panel de control muy similar a la aplicación que permite a un usuario o un ***OEM*** para definir valores específicos para ciertos parámetros de configuración en una pila.

OBEX se puede proporcionar en el sistema Bluetooth de base o por la implementación de la API. El protocolo OBEX proporciona soporte para el intercambio de objetos, y constituye la base del perfil de Bluetooth, tales como el perfil de sincronización y el perfil de transferencia de archivos.

Detección de Servicios (ServiceDiscovery)

Service Discovery (SDP) se refiere al descubrimiento de servicios de Protocolo de la capa que es utilizado cada vez que se desea encontrar servicios en un dispositivo Bluetooth remoto. Esta interface hace uso de DiscoveryAgent. La clase DiscoveryAgent también incluye la funcionalidad proporcionada por el perfil de la aplicación de descubrimiento de servicios. La clase proporciona una interfaz para una aplicación para buscar y recuperar los atributos de un determinado servicio. Hay dos modos de búsqueda de servicios. Para buscar un servicio en un único dispositivo, el método ***searchServices()*** debe ser utilizado. Por otro lado, si no le importa qué dispositivo está en un servicio, el método ***selectService()*** lo hace un servicio de búsqueda en un conjunto de dispositivos remotos [JSpecs].

Bibliografía

Swarnalatha10	Swarnalatha P., Srinivasan A., Kishore G. Intra-communication for banking using Bluetooth Communication. School of Computing Sciences. VIT University.Vellore- 632 014. Vellore, P.C.632 009 India.
Boja10	Boja C., Batagan L., Doinea M., Zamfiroiu Alin. Secure Bluetooth Services in an M-Learning Environment. Economic Informatics Department. Academy of Economic Studies. Romana Square Número. 6, Bucharest, Romania.
Pocatilu09	Pocatilu P., Boja C. Quality characteristics and metrics related to m-learning process, Amfiteatru Economic, , Volúmen 11, número. 26, pp. 346-354 ISSN 1582-9146.
Yow10	Yow K., Tjoa J., Lee K. Bluetooth Park State Scheduling Algorithm for BlueBus, an On-Board Infotainment System for Public Buses in Singapore. School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798 SINGAPORE1
Sriskanthan02	Sriskanthan N., Tan F. School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore, Singapore 639798.Received 17 Septiembre 2001;
Schwingenschoegl00	Schwingenschoegl C., Heigl A. Development of a Service Discovery Architecture for the Bluetooth Radio System. Technische Universität München (TUM), Institute of Communication Networks Arcisstr. 21, D-80333 Munich, 2000.
Lin09	Lin R., Lin P., Shiao W., Lin S. Cultural aspect of interaction design beyond human-computer interaction. Internationalization, Design and Global Development. Proceedings Third International Conference, IDGD 2009. Held as Part of HCI International 2009. Springer Verlag. pp. 49-58. Berlin, Alemania.
Bischoff09	Bischoff O., Wang X., Rainer L., Steffen P. Localization in Wireless Ad-hoc Sensor Networks using Multilateration with RSSI for Logistic Applications. Institute for Electromagnetic Theory and Microelectronics (ITEM), University of Bremen, Otto-Hahn-Allee NW1, D-28359 Bremen, Alemania. Agosto 2009.
Davidrajuh09	Davidrajuh R., Evaluating Performance of a Bluetooth-Based Classroom Tool. University of Stavanger, Department of Electrical Engineering & Computer Science PO Box 8002, 4036 Stavanger, Norway Email: reggie.davidrajuh@uis.no. Journal: International Journal of Mobile Learning and Organisation. Volúmen 3 número 2, Abril 2009

Chavan09	Chavan G. Design and analysis of a mobile file sharing system for opportunistic networks. University of Texas at Arlington. Agosto 2009
Pahlavan02	Pahlavan K., Krishnamurthy P. Principles of Wireless Networks. Prentice Hall 2002
Benbunan07	Benbunan R., Benbunan A. Understanding user behavior with new mobile applications. aSCIS Department, Baruch College, CUNY, Box B11-220, New York, NY 10010, United States Mobile Dreams Factory, Madrid, España. Noviembre 2006.
Miller01	Miller M. Discovering Bluetooth. SYBEX Inc., 1151 Marina Village Parkway, Alameda, CA 94501. 2001
Me05	Me G., Schuster A. A mobile local payment system Bluetooth based. Dipartimento di Informatica, Sistemi e Produzione,Università di "Tor Vergata", Roma, 00183, Italy. Fakultät Informatik, Technische Universität München, Germany. International Symposium on Wireless Communications (ISWSN'05) 2005
Shin10	Shin M, Cornelius C., Peebles D., Kapadia A., Kotz D., Triandopoulos N. AnonySense: A system for anonymous opportunistic sensing Original Research Article. Pervasive and Mobile Computing, Volúmen 7, número 1, Febrero 2011, pp. 16-30
Hahn10	A.G. Hahn, R.J.N. Helmer, T. Kelly, K. Partridge, A. Krajewski, I. Blanchonette, J. Barker, H. Bruch, M. Brydon, N. Hooke, B. Andreass Development of an automated scoring system for amateur boxing Original Research ArticleProcedia Engineering, Volúmen 2, número 2, Junio 2010, pp. 3095-3101
Szweda00	Roy Szweda. POF vs Bluetooth Original Research Article. III-Vs Review, Volúmen 13, Número 3,Junio 2000, Page 55
García00	García de Jalón J., Rodríguez J., Mingo I., Imaz A., Brazález A., Larzabal A., Calleja J., García J. Aprenda Java como si estuviera en primero. Escuela superior de ingenieros, Universidad de Navarra. San Sebastián, Enero 2000
Rodrin06	Rodrin D. Personal Wireless Communications. A Graduate Research Report Submitted for INSS 690. In Partial Fulfillment of the Requirements of the Degree of Master of Science in Management Information Systems. Bowie State University. Maryland in Europe. Julio 2006
Klingsheim04	Klingsheim André N. J2ME Bluetooth Programming. NoWires.org. Julio, 2004.
Benjuya	Benjuya D., Duro A., Peralta R. Java 2 Micro Edition. Introducción a java para dispositivos móviles. Seminario en la Universidad Politécnica de Cataluña.

Alonso09	Redes Privadas Virtuales. Ed. Alfaomega 2009
Mahmoud03	Mahmoud Q. Article. Part II: The Java APIs for Bluetooth Wireless Technology. Abril 2003
Hopkins05	Bluetooth boogies, Part 1: File transfer with JSR-82 and OBEX http://www.ibm.com/developerworks/wireless/library/wi-boogie1/?ca=dgr-lnxw07BluetoothJSR-82, Noviembre 2011.
Horstmann05	Horstmann J. Migration to Open Source Databases. Technical University Berlin. Septiembre, 2005. Revisado Noviembre 2011
IIIVsReview05	Mobile phones and wireless gain strength. III-Vs Review, Volúmen 18, número 9, Diciembre 2005-Enero 2006, Página 6
EuropeanCommission	European Commission – Mobile use up, consumer prices down: Europe's telecoms sector weathering economic downturn, says Commission report http://europa.eu/rapid/pressReleasesAction.do?reference =IP/09/473; Noviembre 2011
eeherald11	http://www.eeherald.com/section/news/nws2011082211.html Agosto 2011. EE Herald 2011
Jsr82Spec	PDF Java. TM APIs for BluetoothTM. Wireless Technology (JSR-82). Mandar a dirección de página. Noviembre 2011
JSpecs	Java specifications/btapi_javadocs/javax/bluetooth/UUID.html Noviembre 2011
Bluecove	http://www.bluecove.org/ Noviembre 2011
Powers07	http://developers.sun.com/mobility/midp/articles/gamepart3/#1 Revisado Noviembre 2011
BluetoothSIG	Bluetooth Special Interest Group. Bluetooth specification version 1.1 and 1.2. http://www.bluetooth.com, Noviembre 2011
Bluetooth.com	http://spanish.bluetooth.com/Bluetooth/Technology/Works/Overview_of_Operation.htm Noviembre 2011
BluetoothBasics11	http://www.bluetooth.com/Pages/Basics.aspx Bluetooth SIG,Noviembre 2011

Wiki10	Pila Bluetooth http://es.wikipedia.org/wiki/Pila_Bluetooth Noviembre 2011
FORUMNOKIA06	PC Connectivity over Bluetooth in JavaTM Applications Noviembre 2011
Nokia2330Spec 08	http://www.developer.nokia.com/Devices/Device_specifications/2330_classic/ 4 November 2008. Nokia 2011.
Nokia5310Spec 07	http://www.developer.nokia.com/Devices/Device_specifications/5310_Xpress Music/ 29 August 2007. Nokia 2011
AlcatelIce3Spec	http://www.phonearena.com/phones/Alcatel-OT-808a_id4390/fullspecs Noviembre 2011
Samsunggtm23 10Spec	http://www.samsung.com/mx/consumer/mobile-phones/mobile-phones/telcel/GT-M2310BBLTCE/index.idx?pagetype=prd_detail&tab=specification Noviembre 2011
Samsunggtm23 10Spec2	http://www.mobilerated.com/samsung-gt-m2310-specifications.html Kalador Entertainment Inc. Revisado: 30.11.2011
Jcp11	http://www.jcp.org/en/home/index Noviembre 2011
MySQL	http://www.mysql.com/why-mysql/ Noviembre 2011
Adobe	http://kb2.adobe.com/cps/147/tn_14787.html Noviembre 2011
UUIDWIK	http://en.wikipedia.org/wiki/Universally_Unique_Identifier Noviembre 2011
AsignedNums	http://www.bluetooth.org/Technical/AssignedNumbers/home.htm Noviembre 2011
Hopkins06	Hopkins Bruce. Article: Using the JSR-82 API for OBEX Image Transfers. Oracle Sun developer network SDN. November, 2006. http://developers.sun.com/mobility/apis/articles/bluetoothobex/ Noviembre 2011
Ticbeat12	http://www.ticbeat.com/sim/oracle-pone-java-dispositivos-ios-android/ Revisado: 11.07.2012

GLOSARIO

ACL Access Control List. Es un concepto de seguridad informática usado para fomentar la separación de privilegios. Es una forma de determinar los permisos de acceso apropiados a un determinado objeto, dependiendo de ciertos aspectos del proceso que hace el pedido.

AFH Adaptive frequency hopping. Es una técnica utilizada para mejorar el rendimiento de Bluetooth para disminuir el impacto de tal interferencia. También se conoce como AFH, esta técnica puede ser implementada a través de varios métodos, cada uno con su propio conjunto inherente de las ventajas y desventajas. Ericsson, líder en el campo de la tecnología inalámbrica Bluetooth, utiliza un método muy adecuado para su amplia solución de diseño basado en Bluetooth se venden como propiedad intelectual (IP). Implementación de Ericsson de la AFH se ve reforzada a través de la utilización de otras técnicas como estándar, brindando una excelente calidad de audio para aplicaciones centradas en la voz de la presencia de las tecnologías inalámbricas múltiples.

API Application programming interface. Una interfaz de programación de aplicaciones o API (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas (también denominadas comúnmente "librerías").

BCC Bluetooth Control Center . Es un software de gestión en el dispositivo que sirve como la autoridad central para cambiar la configuración local de Bluetooth: Encendido o apagado del radio Bluetooth, estableciendo el nombre del dispositivo para hacerlos visible en la detección de dispositivos, habilitar o deshabilitar el modo de descubrimiento de dispositivos, la introducción de números PIN, el establecimiento de los atributos de seguridad por defecto.

Bluetooth Friendlyname. Es una cadena de caracteres que identifican al dispositivo en lugar de su representación con números en sistema hexadecimal de su dirección física.

DBMS. Los sistemas de gestión de bases de datos (en inglés database management system, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Desviación estándar Dstd. La desviación estándar es la medida de la dispersión de los valores respecto a la media (valor promedio).

EDR Enhanced Data Rate. Permite mejorar las velocidades de transmisión en hasta 3Mbps a la vez que intenta solucionar algunos errores de la especificación 1.2.

FHSS Frequency Hopping Spread Spectrum. Espectro ensanchado por salto de frecuencia es una técnica de modulación en espectro ensanchado en el que la señal se emite sobre una serie de radiofrecuencias aparentemente aleatorias, saltando de frecuencia en frecuencia sincrónicamente con el transmisor. Los receptores no autorizados escucharán una señal ininteligible. Si se intentara interceptar la señal, sólo se conseguiría para unos pocos bits. Una transmisión en espectro ensanchado ofrece 3 ventajas principales:

Las señales en espectro ensanchado son altamente resistentes al ruido y a la interferencia.

Las señales en espectro ensanchado son difíciles de interceptar. Una transmisión de este tipo suena como un ruido de corta duración, o como un incremento en el ruido en cualquier receptor, excepto para el que esté usando la secuencia que fue usada por el transmisor.

Transmisiones en espectro ensanchado pueden compartir una banda de frecuencia con muchos tipos de transmisiones convencionales con mínima interferencia.

Firebird. Es un sistema de administración de base de datos relacional (o RDBMS) (Lenguaje consultas: SQL) de código abierto, basado en la versión 6 de Interbase, cuyo código fue liberado por Borland en 2000. Su código fue reescrito de C a C++. El proyecto se desarrolla activamente, el 18 de abril de 2008 fue liberada la versión 2.1 y el 26 de diciembre de 2009 fue liberada la versión 2.5.0 RC1.

Firmware. El firmware es un bloque de instrucciones de programa para propósitos específicos, grabado en una memoria de tipo no volátil (ROM, EEPROM, flash, etc), que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo. Al estar integrado en la electrónica del dispositivo es en parte hardware, pero también es software, ya que proporciona lógica y se dispone en algún tipo de lenguaje de programación. Funcionalmente, el firmware es el intermediario (interfaz) entre las órdenes externas que recibe el dispositivo y su electrónica, ya que es el encargado de controlar a ésta última para ejecutar correctamente dichas órdenes externas.

GPL GNU General Public Licence. La Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License o simplemente sus siglas del inglés GNU GPL, es una licencia creada por la Free Software Foundation en 1989 (la primera versión), y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

GUI Graphic User Interface. La interfaz gráfica de usuario es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador.

Handshaking. Handshaking es una palabra inglesa cuyo significado es "apretón de manos" y que es utilizada en tecnologías de la información, telecomunicaciones, y otras conexas. Handshaking es un proceso automatizado de negociación que establece de forma dinámica los parámetros de un canal de comunicaciones establecido entre dos entidades antes de que comience la comunicación normal por el canal. De ello se desprende la creación física del canal y precede a la transferencia de información normal.

IDE Integrated Development Environment. Un entorno de desarrollo integrado es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Un IDE es un entorno de programación q ue ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Ingres. Es un sistema gestor de bases de datos relacional SQL de código abierto.destinado a soportar grandes aplicaciones comerciales y gubernamentales. Actium Corporation controla el desarrollo de Ingres y hace que la certificación los binarios disponibles para su descarga, así como la prestación de soporte.

J2ME Java 2 Micro Edition. La plataforma Java Micro Edition, o anteriormente Java 2 Micro Edition(J2ME), es una especificación de un subconjunto de la plataforma Java orientada a proveer una colección certificada de APIs de desarrollo de software para dispositivos con recursos restringidos. Está orientado a productos de consumo como PDAs, teléfonos móviles o electrodomésticos.

J2SE Java 2 Standard Edition. Java Platform, Standard Edition o Java SE (conocido anteriormente hasta la versión 5.0 como Plataforma Java 2, Standard Edition o J2SE), es una colección de APIs del lenguaje de programación Java útiles para muchos programas de la Plataforma Java. La Plataforma Java 2, Enterprise Edition incluye todas las clases en el Java SE, además de algunas de las cuales son útiles para programas que se ejecutan en servidores sobre workstations.

JCP *Java Community Process.* Establecido en 1998, es un proceso formalizado el cual permite a las partes interesadas a involucrarse en la definición de futuras versiones y características de la plataforma Java.

***JDK Java Development Kit* .** Es un software que provee herramientas de desarrollo para la creación de programas en Java. Puede instalarse en una computadora local o en una unidad de red.

JSR Java Specification Request. Son las descripciones reales de las especificaciones propuestas y definitivas para la plataforma Java. En un momento dado hay numerosos JSRs movimiento a

través del proceso de revisión y aprobación.Una manera sencilla de estar al día y realizar un seguimiento de la JSR en cada etapa de la revisión se unirá a la lista de correo. Como miembro de la lista de correo, automáticamente puede recibir correos electrónicos en JSR medida que avanzan por las etapas de revisión.

JVM Java virtual Machine. Es un máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java

El código binario de Java no es un lenguaje de alto nivel, sino un verdadero código máquina de bajo nivel, viable incluso como lenguaje de entrada para un microprocesador físico. Como todas las piezas del rompecabezas Java, fue desarrollado originalmente por Sun Microsystems.

Listener. Es un objeto del lenguaje de programación Java que proviene del método (DeviceDiscovery), consiste en habilitar al dispositivo Bluetooth para que esté listo para detectar a otros dispositivos Bluetooth.

MaxDB. Es un sistema de administración de bases de datos adquirido por la compañía SAP para usarse como un repositorio de datos para las aplicaciones de SAP. Desde su adquisición fue nombrado SAPDB y, posteriormente, renombrado a MaxDB por MySQL AB. Antes de SAPDB, la base de datos pasó por otros nombres. Originalmente, se llamaba Adabas D, una base de datos pre-relacional.

MPayment. Es un método alternativo de pago donde se puede utilizar un teléfono móvil para pagar servicios o bienes digitales.

MS/s MSs (Millones de muestras por segundo). En la conversión de datos, una señal analógica se convierte en una secuencia de números, cada uno representando a la amplitud de la señal analógica en un momento en el tiempo. Cada número se llama "muestra". El número de muestra por segundo se denomina frecuencia de muestreo, medido en muestras por segundo.

MySQL. Es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.1 MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual

OEM Original Equipment Manufacturer. Es una empresa que fabrica productos que luego son comprados por otra empresa y vendidos bajo la marca de la empresa compradora.

P2P Peer-to-peer. Una red Peer-to-Peer o red de pares o red entre iguales o red entre pares o red punto a punto (P2P, por sus siglas en inglés) es una red de computadoras en la que todos o algunos aspectos funcionan sin clientes ni servidores fijos, sino una serie de nodos que se

comportan como iguales entre sí. Es decir, actúan simultáneamente como clientes y servidores respecto a los demás nodos de la red. Las redes P2P permiten el intercambio directo de información, en cualquier formato, entre los ordenadores interconectados.

PAN Personal Area Network. Es una red de computadoras para la comunicación entre distintos dispositivos (tanto computadoras, puntos de acceso a internet, teléfonos celulares, PDA, dispositivos de audio, impresoras) cercanos al punto de acceso. Estas redes normalmente son de unos pocos metros y para uso personal, así como fuera de ella.

Piconet. Se conoce como piconet a una red de dispositivos informáticos que se conectan utilizando Bluetooth.

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyados por organizaciones comerciales. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

PPP Point-to-point protocol. (*Protocolo punto a punto*), es un protocolo de nivel de enlace estandarizado en el documento RFC 1661. Por tanto, se trata de un protocolo asociado a la pila TCP/IP de uso en Internet.

QFE Quick Fix Engineering, conocido como hotfix. Un hotfix es un paquete que puede incluir varios archivos y que sirve para resolver un bug específico dentro de una aplicación informática.

Red Oportunista. Es una red de nodos que son conectados inalámbricamente. Las redes oportunistas soportan interacción espontánea entre usuarios móviles que utilicen dispositivos móviles inalámbricos. Los nodos en las redes oportunistas aprovechan el hecho de que puedan comunicarse unos con otros cada vez que entran dentro de cada uno alcance. Los nodos son móviles o fijos.

SAP AG (Systeme, Anwendungen und Produkte) (Sistemas, Aplicaciones y Productos) es una empresa de informática alemana con sede en Walldorf. Comercializa un conjunto de aplicaciones de software empresarial, entre ellas mySAP Business Suite, que provee soluciones escalables, es decir, con capacidad de adaptarse a nuevos requisitos conforme cambian o aumentan las necesidades del negocio del cliente, con más de 1.000 procesos de negocio, que la empresa argumenta se encuentran entre las mejores prácticas empresariales.

SDK Software Development Kit. Un kit de desarrollo de software o SDK (siglas en inglés de software development kit) es generalmente un conjunto de herramientas de desarrollo que le

permite a un programador crear aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de software, frameworks, plataformas de hardware, computadoras, videoconsolas, sistemas operativos, etc.

SIG Special Interest Group. Es un grupo de fabricantes de dispositivos electrónicos que se encarga especialmente en el desarrollo de la tecnología inalámbrica Bluetooth. Actualmente está conformado por Ericsson, Intel, Nokia, Toshiba, 3Com, Agere, Microsoft y Motorola.

Symbian. Es un sistema operativo que fue producto de la alianza de varias empresas de telefonía móvil, entre las que se encuentran Nokia, Sony Ericsson, Psion, Samsung, Siemens, Arima, Benq, Fujitsu, Lenovo, LG, Motorola, Mitsubishi Electric, Panasonic, Sharp, etc. Sus orígenes provienen de su antepasado EPOC32, utilizado en PDA's y Handhelds de PSION. El objetivo de Symbian fue crear un sistema operativo para terminales móviles que pudiera competir con el de Palm o el Windows Mobile de Microsoft y ahora Android de Google Inc., iOS de Apple Inc. y BlackBerry OS.

Thread. En sistemas operativos, un **hilo de ejecución** o subproceso es la unidad de procesamiento más pequeña que puede ser planificada por un sistema operativo. Un hilo es una característica que permite a una aplicación realizar varias tareas a la vez (concurrentemente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, situación de autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente.

Valor Medio. En matemáticas y estadística, la media aritmética (también llamada promedio o simplemente media) de un conjunto finito de números es el valor característico de una serie de datos cuantitativos objeto de estudio que parte del principio de la esperanza matemática o valor esperado, se obtiene a partir de la suma de todos sus valores dividida entre el número de sumandos. Cuando el conjunto es una muestra aleatoriarecibe el nombre de media muestral siendo uno de los principales estadísticos muestrales.

WAP Wireless Application Protocol. (protocolo de aplicaciones inalámbricas) es un estándar abierto internacional para aplicaciones que utilizan las comunicaciones inalámbricas, p.ej. acceso a servicios de Internet desde un teléfono móvil.