ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO DE TELECOMUNICACIÓN

Título del proyecto:

SISTEMA BLUETOOTH PARA EL CONTROL DE DISPOSITIVOS

Alumno: Héctor Martínez Buldáin

Tutor: Dr. D. Jesús Villadangos Alonso

Pamplona, 30 de abril de 2010

INDICE

INTRODUCCIÓN

Muchos de los aparatos comercializados actualmente tienen incorporadas tecnologías inalámbricas, como por ejemplo, Bluetooth, que cada vez esta más difundida ya que viene incluida en la mayoría de teléfonos móviles actuales así como en muchas pdas, o incluso en portátiles y notebook. A pesar de esto sus aplicaciones son pocas como por ejemplo los manos libre de los vehículos, auriculares inalámbricos y poco más.

Al igual que los móviles y otros muchos dispositivos portátiles utilizan la tecnología Bluetooth para comunicaciones de corto alcance también cada día son más los dispositivos portátiles que incorporan la maquina virtual de Java que les permite ejecutar aplicaciones creadas en este lenguaje, pero al igual que ocurre con el Bluetooth de los móviles las aplicaciones java que incorporan los móviles están acotadas a juegos y poco más.

A partir de estas dos tecnologías se pueden crear muchísimas aplicaciones nuevas con utilidades muy diversas, desde nuevas comunicaciones entre teléfonos cercanos, más allá del envío de fotos, comunicaciones entre teléfonos y ordenadores, o comunicaciones entre teléfonos y dispositivos adaptados a utilidades concretas como la que se aborda en este proyecto.

El objeto de este proyecto es facilitar el manejo de dispositivos eléctricos comunes en nuestras viviendas o en los lugares de trabajo a través de una conexión inalámbrica creada entre el teléfono móvil y un módulo Bluetooth que controlara el funcionamiento de dispositivos eléctricos mediante comandos recibidos vía Bluetooth.

Con este proyecto se pueden controlar: una cerradura eléctrica y un punto de luz y la llamada del ascensor, de la misma manera seria posible controlar otros dispositivos que también funcionen con 230 V. El sistema es capaz de controlarlos vía Bluetooth sin afectar a la forma convencional de su funcionamiento, es decir, con sus propios interruptores eléctricos.

Para poder establecer la conexión, el móvil debe contar con una aplicación Java que utiliza el Bluetooth para establecer una conexión. Una vez realizada la conexión, el móvil envía comandos que son recibidos por el módulo de control para realizar una acción específica. La aplicación se desarrolla sobre J2ME, una plataforma Java creada especialmente para dispositivos pequeños que tengan limitadas sus prestaciones.

El módulo Bluetooth que se utiliza en la parte de control es el KC-21, este dispone de terminales programables de forma remota. Aprovechando esta característica, la aplicación Java se conecta vía Bluetooth con el módulo KC-21 y le envía comandos, que harán que dicho módulo trasfiera un valor específico a cada conector. Este conector habilitará un circuito acoplador de potencia, de esta manera, el estado que tenga el conector del módulo determinara el estado del dispositivo eléctrico.

En el capitulo 1 se determinan las características propias de la tecnología Bluetooth, las cuales pueden ser aprovechadas en aplicaciones de control de dispositivos eléctricos dentro del hogar o del lugar de trabajo. Destacando las bondades del Bluetooth en comunicaciones de corto alcance. Además se hace una comparación con otras tecnologías inalámbricas para determinar lo ventajoso que resulta usarla y el porque no se ha utilizado otra.

En el capitulo 2 se hace una descripción del Lenguaje Java poniendo énfasis en Java Micro Edition, un subconjunto creado para dispositivos pequeños, explicando algunas de las herramientas básicas que debe incorporar cada aplicación para ser manejada de forma intuitiva y sencilla por el usuario ya que los dispositivos tan pequeños tienen muy limitadas sus prestaciones y J2ME se diseño considerando estas limitaciones.

También se revisa la Configuración de Dispositivos Conectados **CDC** que hace referencia a **Connected Device Configuration**. Esta es una especificación realizada por Sun dentro del conjunto de tecnologías para computación móvil conocido como J2ME (Java 2 Mobile Edition). Define las capacidades básicas que debe tener un dispositivo móvil con capacidad de conexión y cuya potencia de computación se encuentra entre la de un teléfono o PDA y la de un ordenador.

En el capitulo 3 se resume el desarrollo de la aplicación, se analizan las clases principales que se usan y se muestran partes del código del programa con su resultado en el simulador. El programa esta analizado explicando la interfaz de usuario y la interfaz de comunicación.

En el capitulo 4 se detalla el diseño y construcción del circuito electrónico que adapta el módulo Bluetooth a un sistema de control previo, en el que se programa dicho dispositivo a través del puerto serie para su posterior funcionamiento. También se estudian los circuitos de alimentación y los circuitos acopladores de potencia.

En el capitulo 5 sintetiza la implementación del sistema de control completo que controla un punto de luz y una cerradura eléctrica y un ascensor, y se realizan las pruebas de funcionamiento correspondientes.

Estas pruebas permitieron establecer las conclusiones que se detallan en el capitulo 6, en el que también se hacen recomendaciones en base a la experiencia adquirida durante la realización de este proyecto.

CAPÍTULO 1. TECNOLOGÍA BLUETOOTH

Hoy en día, la tecnología se encuentra muy difundida entre los dispositivos móviles que se encuentran en el mercado, principalmente teléfonos móviles. Las características de esta tecnología permiten usarla como enlace de comunicación entre dispositivos que no deben ser necesariamente del mismo tipo, por ejemplo, un PC con un teléfono o una PDA con unos audífonos, o un PC con los audífonos, etc. Por ello, aprovechando la misma tecnología ya incluida en estos equipos se pueden crear nuevas aplicaciones.

El objetivo principal del presente proyecto es reutilizar dispositivos existentes en el mercado para darles un uso nuevo utilizando la tecnología Bluetooth. En este caso, se controlará remotamente dispositivos dentro del hogar desde un teléfono móvil. El dispositivo a utilizar debe tener ciertas características que serán analizadas más adelante.

1.1 MANEJO DE LA TECNOLOGÍA EN EQUIPOS MÓVILES

La tecnología Bluetooth consta de tecnología de radio, un conjunto de protocolos y varios perfiles de interoperabilidad. Cada dispositivo Bluetooth esta provisto de uno o más perfiles que identifican el tipo de dispositivo y su aplicación.

El uso de la tecnología Bluetooth dentro de los equipos móviles se facilita al utilizar aplicaciones basadas en lenguajes de programación, como por ejemplo Java, que se ejecutan sobre sus sistemas operativos. Las aplicaciones Java para dispositivos móviles se encuentran muy difundidas entre la mayoría de equipos actuales.

Java es un conjunto de tecnologías que abarca a todos los ámbitos de la programación a nivel mundial. En 1999 Sun Microsystems lanzo la plataforma Java 2 Platform Micro Edition [XIII) para desarrollar aplicaciones para pequeños dispositivos, que tienen como tal, algunas capacidades reducidas como su interfaz gráfico, procesamiento, memoria, etc.

Java ME (J2ME) permite interactuar con otros dispositivos Bluetooth de manera eficiente. Las interfaces generadas para las aplicaciones a través de Java se conocen como APIs, *Application Programming Interfaces,* [XIII) y son la base de las aplicaciones.

Los APIs de Bluetooth se encuentran definidos en la especificación de Java JSR 82 (APIs para Bluetooth). Usando APIs sobre la configuración conocida como *Connected Limited Device Configuration* (CLDC) de Java ME [1) se pueden crear aplicaciones Java ME conocidas como MIDlet. El API de Bluetooth, usado en un MIDlet[5), posibilita el descubrimiento y registro de dispositivos, descubrimiento y registro de servicios, establecimiento de la comunicación, envío y recepción de datos, acceso y control sobre otros dispositivos que soporten Bluetooth.

La integración de Bluetooth con otras herramientas, como la plataforma Java, abre una ventana inmensa de posibilidades para la creación de aplicaciones que pueden ser diseñadas más a la medida de pequeños usuarios.

1.2 SISTEMA BLUETOOTH PARA CONTROL DE DISPOSITIVOS

Para hacer posible un control de dispositivos eléctricos usando Bluetooth, es necesario que estos también cuenten con la tecnología, o bien sean adaptados a otros que si lo hagan. Para el control del acceso (Por ejemplo: un portero automático) y la iluminación (Por ejemplo: una lámpara) de una habitación, se puede emplear módulos Bluetooth que les proporcionen esta característica.

Entonces no Sería necesario cambiar todos los dispositivos de la casa sino más bien adaptarlos.

El sistema implementado constaría de un dispositivo Mobil (teléfono móvil), un módulo Bluetooth y un dispositivo eléctrico común. Dentro del teléfono se implementara una aplicación que establezca la conexión y sea la interfaz entre el usuario y el enlace. El módulo Bluetooth consta de un sistema de radio, una antena integrada, un pequeño procesador y una memoria flash. El módulo puede ser programado para interactuar con el dispositivo que se desea controlar. Una vez establecida la conexión, el equipo móvil será capaz de enviar comandos que controlan el dispositivo eléctrico a través del módulo Bluetooth.

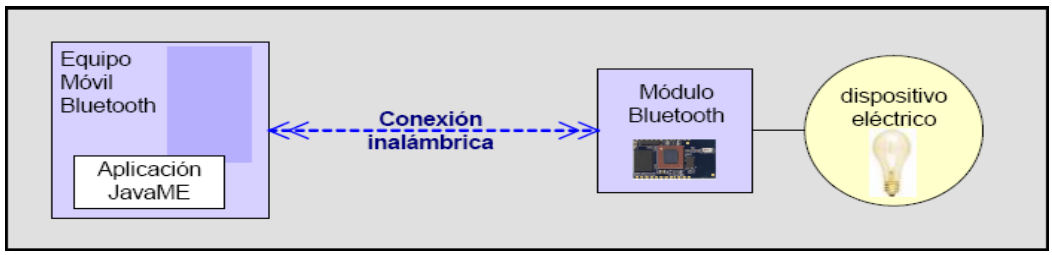


Figura 1.1 Esquema del sistema de control simple

1.2.1 MÓDULO BLUETOOTH

Un módulo Bluetooth tiene una parte de Software y una de Hardware. Por ejemplo, en el módulo KC-21, que Sería utilizado en este proyecto, la parte de hardware esta compuesta por: un dispositivo de radio, encargado de modular y transmitir la señal; y un controlador digital. El controlador digital esta compuesto por una CPU, por un procesador de señales digitales, *Digital Signal Processor* (DSP), llamado controlador de Enlace, *Link Controller* (LC) y las interfaces de comunicación.

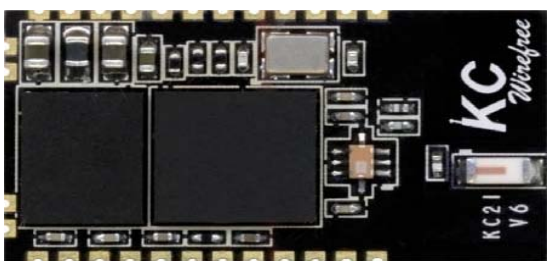


Figura 1.2 Módulo Bluetooth KC-21

El LC o *Link Controller* esta encargado de hacer el procesamiento de banda base y del manejo de los protocolos ARQ y FEC de capa física. Además, se encarga de las funciones de transferencia, tanto asíncrona como sincronía, codificación de audio y encriptación de datos.

La CPU se encarga de atender las instrucciones del dispositivo relacionadas con Bluetooth para axial simplificar su operación. Para ello, sobre la CPU ejecuta un software denominado *Link Manager* que tiene la función de comunicarse con otros dispositivos por medio del protocolo LMP.

Entre las tareas realizadas por el LC y el *Link Manager* destacan las siguientes: envío y recepción de datos, empaginamiento y peticiones, determinación de conexiones, autenticación, negociación y determinación de tipos de enlace, determinación del tipo de cuerpo de cada paquete, ubicación del dispositivo en modo sniff o hold.

1.3 LA ESPECIFICACIÓN BLUETOOTH.

Bluetooth es una especificación que define un estándar global de comunicaciones inalámbricas para redes de área personal, que permite la transmisión de voz y datos entre diferentes equipos mediante un enlace por radiofrecuencia en entornos de comunicaciones móviles y fijas. La especificación Bluetooth esta recogida por el grupo de trabajo 802.15.1 del IEEE.

1.3.1 DESCRIPCIÓN TÉCNICA DE BLUETOOTH

La tecnología inalámbrica Bluetooth está orientada a aplicaciones de voz y datos.
La frecuencia de radio con la que trabaja se sitúa en el rango de 2400 – 2483,5 MHz. de la banda Industrial Científica y Medica, *Industrial Scientific and Medical* (ISM), disponible a nivel mundial y que no requiere licencia.

El estándar ha venido mejorando a partir de la versión 1.0, que se ratifico en julio de 1999. Se han publicado sucesivas versiones:

a. Versión 1.1: define una velocidad de transmisión de hasta 723.1 Kbps. Soluciona erratas de la especificación 1.0. Añade el Indicador de Calidad de señal Recibida [IV], *Received Signal Strength Indication* (RSSI).

b. Versión 1.2: maneja velocidades de hasta 1 Mbps. Implementa la técnica de salto en frecuencia, *Adaptive Frequency Hopping* (AFH), para mejorar la resistencia a interferencias, proveyendo una solución inalámbrica complementaria que ofrece una transmisión más eficiente y una encriptación más segura para hacer posible la coexistencia de Bluetooth y Wi-Fi en el espectro de los 2.4 GHz. Introduce el tipo de enlace para aplicaciones de audio eSCO, *extended Synchronous Connections*" que mejora la calidad de voz[IV]. Así mismo, Incluye mejoras en el HCI, *Host Controller Interface*, para una sincronización más rápida de las comunicaciones.

c. Versión 2.0: Es una versión compatible con las anteriores. Incorpora la tecnología de Velocidad de Datos Mejorada, *Enhanced Data Rate* (EDR), que incrementa Sistema Bluetooth para el control de dispositivos.

las velocidades de transmisión hasta 3 Mbps. Según la especificación, la potencia de transmisión define tres tipos de productos [IV]:

a. Clase 1: 100 mW (20 dBm), con un alcance aproximado de 100 m.
b. Clase 2: 2.5 mW (4 dBm), con un alcance aproximado de 10 m.
c. Clase 3: 1 mW (0 dBm), con un alcance menor a 1 m.

La interfaz de radio Bluetooth esta basado en una antena nominal de 0dBm, por lo tanto, cada dispositivo puede variar su potencia de transmisión.

La transmisión bidireccional se consigue mediante la técnica de Acceso Múltiple Duplex por División de Tiempo, *Time División Duplex* (TDD). Esta técnica permite transmitir en una dirección a la vez, de esta manera, la transmisión y la recepción ocurren en diferentes slots, evitando las colisiones.

Si más de una piconet usa, en un instante dado, el mismo slot de tiempo existirá una colisión. La perdida de datos se puede corregir usando FEC y detención ARQ.

Se definen dos modos de modulación: Un modo obligatorio, llamado modo de transferencia básica, que usa una modulación GFSK para reducir al mínimo la complejidad del transmisor/receptor. Un modo opcional, llamado de transferencia de datos mejorada (Bluetooth v2.0+EDR), que usa modulación PSK y cuenta con dos variantes:π/4 -DQPSK y 8DPSK. La Modulación por Desplazamiento de Frecuencia Gaussiana, *Gaussian Frequency Shift Keying* (GFSK), define un 1 binario representado con una desviación positiva de frecuencia y un 0 binario con una desviación negativa de la frecuencia [IV].

Bluetooth implementa Expansión del Espectro por Salto de Frecuencia, *Frecuency Hopping Spread Spectrum* (FHSS), con 1600 saltos por segundo entre 79 frecuencias. Este sistema divide la banda de frecuencia en 80 canales con un ancho de banda de 1 MHz para cada canal, durante la conexión van cambiando de un canal a otro de manera pseudo-aleatoria. Con esto se consigue que el ancho de banda instantáneo sea muy pequeño y se tenga una transmisión efectiva sobre el total del ancho de banda Bluetooth que emplea la técnica de Salto de Frecuencias Adaptivo, *Adaptive Frequency-Hopping*5 (AFH), que está basado en FHSS con mejoras en cuanto a interferencia [IV].

El Salto de Frecuencias Adaptivo es muy útil para permitir la coexistencia de Bluetooth con otros sistemas inalámbricos. Este sistema evita las interferencias usando el salto de frecuencias dentro de una parte del espectro que no esté siendo usado. Lo que se busca es diferenciar las frecuencias buenas de las malas. Primero, se define la secuencia de salto normal, y luego se aplica un algoritmo AFH que la modifique para trabajar solamente sobre frecuencias buenas.

La diferenciación se realiza tomando en cuenta varios parámetros como CRC, HEC, FEC, RSSI, nivel de paquetes perdidos, *Packet Loss Ratio* (PLR), por cada canal o técnicas como *Slave's classifications data, Transmissión sensing*, entre otras [IV].

La banda base es la parte del sistema Bluetooth que especifica o introduce los procedimientos de acceso de medios y capa física entre dispositivos Bluetooth.

Sistema Bluetooth para el control de dispositivos.

Usa protocolos combinando conmutación de circuitos y paquetes, apropiados para voz y datos.

Se definen dos tipos de enlaces para soportar aplicaciones de voz y datos:

a. Enlace asíncrono sin conexión, *Asynchronous Connectionless* (ACL): conexiones simétricas o asimétricas punto-multipunto entre maestro y esclavo [IV]. Esta conexión es utilizada para el tráfico de datos, sin garantía de entrega, y se retransmiten paquetes. La máxima velocidad de envío es de 721 Kbps en una dirección y de 57.6 Kbps en la otra [V].

b. Enlace síncrono orientado a conexión, *Synchronous Connection-Oriented* (SCO): conexiones simétricas punto a punto entre maestro y esclavo[V]. Esta conexión es capaz de soportar voz en tiempo real y tráfico multimedia. Su velocidad de transmisión es de 64 KB/s. Reduce el consumo de energía a pesar del incremento de velocidad.

Los dispositivos electrónicos equipados con tecnología Bluetooth pueden conectarse y comunicarse de forma inalámbrica mediante redes ad-hoc de corto alcance denominadas Piconets. Un maestro puede tener hasta 256 esclavos, pero solamente puede conectarse simultáneamente con hasta siete dentro de una misma piconet. Así mismo, un esclavo puede pertenecer a varias piconets al mismo tiempo, pero estar activa solo en una piconet a la vez. Las piconets se establecen de forma dinámica y automática cuando los dispositivos Bluetooth se encuentran en el mismo radio de acción.

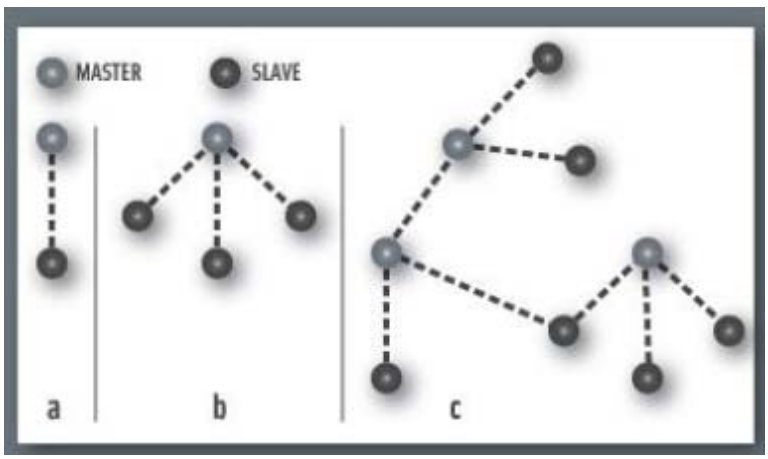


Figura 1.3 Tipos de Piconets

En la Figura 1.3 se observan ejemplos de piconets,

(a) Piconet con un solo esclavo.
(b) Configuración multipunto con tres esclavos.
(c) Scatternet, es decir, un dispositivo que es maestro en una piconet puede ser esclavo en otra.

1.3.2 VENTAJAS DE BLUETOOTH

Es fácil reconocer las ventajas sobre otros tipos de comunicación inalámbrica fijándose en la descripción técnica anterior. Además, los objetivos que la tecnología Bluetooth persigue son los que permitieron crear expectativas en la factibilidad de este proyecto, algunos de estos se resumen a continuación.

a. Bluetooth es una tecnología disponible mundialmente

La especificación Bluetooth esta disponible de forma gratuita para empresas afiliadas de todo el mundo. Fabricantes de distintos sectores industriales están implantando esta tecnología en sus productos para reducir el número de cables y lograr conexiones sin interrupciones, transmitir sonido estéreo, transferir datos o para establecer comunicaciones de voz.

Como ya se ha mencionado, la tecnología Bluetooth funciona en la banda de 2.4 GHz, una de las bandas de radio industrial, científica y médica (ISM) que no requiere licencia. Por tanto, no existen gastos asociados al uso de la tecnología Bluetooth[VII], excepto aquéllos directamente vinculados al dispositivo.

Desde la publicación de la primera especificación Bluetooth en 1999, más de 4.000 empresas se han afiliado al Grupo de Interés Especial, *Special Interest Group* (SIG), de Bluetooth, y el número de productos Bluetooth disponibles en el mercado se multiplica con rapidez.

b. Bluetooth funciona en una amplia gama de dispositivos

La tecnología Bluetooth está disponible en todo un abanico de dispositivos, desde teléfonos móviles hasta instrumental médico, pasando por automóviles, electrodomésticos, etc. Abarca una gran variedad de usuarios, desde consumidores particulares hasta mercados industriales. Su bajo consumo de energía, reducido tamaño y el escaso costo de los chips permite emplear la tecnología Bluetooth hasta en los dispositivos más pequeños.

c. Bluetooth es fácil de usar

El estándar Bluetooth es una tecnología que no necesita una infraestructura fija y es sencilla de instalar y configurar. La conexión se realiza sin cables. El proceso resulta muy sencillo para los nuevos usuarios: una vez adquirido el producto Bluetooth, basta con comprobar los perfiles disponibles y conectarlo a otro dispositivo Bluetooth con los mismos perfiles. El usuario lleva consigo en todo momento su red de área personal (PAN) e incluso puede conectarse a otras.

d. El emisor de radio Bluetooth consume poca energía y puede integrarse a equipos alimentados por baterías.

La tecnología inalámbrica Bluetooth exhibe, entre sus características, poco consumo de energía por lo que se puede usar baterías para su funcionamiento. Por ejemplo,

el módulo KC-21 tiene un consumo aproximado de 16.5 mA en modo Standby y hasta 35 mA en modo de conexión a la máxima velocidad.

e. Bluetooth es un sistema basado en un protocolo robusto y seguro

Actualmente, la información ya no discurre por cables, y debe ser enviada a su destinatario de forma segura, sin ser interceptada [IV] Los estándares que rigen las comunicaciones inalámbricas están evolucionando y presentan varios formatos para garantizar seguridad a los usuarios.

Desde su creación, Bluetooth se ha esforzado para que los usuarios de este estándar global puedan sentirse plenamente seguros al conectarse. Cuenta con un grupo de expertos en seguridad formado por ingenieros de las empresas afiliadas. Este grupo suministra información decisiva sobre cuestiones de seguridad y sus consejos se tienen en cuenta en el proceso de desarrollo de las especificaciones inalámbricas de la tecnología Bluetooth.

Al estar disponible en todo el mundo a través de la banda ISM abierta de 2.4 GHz, la fiabilidad fue prioritaria desde un primer momento. Además, gracias a la función de salto adaptivo de frecuencia (AFH) se limitan las interferencias de otras señales.

Bluetooth cuenta con seguridad integrada, como el cifrado de 128 bits y la autenticación mediante código PIN. El número de identificación personal (PIN) es un código alfanumérico compuesto de cuatro o más caracteres que se asocia temporalmente a un producto para realizar un emparejamiento.

1.3.3 COMPARACIÓN CON OTRAS TECNOLOGÍAS

1.3.3.1 Bluetooth e Infrarrojo

Las especificaciones que constituyen el estándar internacional para el desarrollo de sistemas de comunicaciones a través de rayos infrarrojos adopta el nombre de IrDA, *Infrared Data Association*. IrDA Define comunicaciones bidireccionales punto a punto empleando un haz de luz infrarroja que requiere línea de vista, un ángulo menor a 30 grados y una distancia que no excede un metro para obtener tasas de transmisión de datos entre 9.6Kbps y 16Mbps dependiendo del entorno[VII]

La implementación de estos sistemas de modulación pueden ser más compleja y costosa que el sistema de modulación GFSK usada por Bluetooth.

Para velocidades de hasta 1.152Mbps, se usa la modulación RZI, *Return to Zero Inverted*. Para velocidades de 4 Mbps, se usa modulación 4PPM, *Pulse Position Modulation.*

Los dispositivos que se encuentran comunicándose deben estar cara a cara dentro de un margen de más o menos un metro de distancia para realizar un intercambio de información que los involucra exclusivamente a ellos.

Se maneja un sistema maestro - esclavo, que según la IrDA se define como estación primaria - estación secundaria. La estación primaria es la encargada de enviar los Sistema Bluetooth para el control de dispositivos.

comandos de inicio de conexión y de transferencia. Además, garantiza el flujo organizado y controlado de los datos, así como el tratamiento de los errores en la transmisión IrDA, y define comunicación Half-Duplex[VII].

La transmisión de señales infrarrojas es direccional dentro de un ángulo sólido medio de 15 grados, con el objetivo de minimizar las interferencias con dispositivos que se encuentran cercanos.

Además de los otros dispositivos alrededor de los dos que participan en una comunicación, la transmisión es sensible a las componentes infrarrojas contenidas en la luz solar o en luces fluorescentes.

El diseño del hardware en IrDA es tal, que no pueden detectarse las colisiones. Axial, que es el software empleado para cada aplicación, es quien debe realizar el control de estos inconvenientes.

Debido a la similitud de aplicaciones, se considera importante delimitar las ventajas entre una y otra tecnología. Por ejemplo:

a. El infrarrojo requiere de una comunicación lineal entre transmisor y receptor, lo que hace imprescindible la línea de vista para su efectiva transmisión. Esto puede reducir la interferencia si se usa a muy cortas distancias, lo cual se traduce en una desventaja.

b. Las frecuencias de la banda del infrarrojo no permiten la penetración a través de paredes, dándole una importante ventaja a la radiofrecuencia que opera Bluetooth.

c. La comunicación con IrDA siempre será uno a uno. Esta podría ser una ventaja desde el punto de vista de la seguridad, ya que el usuario determina directamente con quien se está comunicando y el tiempo de negociación de la comunicación puede ser menor, mientras con Bluetooth es necesaria una etapa previa de búsqueda y emparejamiento. La gran ventaja de Bluetooth es la posibilidad de manejar aplicaciones multipunto.

d. Bluetooth permite la generación de redes.

e. La velocidad de transmisión de datos es más elevada en la tecnología IrDA, alcanzando hasta 16Mbps, compensando así la corta distancia y el requisito de línea de vista. Bluetooth puede alcanzar un máximo 3Mbps a distancias de hasta 100m[VII].

f. Bluetooth permite movilidad dentro del área de cobertura. Para IrDA ofrecer desplazamiento es muy difícil ya que el haz de luz admitido debe ser menor a 30 grados dentro de un metro de radio.

g. La interferencia es muy importante para ambas tecnologías, mientras IrDA está afectado por la luz fluorescente y solar. Bluetooth se preocupa de no estar afectado por otros sistemas que también usan la frecuencia de 2.4GHz.

Tanto Bluetooth como IrDA, especifican una comunicación inalámbrica a cortas distancias, sin embargo Bluetooth podría sustituir algunas aplicaciones de infrarrojo por las claras ventajas que provee.

1.3.3.2 Bluetooth y WiFi

La norma del IEEE (*Institute of Electrical and Electronic Engineers*) 802.11 representa el primer estándar (aparece en 1990) para productos WLAN de una organización independiente reconocida a nivel internacional, que además ha definido las principales normas en redes LAN cableadas[IV].

En el año 1999, se aprobó el estándar 802.11b, una extensión del IEEE 802.11 para WLAN empresariales, con una velocidad de 11 Mbps y un alcance de 100 metros, que al igual que Bluetooth y Home RF, también emplea la banda de ISM de 2,4 GHz, pero en lugar de una simple modulación de radio digital y salto de frecuencia (FH), utiliza técnicas de espectro expandido por secuencia directa (DSSS).

En julio de 1999 la IEEE publicó el suplemento del estándar en 802.11a, que usa Modulación Ortogonal por División de Frecuencia, *Orthogonal Frequency División Multiplexing* (OFDM) y alcanza una velocidad de hasta 54 Mbps en la banda de 5 GHz.

En el año 2003, se aprobó el estándar IEEE 802.11g, compatible con el IEEE 802.11b, capaz de alcanzar una velocidad de 54 Mbps[IV].

El IEEE 802.11 define tres tipos de modulación en la capa física para la transmisión y recepción de tramas:

a. Espectro expandido por secuencia directa DSSS
b. Espectro expandido por salto de frecuencias FHSS
c. Luz Infrarroja en banda base sin modular

Además, define señalización, características de la transmisión de datos, topologías, tipos de medios inalámbricos, posibles velocidades, potencia y banda de frecuencia. Las topologías definidas son Ad-hoc y de Infraestructura.

WiFi usa Acceso Múltiple por detección de Portadora con Evasión de Colisiones, *Carrier Sense Múltiple Access, Collisión Avoidance* (CSMA/CA), antes de transmitir información, la estación escucha el medio, para determinar su estado. Si el medio esta desocupado la estación espera un tiempo determinado por algoritmos de backoff y luego transmite, de esta manera evita las colisiones[VII].

WiFi usa sistemas de seguridad como:

a. WEP ofrece dos niveles de seguridad, encriptación a 64 o 128 bit.
La encriptación usa un sistema de claves. La clave de la tarjeta de red del cliente debe coincidir con la clave del AP.

b. WPA emplea cifrado de clave dinámica, es decir, la clave está cambiando constantemente y hacen que las incursiones en la red inalámbrica sean más difíciles que con WEP. Hoy en DIA Wi-Fi amplia más sus aplicaciones gracias a que cada vez son más los dispositivos móviles que salen al mercado (ordenadores portátiles, agendas electrónicas, teléfonos). Las redes inalámbricas son la mejor solución para conectarlos a la red, aprovechando al máximo todas sus ventajas, sin perder la flexibilidad y movilidad que proporciona no necesitar cables.

Las principales aplicaciones para este estándar son: conexión a Internet, Telefonía IP, Redes privadas, CCTV (Circuito Cerrado de Televisión), Gestión de datos, etc.

Una red inalámbrica WiFi necesariamente requiere de una configuración tanto en hardware como en software. Un dispositivo Bluetooth se comunica sin la necesidad de configurar hardware o drivers.

Bluetooth se puede utilizar sin haber instalado una infraestructura de red previamente y aunque las opciones de control de dispositivos aumentan al usar 802.11b, el costo también es mucho más alto y también el consumo de energía.

Los dispositivos Bluetooth también puede acceder a los recursos de una LAN, para esto, el punto de acceso de LAN debe proporcionar las capacidades de la conversión protocolares. En resumen, ambas tecnologías podrían coexistir en un mismo ambiente llegándose a complementar.

1.3.3.3 Bluetooth y ZigBee

ZigBee es una alianza, sin animo de lucro, de unas 25 empresas, la mayoría de ellas fabricantes de semiconductores, con el objetivo de auspiciar el desarrollo e implantación de una tecnología inalámbrica de bajo costo como un sistema estándar de comunicaciones, vía radio y bidireccional, para usarlo dentro de dispositivos de domótica, automatización de edificios (inmótica), control industrial, periféricos de PC y sensores médicos.

En ZigBee una red simple puede soporta hasta 256 dispositivos y una red extendida puede contener hasta 256 subredes. En tanto que Bluetooth soporta un máximo de 7 dispositivos por red y una red extendida puede soportar hasta 8 subredes. En este punto ZigBee llega a ser totalmente superior a Bluetooth.

Las soluciones sobre el estándar ZigBee en conexión de redes, se centran en aplicaciones especificas de sondeo y monitoreo, por ejemplo: automatización de edificios y hogares, equipos para la salud y monitoreo de signos vitales, control industrial, electrónica de consumo, PC y periféricos, control comercial y de iluminación, etc. Mientras Bluetooth tiene aplicaciones como transferencia de archivos, escritorio Inalámbrico, conexión a Internet, acceso Inalámbrico a LAN, transmisión de audio y video, control remoto de dispositivos, etc.

Bluetooth y ZigBee tienen varios campos comunes de aplicación principalmente en la electrónica de consumo sin cables (PC – periféricos) pero también tienen marcadas diferencias en la forma de transmisión de datos.

Sistema Bluetooth para el control de dispositivos.

La IEEE 802.15.4 ofrece tres bandas de frecuencia en las cuales operar: 2.4 GHz, 915MHz y 868 MHz. La frecuencia de 2.4 GHz, especifica la operación en la banda Industrial, Medica y Científica (ISM), que prácticamente esta disponible en todo el mundo, mientras que la frecuencia de 865 MHz opera en Europa y 915 MHz en Estados Unidos[VII].

El estándar IEEE 802.15.4 utiliza la técnica de Espectro Ensanchado por Secuencia Directa, *Direct Secuence Spread Spectrum* (DSSS) para transmitir la información a través del medio. Bluetooth emplea el método FHSS el cual es más seguro que DSSS empleado por ZigBee, debido a que los saltos de frecuencia realizados por la señal son de manera aleatoria.

Las velocidades de transmisión son de 250 Kbps en la banda de 2.4 GHz, 40 Kbps en la banda de 915 MHz y 20 Kbps en la banda de 868 MHz. Bluetooth puede llegar a velocidades de hasta 3Mbps.

Para las frecuencias de 915 MHz y 868 MHz la señal es modulada con BPSK, *Binary Phase Shift Keying*. Mientras que a la frecuencia 2.4 GHz se emplea una técnica de modulación O-QPSK. IEEE 802.15.4 define 27 canales diferentes; por lo tanto, la capa física debe sintonizar al dispositivo dentro del canal a utilizarse. La banda de 2.4 GHz soporta 16 canales entre 2.4 y los 2.4835 GHz, con un espacio entre canales de 5 MHz.

El estándar 802.15.4 define CSMA/CA. El dispositivo escucha el canal, si éste está ocupado, entonces el algoritmo CSMA/CA le asigna un número de periodos de backoff que deberá esperar antes de sondear nuevamente el canal.

Una red ZigBee puede constar de un máximo de 65535 nodos distribuidos en subredes de 255 nodos, frente a los 8 máximos de una subred Bluetooth.

ZigBee define múltiples topologías, como se puede observar en la Figura 1.4:
(a) Topología en estrella, donde se tiene un único nodo trabajando como coordinador PAN;
(b) Topología mesh, donde se consigue conectividad total de todos los FFDs que conforman la red con el FFD que actúa como coordinador PAN.
(c) Topología cluster tree, que es la asociación de varias redes;

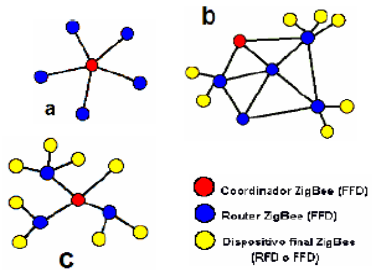


Figura 1.4 topologías ZigBee

Con respecto a la sensibilidad, el IEEE 802.15.4 especifica –85 dBm para la frecuencia de 2.4 GHz y de -92 dBm para las frecuencias de 868 y 915 MHz. El estándar IEEE 802.15.4 especifica que cada dispositivo debe de ser capaz de transmitir al menos a 1 mW, pero dependiendo de las necesidades de la aplicación, la potencia de transmisión

Sistema Bluetooth para el control de dispositivos.

puede variar. Los dispositivos típicos (1mW) se espera que cubran un rango de entre 10 y 75 m; sin embargo, con una buena sensibilidad y un incremento moderado en la potencia de transmisión, se obtiene mayores coberturas. Bluetooth maneja potencias algo más elevadas, pero puede alcanzar hasta 100m[VII].

ZigBee tiene algunas ventajas en comparación a Bluetooth, pero no está comercializada y generalizada en todo el mundo por lo que usarla todavía resulta más costoso.

Mientras que el Bluetooth se usa para aplicaciones como los teléfonos móviles y la informática casera, la velocidad de ZigBee se hace insuficiente para estas tareas, desviándolo a usos tales como la domótica, los productos dependientes de la batería, los sensores médicos, y en artículos de juguetería, en los cuales la transferencia de datos es menor.

1.3.3.4 Otras tecnologías

Al igual que ZigBee, existen más estándares que todavía no se terminan de implementar o dar a conocer mundialmente. axial por ejemplo:

Home RF, que es una iniciativa de varias empresas que se unieron en el año 1998 para crear una tecnología de transmisión digital inalámbrica abierta para que computadores, impresoras, teléfonos, modems y cualquier otro dispositivo digital pudiera intercambiar datos sin necesidad de usar cables. Se basa en un modelo equivalente al estándar de los teléfonos móviles GSM. Transporta voz y datos por separado.

UPnP (Universal Plug&Play, UPnP), que es una arquitectura abierta y distribuida que permite a las aplicaciones de los dispositivos conectados a una red intercambiar información y datos de forma sencilla y transparente para el usuario final, sin necesidad de que este tenga que ser un experto en la configuración de redes, dispositivos o sistemas operativos.

X-10, que fue diseñado en Escocia entre los anos 1976 y 1978 con el objetivo de transmitir datos por las líneas de baja tensión a muy baja velocidad (60 bps en EEUU y 50 bps en Europa) y costos muy bajos usando las líneas de eléctricas de la vivienda). Utiliza la línea eléctrica (220V o 110V) para transmitir señales de control entre equipos de automatización del hogar en formato digital. Las señales de control de X10 se basan en la transmisión de ráfagas de pulsos de RF (120 kHz) que representan información digital. Estos pulsos se sincronizan en el cruce por cero de la señal de red (50 Hz o 60 Hz). Con la presencia de un pulso en un semiciclo y la ausencia del mismo en el semiciclo siguiente se representa un '1' lógico y a la inversa se representa un '0'. A su vez, cada orden se transmite 2 veces. Cada orden involucra 11 ciclos de red (220 ms para 50 Hz y 183,33, para 60Hz). Hay 256 direcciones soportadas por el protocolo. Fue creado principalmente para el control remoto de dispositivos eléctricos[VII).

Jini, una tecnología desarrollada por Sun Microsystems, que proporciona un mecanismo sencillo para que diversos dispositivos conectados a una red puedan colaborar y compartir recursos sin necesidad de que el usuario final tenga que planificar y configurar dicha red.

Sistema Bluetooth para el control de dispositivos.

Axial como las anteriores, existen muchas otras que tienen ventajas y desventajas. Sin embargo, para aplicaciones sencillas como la del presente proyecto, Bluetooth sigue siendo la mejor opción.

En conclusión, la utilización de una tecnología que está al alcance de todos e implementada en la mayoría de dispositivos comercializados actualmente puede generar un mejor aprovechamiento de la misma.

CAPÍTULO 2. DESCRIPCIÓN DE JAVA 2 MICRO EDITION

Actualmente, existe una infinidad de lenguajes de programación que son usados para crear diversos sistemas encaminados a implementar Tecnologías de la Información y Comunicaciones. Los lenguajes de mayor demanda son los Lenguajes Orientados a Objetos.

Al igual que Bluetooth, Java es un estándar incluido en la mayoría de teléfonos móviles. Éste es el principal motivo por el que se consideró a este lenguaje el más adecuado para llevar a cabo el proyecto.

Se creará una aplicación Java para un teléfono móvil que permita realizar una conexión Bluetooth con el módulo de control y sobre la cual se envían los comandos que permiten controlar los dispositivos eléctricos conectados a éste.

El lenguaje Java es reconocido mundialmente por su capacidad de adaptación a cualquier ambiente. Los desarrolladores de Java optaron por dotar al lenguaje de un conjunto de capacidades orientadas a objetos que permitiesen añadir librerías de apoyo a medida que fuera necesario y en función de las necesidades tecnológicas.

Este proyecto se enfoca en teléfonos móviles que sin importar la marca son compatibles con aplicaciones Java. Las características concretas de este tipo de dispositivos han obligado a los desarrolladores de Java a construir un subconjunto del lenguaje y a reconfigurar sus principales bibliotecas para permitir su adaptación a un entorno con poca capacidad de memoria, poca velocidad de proceso y pantallas de reducidas dimensiones. Para esto se creó una nueva plataforma de desarrollo y ejecución sobre la que se centra este proyecto: Java ME, conocida inicialmente como *Java 2 Platform Micro Edition*.

J2ME (Java ME) es una plataforma para pequeños dispositivos que permite la ejecución de programas creados en un entorno Java. Actualmente, los teléfonos móviles se manejan sobre Sistemas Operativos completos, entre los más conocidos está Symbian, usado por los teléfonos Nokia, Sony-Ericsson, Motorola, entre otros; de la misma manera, gran cantidad de móviles cuentan con la tecnología Bluetooth e incluyen un entorno Java, es decir, pueden usar aplicaciones Java[2].

En este capítulo se revisarán las características del lenguaje Java, principalmente de la plataforma Java ME, que será usada para crear una aplicación que maneje tecnología Bluetooth y pueda ser usada por un teléfono móvil compatible con Java ME.

2.1 DESCRIPCIÓN DEL LENGUAJE JAVA

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

La plataforma Java ha experimentado numerosos cambios desde las primeras versiones. En diciembre de 1998, Sun Microsystems lanza la nueva versión de Java, conocida como JAVA 2 PLATFORM[2].

Dentro de la Plataforma Java 2 se pueden encontrar tres diferentes entornos para desarrollo y ejecución de aplicaciones: Java SE, Java EE y Java ME, como se puede ver en la Figura 2.1.

La decisión de separarlos fue debida a que no todas las características de la Plataforma Java son necesarias para el desarrollo de aplicaciones.

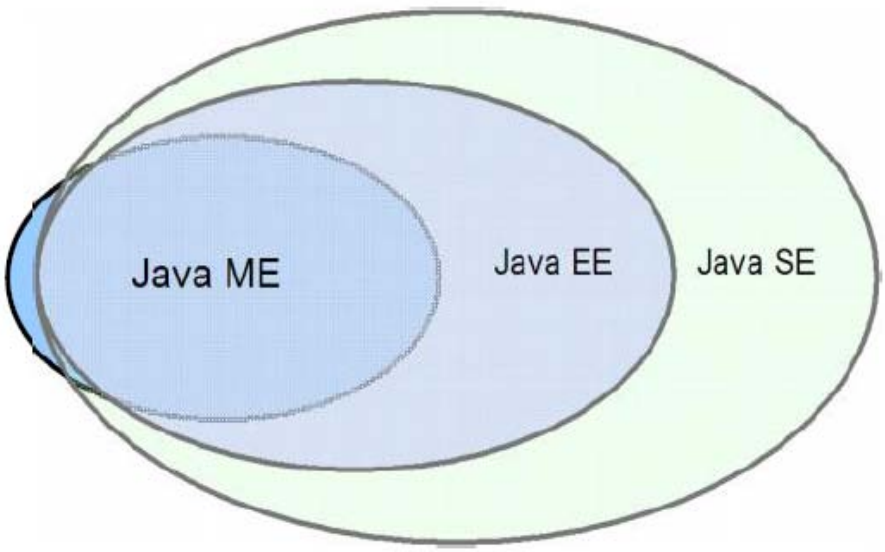


Figura 2.1 Diagrama de Entornos de la Plataforma Java

Los entornos específicos que están definidos para la Plataforma Java son los que se resumen a continuación:

a. Java SE (J2SE), Java Standard Edition, por decirlo de alguna manera, la base de la tecnología Java permite el desarrollo de applets (aplicaciones que se ejecutan en un navegador web) y aplicaciones independientes (stand-alone). Java SE es el heredero directo del Java original[7].

b. Java EE (J2EE), Java Enterprise Edition, está basado en Java SE, pero añade una serie de características necesarias en entornos empresariales, relativos a redes, acceso a

Sistema Bluetooth para el control de dispositivos.

datos y entrada/salida que requieren mayor capacidad de proceso, almacenamiento y memoria.

c. Java ME (J2ME), Java 2 Micro Edition, es un Subconjunto de Java SE para dispositivos con recursos más limitados. Este entorno es el más adecuado para realizar aplicaciones en teléfonos, por tal razón, serán objeto de una revisión más profunda[2].

Cada entorno usa las librerías apropiadas para su área de desarrollo, estas librerías vienen agrupadas en lo que se conoce como APIs.

2.1.1 APLICATION PROGRAMMING INTERFACES, APIS

Un API Java es una Interfaz de Programación de Aplicaciones provista por los creadores del lenguaje Java, y que da a los programadores los medios para desarrollar aplicaciones Java.

A través de un procedimiento formal, los expertos de la industria desde un concepto general pueden dar forma futura al estándar. Este proceso se conoce como JCP, Java Community Process y conlleva el uso de documentos oficiales que describen las especificaciones y tecnologías propuestas para añadirse en la plataforma Java, estos documentos se conocen como *Java Specification Request* (JSR).

Existen más de 300 JSR en general y más de 80 para JavaME, algunos ejemplos de ellos se pueden apreciar en la Tabla 2.1 [I]

JSR	DESCRIPCIÓN
30	Java METM Connected, Limited Device Configuration
	Esta especificación definirá un estándar para la configuración de JavaTM 2 Platform Micro Edition (Java METM) para dispositivos pequeños con recursos limitados con conexión (CLDC)
36	Connected Device Configuration
	La configuración de dispositivos con conexión (CDC) provee las bases de JavaTM 2 platform Micro Edition para dispositivos que tienen un procesador mínimo de 32-bits y amplia memoria.
37	Mobile Information Device Profile for the Java METM Platform
	Esta especificación definirá un perfil que extenderá y ampliará el Java METM Connected, Limited Device Configuration" (JSR-000030), habilitando el desarrollo de aplicaciones para aparatos de información móviles y dispositivos de comunicación por voz.
46	Foundation Profile
	El Foundation Profile es un conjunto de APIs pensando para aplicaciones corriendo en dispositivos pequeños que tienen algún tipo de conexión a red.
62	Personal Profile Specification
	El Java METM Personal Profile provee el ambiente Java ME para aquellos dispositivos que tengan una necesidad de contar con un alto grado de conectividad a Internet y fidelidad Web.

Sistema Bluetooth para el control de dispositivos.

68	Java METM Platform Specification
	Esta especificación definirá la siguiente mejor revisión del JavaTM 2 Platform, Micro Edition.
75	PDA Optional Packages for the Java METM Platform
	Este JSR produce dos paquetes opcionales separados por características comúnmente encontradas en PDAs y otros dispositivos móviles Java ME: uno para acceder a datos PIM y otro para acceder a sistemas de ficheros.
82	JavaTM APIs for Bluetooth
	Bluetooth es un estándar importante que está emergiendo para la integración gíreles de dispositivos pequeños. La especificación estandariza un conjunto de APIs de Java, para los dispositivos que soportan Java, que permiten la integración a un ambiente Bluetooth
118	Mobile Information Device Profile 2.0
	Esta especificación definirá un perfil que ampliará y extenderá el "Java METM Mobile Information Device Profile" (JSR-000037).
134	JavaTM Game Profile
	Define un perfil Java 2 Micro Edition con fines de desarrollo de juegos dirigidos a los consumidores de dispositivos de juegos computadores de escritorio.
135	Mobile Media API
	Especifica un API multimedia para Java METM, para el sencillo y fácil acceso para el control básico de audio y a los recursos multimedia, al mismo tiempo que hace frente a la escalabilidad y el apoyo de características más sofisticadas.
139	Connected Limited Device Configuration 1.1
	Esta especificación definirá una versión revisada del Java METM Connected, Limited Device Configuration (CLDC).
205	Wireless Messaging API 2.0
	Este JSR extenderá y ampliará el API para mensajes wireless (JSR-000120)
216	Personal Profile 1.1
	Este JSR actualizará la especificación del actual Personal Profile (JSR-62) para reflejar el API J2SETM 1.4.
218	Connected Device Configuration (CDC) 1.1
	Este JSR define una revisión de la especificación Java ME CDC. Este JSR provee actualizaciones para el núcleo existente basadas en el J2SE, v1.4, no gráficas, para dispositivos pequeños.
246	Device Management API
	Habilita a las aplicaciones Java METM el acceso a dispositivos de gestión de implementaciones.
927	Java TVTM API 1.1
	El mantenimiento de la especificación Java TVTM.

Tabla 2.1 Descripción de algunos JSR

Los teléfonos móviles tienen incluidos algunos de estos APIS. Para ser usados en este proyecto es importante que cuenten principalmente con los siguientes: JSR82 / Bluetooth; JSR118 / *Mobile Information Device Profile* 2.0 y JSR139 / *Connected Limited Device Configuration* 1.1, que se analizarán más adelante.

2.2 PLATAFORMA JAVA MICRO EDITION (Java ME)

La elección de un entorno Java específico depende del tipo de aplicación y del tipo de dispositivo en el cual se ejecutará. Java Platform, Micro Edition (Java ME) es una colección de tecnologías y especificaciones para crear una plataforma que se ajuste a las necesidades de dispositivos móviles y que se pueden combinar para crear un completo entorno de ejecución de Java.

El entorno de ejecución estará compuesto por: una máquina virtual, una configuración, un perfil y otros paquetes opcionales[7].

En la Figura 2.2 se pueden observar la Plataforma Java y sus diferentes entornos, incluyendo los componentes del entorno Java ME.

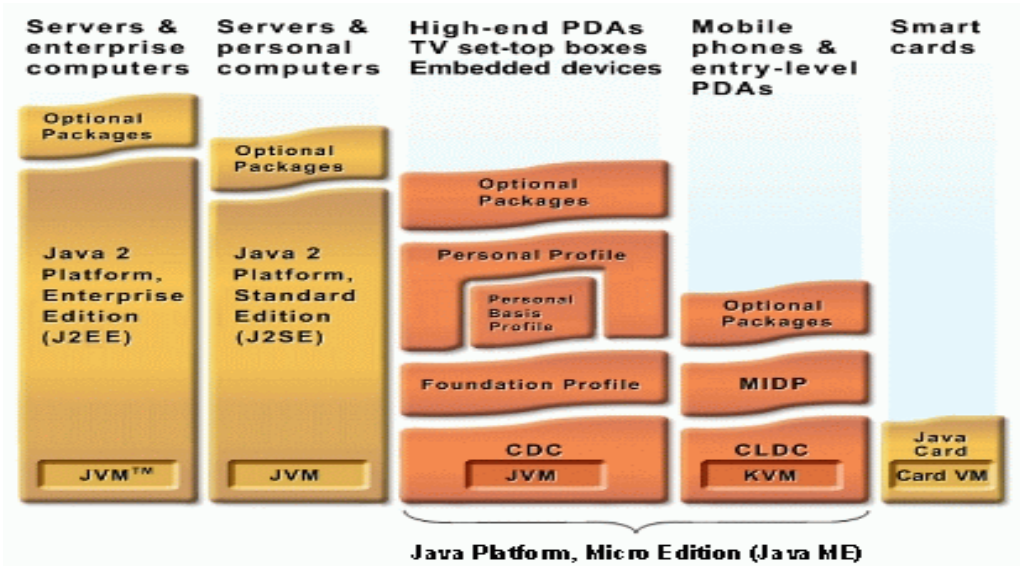


Figura 2.2 Plataforma Java y sus componentes.

2.2.1 MÁQUINAS VIRTUALES PARA JAVA ME

"Una máquina virtual de Java (JVM) es un programa encargado de interpretar código intermedio (bytecode) de los programas Java precompilados a código de máquina ejecutable por la plataforma, efectuar las llamadas pertinentes al sistema operativo subyacente y observar las reglas de seguridad y corrección de código definidas para el lenguaje Java". La JVM permite prescindir de un sistema operativo específico en cada equipo y hace que la aplicación sea independiente del mismo. Dependiendo de las capacidades del equipo, la Máquina Virtual de Java ME puede ser de varios tipos, entre los más usados se tiene:

Sistema Bluetooth para el control de dispositivos.

a. KVM, corresponde a la máquina virtual más pequeña desarrollada por Sun. Su nombre proviene de kilobytes ya que ocupa entre 40 y 80 kilobytes de memoria. Está orientada a dispositivos con baja capacidad computacional y de memoria. Ésta es la que se usa generalmente con dispositivos móviles como teléfonos móviles.

b. CVM, corresponde a la máquina virtual con algunas características adicionales a la KVM, CVM significa Compact Virtual Machine. Está orientada a dispositivos electrónicos con procesadores de 32 bits de gama alta y con alrededor de 2Mb o más de memoria RAM. Soporta las mismas características que J2SE[7].

2.2.2 CONFIGURACIONES JAVA ME

Una configuración describe las características mínimas, en cuanto a la configuración de hardware y software, usando un conjunto de APIs (Application Programming Interfaces) Java que permiten desarrollar aplicaciones para un grupo de dispositivos. Éstas APIs describen las características básicas, comunes a todos los dispositivos[7]:

a. Características soportadas por lenguaje de programación Java.
b. Características soportadas por la Máquina Virtual Java.
c. Bibliotecas básicas de Java soportadas.

Existen dos configuraciones dentro de Java ME:

a. CDC, orientada a dispositivos con menos limitaciones.
b. CLDC, orientada a dispositivos con limitaciones computacionales y de memoria considerables.

2.2.2.1 Configuración de Dispositivos con Conexión CDC

La CDC funciona en dispositivos con algunas capacidades de memoria y procesamiento, así como: equipos de televisión por Internet, decodificadores de TV digital, electrodomésticos, sistemas de navegación, etc. CDC usa una Máquina Virtual Java (CVM) con características similares a la JVM de J2SE, tomando en cuenta las limitaciones en la interfaz gráfica y en memoria del equipo[7].

La configuración CDC define características incluidas por el lenguaje Java, las funcionalidades de la máquina virtual Java (CVM), las mismas APIs usadas por CDLC necesarias para el desarrollo de aplicaciones, un subconjunto de las librerías de J2SE, los requerimientos de hardware de los dispositivos, E/S, acceso a redes, seguridad, etc[7].

2.2.2.2 Configuración de Dispositivos Limitados con Conexión CLDC

CLDC es una especificación general para un amplio abanico de dispositivos, que van desde teléfonos móviles hasta PDAs y otros, que se caracterizan por sus limitaciones en cuanto a capacidad gráfica, cómputo y memoria. Algunas de estas restricciones vienen

Sistema Bluetooth para el control de dispositivos.

dadas por el uso de la KVM, necesaria al trabajar con la CLDC debido a su pequeño tamaño[7].

La configuración CLDC define características incluidas por el lenguaje Java, las funcionalidades de la máquina virtual Java, las APIs (Application Programming Interfaces) necesarias para el desarrollo de aplicaciones y un subconjunto muy pequeño de las librerías de J2SE.

Los dispositivos que usan CLDC se caracterizan por:

a. Disponer como mínimo de 128 Kb de memoria para la Máquina Virtual

b. Tener un procesador de 16 o 32 bits con al menos 25 MHz de velocidad.

c. Ofrecer bajo consumo, debido a que estos dispositivos trabajan con suministro de energía limitado, normalmente baterías.

d. Tener conexión a algún tipo de red, normalmente sin cable, con conexión intermitente y ancho de banda limitado (unos 9600 bps) [7].

2.2.2.2.1 Librerías incluidas en CLDC

Las librerías en general entregan al programador un conjunto de funciones para realizar tareas comunes, manejar elementos y objetos, funciones y accesos. El numero de librerías incluidas en CLDC es menor que en CDC, esto debido al tipo de dispositivos que usan esta aplicación y su alcance.

paquete CLDC	Descripción
java.lang	Clases e interfaces de la Máquina Virtual. Subconjunto de J2SE.
java.util	Clases, interfaces y utilidades estándar. Subconjunto de J2SE.
javax.microedition.io	Clases e interfaces de conexión genérica CLDC
java.io	Clases y paquetes estándar de E/S. Subconjunto de J2SE.

Tabla 2.2 Librerías incluidas en CLDC

2.2.3 PERFILES DE JAVA ME

Un perfil especifica las características de un dispositivo según su tipo, tomando en cuenta su uso y las aplicaciones que se pueden ejecutar en estos, por ejemplo: teléfonos móviles, agendas electrónicas, electrodomésticos, audífonos y manos libres, etc. La parte gráfica es un elemento importante a definir en un perfil, no es lo mismo la interfaz gráfico para un teléfono móvil que una interfaz táctil, por ejemplo.

El Perfil especifica estas características mediante APIs, mientras que la Configuración hace algo similar a nivel de grupos de dispositivos. Así, un perfil siempre debe estar construido sobre una configuración determinada. En la Figura 2.3 se encuentra un esquema de la arquitectura del entorno Java ME[7]..

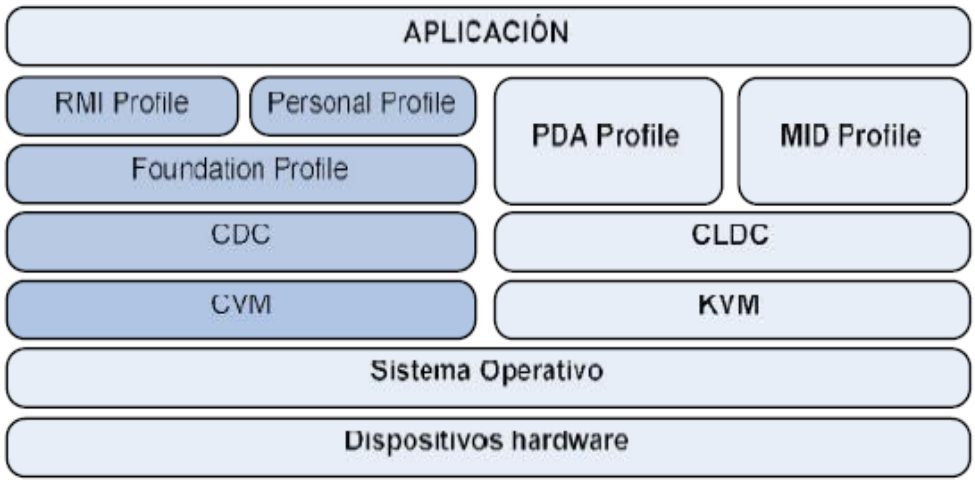


Figura 2.3 Arquitectura del entorno de ejecución de Java ME

En resumen, un Perfil no es nada más que un conjunto de APIs que le dan a una configuración un carácter más específico.

Para la configuración CLDC se tienen los siguientes perfiles:

a. *PDA Profile* (PDAP): Está construido sobre CLDC. Pretende abarcar PDAs de gama baja, tipo Palm, con una pantalla y algún tipo de puntero.

b. *Mobile Information Device Profile* (MIDP): Este perfil está construido sobre la configuración CLDC y define un entorno Java para teléfonos móviles, buscapersonas y dispositivos similares con recursos limitados[7]..

Los dispositivos que usan MIDP deben tener las siguientes características:

a. Suficiente memoria para correr las aplicaciones.

b. Una pantalla de alrededor de 96 por 56 píxeles.

c. Un teclado o una pantalla táctil.

d. Capacidad de conexión inalámbrica.

El perfil MIDP da a la aplicación la capacidad de controlar la interfaz de usuario, el almacenamiento, las conexiones de red, etc.

Paquetes del MIDP	Descripción
javax.microedition.lcdui	Clases e interfaces gráficas para el usuario
javax.microedition.rms	Record Management Storage. Soporte para el almacenamiento persistente del dispositivo
javax.microedition.midlet	Clases de definición de la aplicación
javax.microedition.io	Clases e interfaces de conexión genérica
java.io	Clases e interfaces de E/S básica
java.lang	Clases e interfaces de la Máquina Virtual
java.util	Clases e interfaces de utilidades estándar

Tabla 2.3 Paquetes que están incluidos en el perfil MIDP[7].

Sistema Bluetooth para el control de dispositivos.

2.2.4 PAQUETES OPCIONALES

Los paquetes opcionales son APIs, similares a los que tienen incluidos los Perfiles y las Configuraciones, que pueden ser empaquetados con J2ME conjuntamente para crear una gran pila de software.

Entre los paquetes opcionales para dispositivos móviles se encuentra el API de Bluetooth, JSR 82, que incluye los recursos necesarios para crear aplicaciones que permitan controlar el Bluetooth del dispositivo móvil y por lo tanto, será usado en el presente proyecto.

2.2.4.1 APIs de Java para Bluetooth, JSR-82

JSR-82, Java Specification Request for Bluetooth, es el nombre oficial JCP para el conjunto de APIs para Bluetooth. Los APIs JSR 82 son muy flexibles, ya que permiten trabajar tanto con aplicaciones nativas Bluetooth como con aplicaciones Java Bluetooth.

JSR-82 permite, entre otras, las siguientes operaciones:
a. Registro de servicios.
b. Descubrimiento de dispositivos y servicios.
c. Establecer conexiones entre dispositivos.
d. Usar dichas conexiones para mandar y recibir datos
e. Manejar y controlar las conexiones de comunicación.
f. Ofrecer seguridad a dichas actividades[I]..

2.2.4.1.1 Paquetes incluidos en el JSR82

Actualmente el JSR-82 consta de dos paquetes independientes:
a. javax.bluetooth, son 13 clases e interfaces necesarios para establecer una comunicación inalámbrica usando el protocolo Bluetooth.
b. javax.obex, son 8 clases necesarias para enviar objetos entre dos dispositivos, independientemente del mecanismo de transporte usado entre ellos.

El presente proyecto estará basado en las clases incluidas en el paquete javax.bluetooth. En la Tabla 2.4 se pueden observar una breve descripción.

CLASE	DESCRIPCIÓN
DiscoveryListener	Esta interfaz permite recibir eventos de descubrimiento de dispositivos y de descubrimiento de servicios.
L2CAPConnection	Esta interfaz representa un canal L2CAP orientado a conexión.
L2CAPConnectionNotifier	Esta interfaz provee un notificador de una conexión L2CAP.
ServiceRecord	Esta interfaz describe las características de un servicio Bluetooth.
DataElement	Esta clase define los varios tipos de datos que se puede tener

Sistema Bluetooth para el control de dispositivos.

	como valor de un atributo de un servicio Bluetooth.
DeviceClass	Esta clase representa el tipo de dispositivo,'class of device (CoD)', grabado definido por la especificación Bluetooth.
DiscoveryAgent	Esta clase provee métodos para establecer el descubrimiento de dispositivos y de servicios.
LocalDevice	Esta clase representa el dispositivo Bluetooth.
RemoteDevice	Esta clase representa un dispositivo Bluetooth remoto.
UUID	UUID, esta clase define identificadores universalmente únicos, '*universally unique identifiers,* UUID'
BluetoothConnection Exception	BluetoothConnectionException es devuelto cuando la conexión Bluetooth (L2CAP, RFCOMM, OBEX) no pudo ser establecida exitosamente.
BluetoothStateException	La excepción BluetoothStateException es devuelta cuando una petición se hace para el sistema Bluetooth que puede soportar en su estado actual
ServiceRegistrationExce ption	ServiceRegistrationException es devuelta cuando se produce un fallo al grabar un servicio a la base de datos local, *Service Discovery Database* (SDDB) o para modificar uno existente. Authenticator Esta interfaz provee una forma de responder a un desafío y a una cabecera de respuesta de autenticación.
ClientSession	Esta interfaz provee métodos para requerimientos OBEX.
HeaderSet	Esta interfaz define métodos que establecen y dan los valores a las cabeceras OBEX.
Operation	Esta interfaz provee formas de manipular una operación OBEX única PUT o GET.
SessionNotifier	Esta interfaz define un notificador para conexiones OBEX serverside.
PasswordAuthentication	Esta clase lleva las combinaciones de nombre y contraseña.
ResponseCodes	Esta clase contiene la lista de los códigos de respuesta válidos que un servidor puede enviar a un cliente.
ServerRequestHandler	Esta clase define un evento de escucha que responderá a los requerimientos OBEX hechos por el servidor.

Tabla 2.4 Clases incluidas en javax.bluetooth [7]. [I].

2.2.4.1.2 Beneficios del API de JAVA para Bluetooth

Hay dos ventajas claves para usar los APIs oficiales de Java para Bluetooth sobre un API basado en C (nativo):

a. API es independiente del stack y del radio enlace.
b. Bluetooth API está estandarizado a nivel mundial.

Al usar estas librerías específicas de Bluetooth sobre el lenguaje Java tradicional, es posible crear una infinidad de aplicaciones.

2.3 APLICACIONES JAVA ME

Este proyecto incluye el desarrollo de una aplicación Java basada en el perfil MIDP sobre la configuración CLDC y que además usa el API de Bluetooth.

Para conseguir una aplicación Java ME se debe pasar por varias fases desde la edición del código hasta poder tener listo el MIDlet que se instalará en el dispositivo móvil. Las fases de desarrollo de un MIDlet se resumen a continuación.

2.3.1 FASE DE EDICIÓN

El código fuente o código de programación se puede editar en programas especializados como NetBeans, Eclipse, etc. ampliamente usados para crear aplicaciones de Java a cualquier nivel; o simplemente en un editor de texto cualquiera como Notepad de Windows, como se observa en la Figura 2.4. El archivo debe llevar la extensión *.java* que lo identifica como código de programación Java para pasar a las siguientes fases.

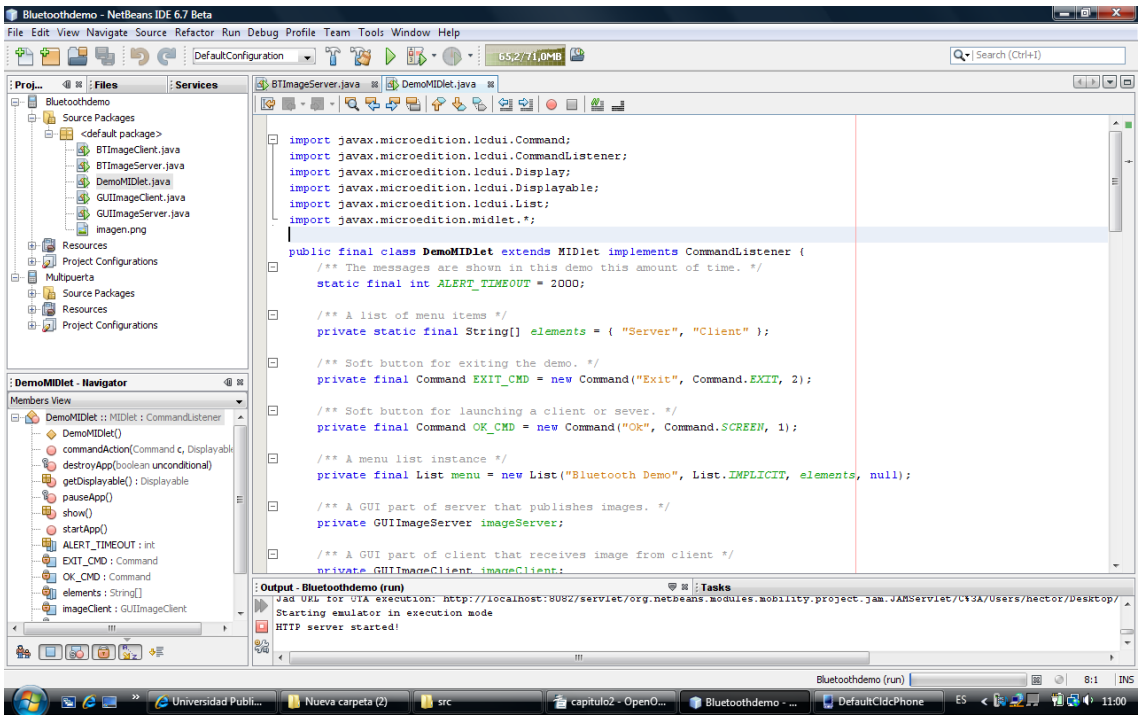


Figura 2.4 Edición del Programa Bluetoothdemo.java

2.3.2 FASE DE COMPILACIÓN

Para el proceso de compilación también se pueden usar cualquiera de los programas citados anteriormente. Además, se pueden usar otras herramientas creadas específicamente para la programación de equipos móviles, por ejemplo: “Sun One Studio” de Sun Microsystems o “JBuilder” de Borland.

Sistema Bluetooth para el control de dispositivos.

Para usar cualquiera de estas herramientas, es imprescindible tener instalada una Máquina Virtual Java en el computador, que también se la conoce como Entorno de Ejecución Java y es un programa nativo, ejecutable desde Windows, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (bytecode), el cual es usado por cualquier compilador del lenguaje Java.

Sin importar el compilador que se use, las aplicaciones Java estarán organizadas en proyectos, el resultado de un proyecto será el MIDlet.

Los proyectos están organizados con una estructura de directorios donde se puede encontrar los archivos generados.

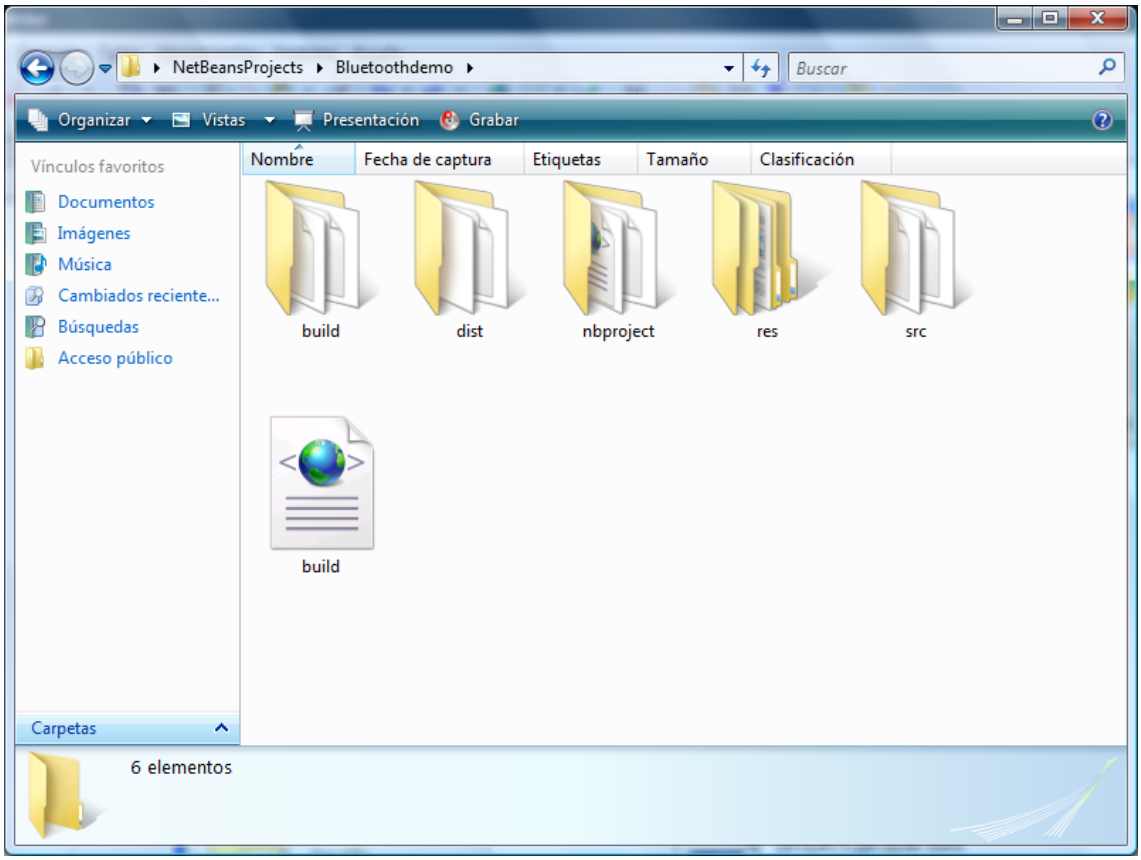


Figura 2.5 Directorio del Proyecto

En la Figura 2.5 se puede observar una vista del directorio creado usando Sun NetBeans IDE 6.7 beta.

En la carpeta *scr* se ubicarán los archivos *.java*, en la carpeta *res* todos los *recursos* como gráficos, fotos, sonidos, etc. en la carpeta bin los archivos ejecutables que deberán importarse a los móviles para que la aplicación se ejecute en estos, etc.

2.3.4 FASE DE DEPURACIÓN Y EJECUCIÓN

Una vez que se han completado los procesos de compilación y preverificación, el MIDlet se puede ejecutar en un simulador.

Sistema Bluetooth para el control de dispositivos.

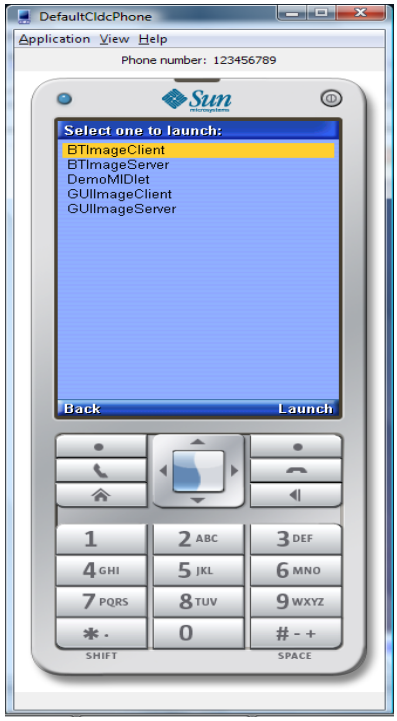


Figura 2.7 Modelos de teléfonos para simulaciones

En fases de prueba se deberá ejecutar el MIDlet sobre un emulador. Se pueden hacer a través del comando Run en el Sun Java Wireless Toolkit o F6 en NetBeans.

2.3.5 FASE DE EMPAQUETAMIENTO

Una vez que se han realizado todas las pruebas y simulaciones en el computador, la aplicación deberá ser empaquetada. Los compiladores empaquetan la aplicación y dan como resultado dos tipos de archivos: los archivos JAR y los archivos JAD. Un archivo JAR es un archivo comprimido (en formato ZIP) que contiene las clases (*.class*) que ha generado la compilación de nuestro programa, puede contener los recursos necesarios para el MIDlet como sonidos, gráficos, etc.

El segundo archivo necesario para la instalación de MIDlets son los archivos JAD. El archivo JAD contiene información necesaria para la instalación de los MIDlets contenidos en el archivo JAR. Un archivo puede contener más de un MIDlet. Cuando ocurre esto, se estaría hablando de un MIDlet suite.

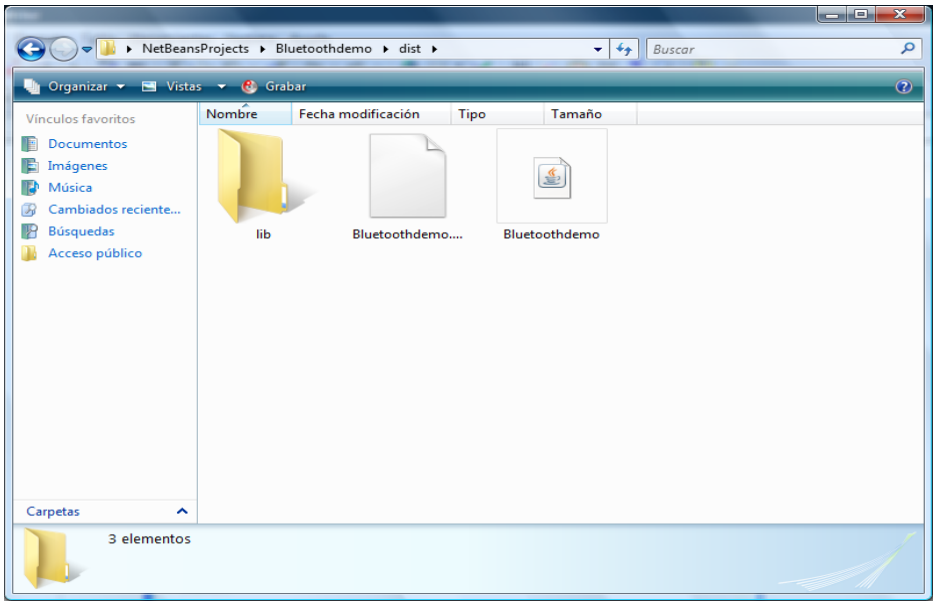


Figura 2.8 Contenido de la carpeta *dist*

Los parámetros contenidos en el archivo JAD pueden ser editados mediante el botón Settings de KToolBar o en el menú propiedades, en NetBeans. Aquí se puede editar la información del MIDlet como el nombre, la versión o el autor del MIDlet (o de los MIDlets). Estos archivos se guardan en la carpeta dist del proyecto, como se puede ver en la Figura 2.8.

2.3.5.1 Archivo JAR

Un archivo JAR es un archivo comprimido (en formato ZIP) que contiene todas las clases (*class*) que se ha generado después de la compilación de un programa. Además puede contener los recursos necesarios para el MIDlet como sonidos, gráficos, etc. Finalmente, contiene el archivo de manifiesto. Este archivo contiene información sobre las clases contenidas en el archivo JAR.

2.3.5.2 Archivo JAD

El archivo JAD contiene información necesaria para la instalación de los MlDlets contenidos en el archivo JAR. Un archivo puede contener más de un MlDlet. Cuando ocurre esto, se trata de un *MIDIet suite.*

El Gestor de Aplicaciones puede usar este archivo para obtener información útil sobre el *MIDlet*. Este archivo al igual que el manifiesto define una serie de atributos. El archivo JAD es opcional, y debería poseer los atributos detallados a continuación en la Tabla 2.7:

ATRIBUTO	DESCRIPCIÓN
MIDlet-Name	Nombre de la MIDlet suite.
MIDlet-Vendor	Nombre del desarrollador.
MIDlet-Version	Versión del MIDlet.

Sistema Bluetooth para el control de dispositivos.

MIDlet-Configuration	Configuración necesitada para ejecutar el MIDlet.
MIDlet-Profile	Perfil necesitado para ejecutar el MIDlet.
MIDlet-Jar-URL	URL del archivo JAR de la MIDlet suite.
MIDlet-Jar-Size	Tamaño en bytes del archivo JAR.
Atributos opcionales	Descripción
MIDlet-Data-Size	Mínimo número de bytes de almacenamiento persistente usado por el MIDlet.
MIDlet-Delete-Confirm	Confirmación a la hora de eliminar el MIDlet.
MIDlet-Description	Descripción de la MIDlet suite.
MIDlet-Icon	Archivo png incluido en el JAR.
MIDlet-Info-URL	URL con información de la MIDlet suite.
MIDlet-Install-Notify	Indica que el AMS notifique al usuario de la instalación del nuevo MIDlet.

Tabla 2.7 Atributos contenidos en el archivo JAD[XII].

Todas las fases que se mencionaron se deben realizar sistemáticamente y se volverán a repetir cada vez que se haga algún cambio en el código fuente. Si en alguna fase se presenta un problema, será necesario modificar el código fuente y volver a comenzar el proceso.

En el siguiente capítulo se profundizará en el desarrollo del código para crear una aplicación que permita realizar una conexión Bluetooth.

CAPÍTULO 3. DESARROLLO DE UNA APLICACIÓN BLUETOOTH/J2ME.

El Lenguaje Java es una Plataforma demasiado extensa, por esta razón se debe escoger el entorno adecuado según la aplicación que se quiera conseguir. Este proyecto se define dentro del entorno Java ME, para microdispositivos (J2ME).

En este capítulo se va ha desarrollar el código en Lenguaje Java, es decir, se programará el MIDlet que permita usar la tecnología Bluetooth que incorporan los dispositivos móviles, de una manera intuitiva y fácil de manipular por parte del usuario.

3.1 HERRAMIENTAS DE DESARROLLO

Este proyecto ha sido creado con la ayuda de NetBeans IDE 6.7 Beta para Windows. También ha sodo probado con la ayuda de Sun Java Wireless Toolkit 2.5.2 for CLDC y simulado también con la ayuda de CLDC KToolBar. A continuación se revisarán algunas características de estas herramientas.

3.1.1 NETBEANS IDE 6.7 Beta

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cientos de socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

NetBeans es una plataforma para el desarrollo de aplicaciones Java usando un Entorno de Desarrollo Integrado (IDE), una herramienta para programadores usada para editar, compilar, depurar y ejecutar programas. NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, Web, EJB y aplicaciones móviles).

En la Figura 3.1 se observa la presentación de NetBeans IDE 6.7 Beta, programa con el que se ha realizado este proyecto.

Figura 3.1 NetBeans IDE 6.7 Beta

NetBeans Mobility es un módulo usado por la plataforma NetBeans para desarrollar aplicaciones móviles, incluye las librerías necesarias y la opción de edición del código fuente de manera interactiva para implementar aplicaciones para dispositivos móviles.

3.1.2 SUN JAVA WIRELESS TOOLKIT 2.5 FOR CLDC

Es un conjunto de herramientas para el desarrollo de aplicaciones inalámbricas que se basan en la plataforma J2ME, *Connected Limited Device Configuration* (CLDC) y *Mobile Information Device Profile* (MIDP), diseñadas para funcionar en teléfonos móviles y otros pequeños dispositivos móviles. Sun Java Wireless Toolkit 2.5 incluye entornos de emulación, características de optimización y rendimiento, documentación y ejemplos muy útiles para crear aplicaciones. Es similar a J2ME Wireless Toolkit 2.2 e incluye varios modelos de teléfonos para la simulación.

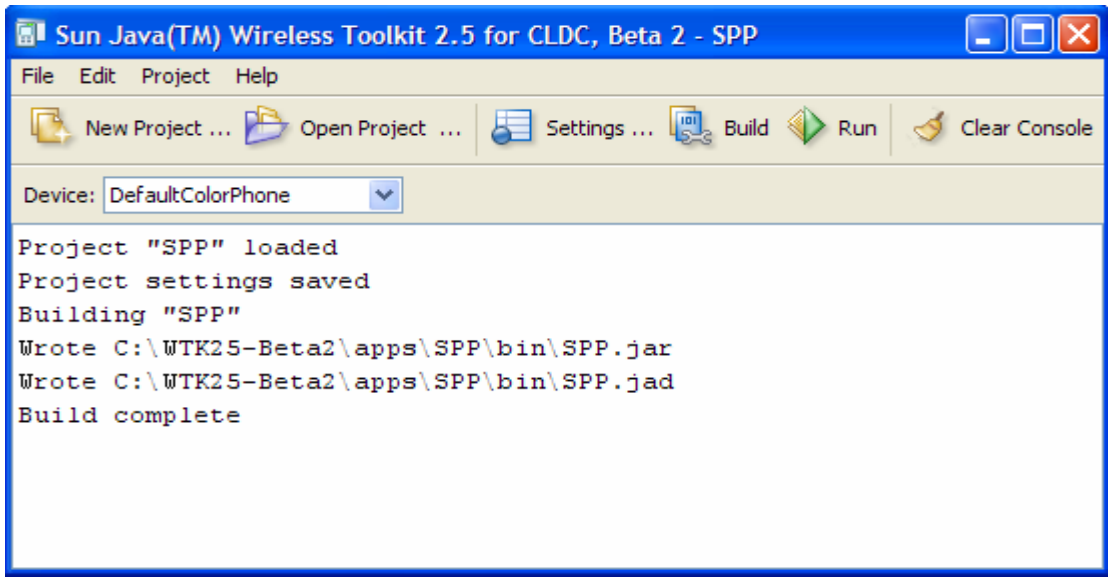


Figura 3.2 Pantalla principal de CLDC KTollbar

Sistema Bluetooth para el control de dispositivos.

3.2 PROGRAMACIÓN EN JAVA ME

Un MIDlet tiene que ejecutarse en un entorno muy concreto (un dispositivo con soporte Java ME). Un MIDlet tiene que heredar de la clase MIDlet e implementar una serie de métodos de dicha clase. La clase de la que ha de heredar cualquier MIDlet es javax.microedition.midlet.MIDIet.*

Un MlDlet puede estar en tres estados diferentes: en ejecución, en pausa o finalizado. Dependiendo del estado en el que esté, la máquina virtual llamará al método heredado correspondiente, es decir, startApp() cuando entre en ejecución, pauseApp() cuando el MlDlet entre en pausa y destroyApp() a la finalización del MIDlet[7]..

Las clases de javax.microedition.Icdui.* dan soporte a la interfaz de usuario. Permiten controlar la pantalla del dispositivo y también la entrada/salida desde el teclado.

Dentro de la aplicación, la interfaz de usuario interactua con la interfaz de comunicación, la cual establece la conexión Bluetooth, para alcanzar el objetivo de control de este proyecto.

En la Figura 3.3 se puede observar el Diagrama de Flujo definido para esta aplicación.

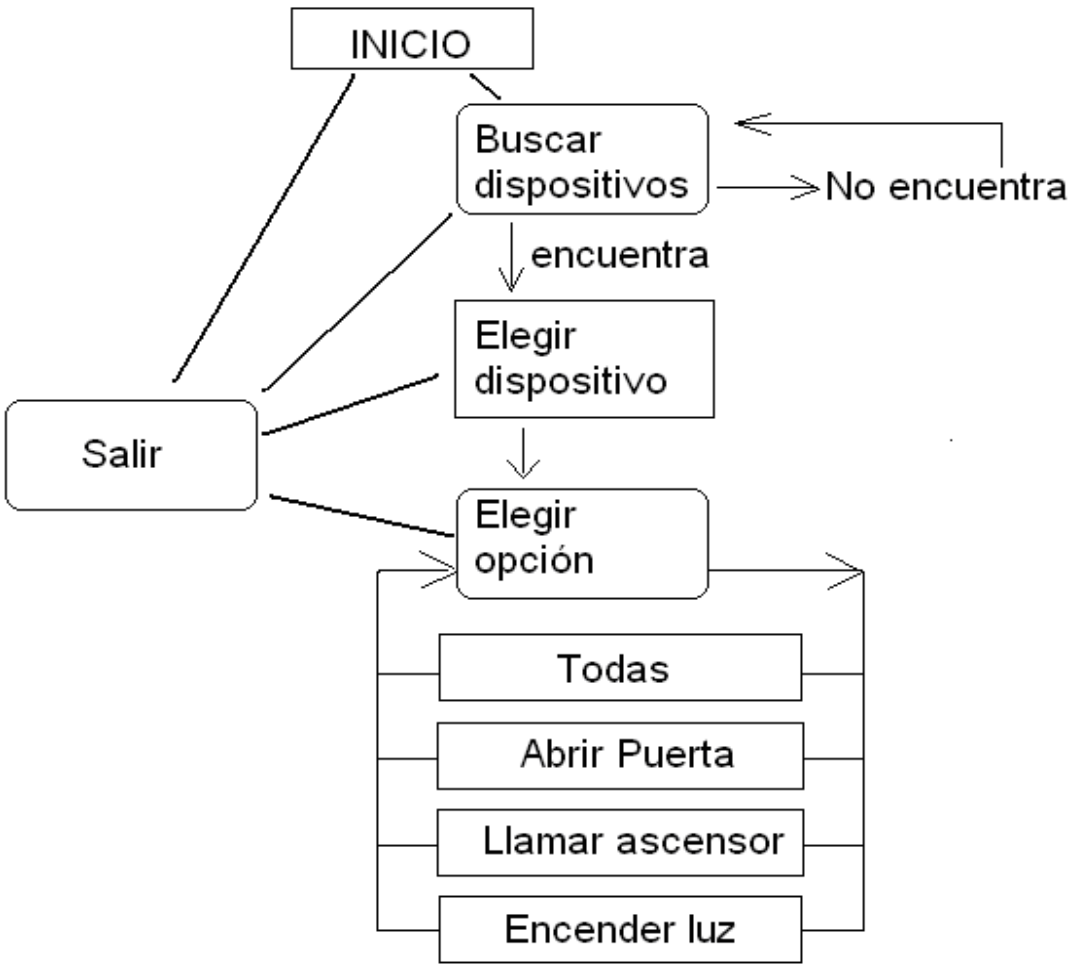


Figura 3.3 Diagrama de Flujo de la Aplicación Bluetooth

Sistema Bluetooth para el control de dispositivos.

3.2.1 PROGRAMACIÓN DE LA INTERFAZ DE USUARIO

La aplicación Java ME estará sustentada principalmente en el API JSR 82, por otro lado CLDC que hereda algunas de las clases de J2SE, y MIDP que añade nuevas clases que permitirán crear interfaces de usuario.

Las clases más importantes de J2SE que ofrece CLDC son las siguientes[7]. [XIII].:
a. java.lang
b. java.util
c. java.io

Además MIDP añade los siguientes paquetes:

a. javax.microedition.midlet
b. javax.microedition.lcdui
c. javax.microedition.io
d. javax.microedition.rms

El paquete javax.microedition.midlet, es el más importante de todos. Sólo contiene a la clase MIDlet, que ofrece un marco de ejecución para aplicaciones sobre dispositivos móviles. El paquete javax.microedition.lcdui ofrece una serie de clases e interfaces de utilidad para crear interfaces de usuario.

3.2.1.1 Elementos de la Interfaz de Usuario

Comand es un elemento que permite interaccionar con el usuario y le permite introducir comandos. Están disponibles los siguientes tipos de comandos:

COMANDO	DESCRIPCIÓN
OK	Confirma una selección
CANCEL	Cancela la acción actual
BACK	Traslada al usuario a la pantalla anterior
STOP	Detiene una operación
HELP	Muestra una ayuda
SCREEN	Tipo genérico referente a la pantalla actual
ITEM	Tipo genérico referente a un elemento de la pantalla actual

Tabla 3.1 Tipos de comandos[XIII].

A veces, y dependiendo del modelo y marca del dispositivo, sólo se pueden mostrar un número limitado de comandos en la pantalla. Al resto se puede acceder mediante un menú. En la Figura 3.4 se puede observar una parte del código de este programa. La aplicación Menu() define varios comandos con sus respectivas características. En la función Menu() se define un formulario y se le añaden los comandos creados.

```
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Form;
import javax.microedition.midlet.*;
/**
 * @author hector
 */
public class Menu extends MIDlet {
    private Command comenzar = new Command("Buscar",    Command.ITEM, 1);
    private Command cancelar = new Command("Cancelar",  Command.ITEM, 1);
    private Command inicio   = new Command("inicio",    Command.ITEM, 1);
    private Command fin      = new Command("fin",       Command.ITEM, 1);
    private Command pausa    = new Command("pausa",     Command.ITEM, 1);
    private Command arranque = new Command("arranque",  Command.ITEM, 1);
    private Form busqueda;
    public void startApp() {
        busqueda = new Form ("MENU");
        busqueda.addCommand(comenzar);
        busqueda.addCommand(fin);
        busqueda.addCommand(pausa);
        busqueda.addCommand(arranque);
        busqueda.addCommand(inicio);
        busqueda.addCommand(cancelar);
        Display.getDisplay(this).setCurrent(busqueda);
    }
```

Figura 3.4 Dentro de la aplicación se definen los comandos

Figura 3.5 Comandos agrupados en un menu dentro del formulario

3.2.1.1.1 La clase Screen

Hereda directamente de Displayable y permite crear las interfaces gráficas de alto nivel. Un objeto que herede de la clase Screen será capaz de ser mostrado en la pantalla. Se pueden encontrar cuatro clases que heredan de Screen y que sirven de base para crear las interfaces de usuario, son: Alert, Form, List y TextBox[7]..

Se puede imaginar como una serie de fichas de las cuales sólo se puede mostrar una cada vez. Para cambiar de una pantalla a otra se debe usar el método setCurrent de la clase Display.

Cada uno de las clases anteriores dispone de los métodos (realmente lo heredan de Screen):

a. String getTitle () - Devuelve el título de la pantalla
b. void setTitle (String s) - Establece el título de la pantalla
c. Ticker getTicker () - Devuelve el ticker de la pantalla
d. void setTicker(Ticker ticker) - Establece el ticker de la pantalla

Ticker es una línea de texto que aparece en la parte superior de la pantalla con un scroll lateral automático.

La clase Alert, permite mostrar una pantalla de texto durante un tiempo o hasta que se produzca un comando de tipo OK. Se utiliza para mostrar errores u otro tipo de mensajes al usuario. El tipo de alerta puede ser uno de los siguientes: ALARM, CONFIRMATION, ERROR, INFO, WARNING [7]..

La diferencia entre uno y otro tipo de alerta es básicamente el tipo de sonido o efecto que produce el dispositivo. En la Figura 3.6 se puede observar el código usado para crear este tipo de pantalla a través del método get_Informacion().

```
public void Alert () {
        pantalla  = Display.getDisplay(this);
        pantalla.setCurrent(null);
        Alert alerta = new Alert("ERROR","Fuera de cobertura", null, AlertType.ERROR);
        pantalla.setCurrent(alerta);
        alerta.setTimeout(10000);
}
```

Figura 3.6 Pantalla de Alerta

En este método, se define un Alert llamado Información, se le añade el título, un texto, una imagen y se configura el tiempo que se debe mostrar. La Figura 3.7 muestra el resultado del código anteriormente detallado.

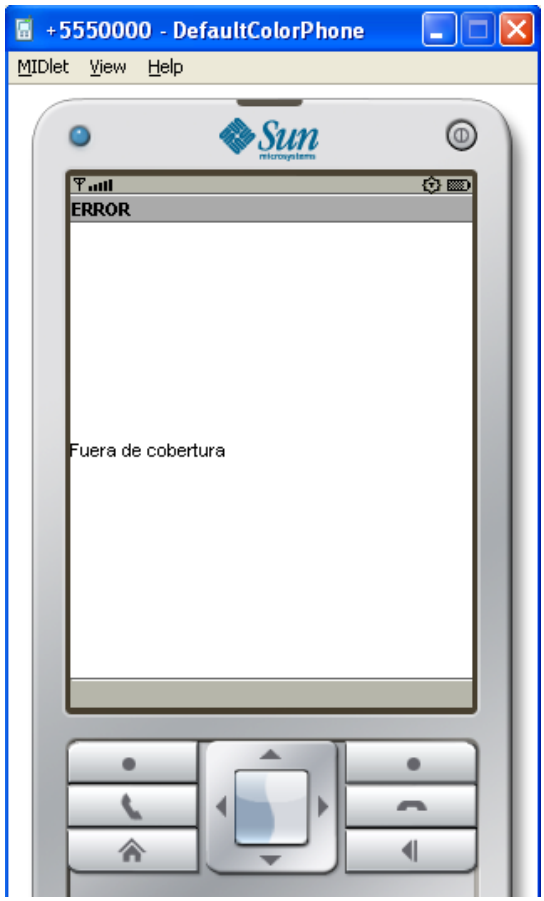


Figura 3.7 Alerta de Información mostrada en la aplicación

La clase List, permite crear listas de elementos seleccionables. Los principales tipos de lista son:

EXCLUSIVE, que permite seleccionar un solo elemento a la vez.
IMPLICIT, que permite seleccionar un elemento usando un comando;
MULTIPLE, que permite tener varios elementos seleccionados simultáneamente.

Un ejemplo del uso de esta clase se encuentra en la Figura 3.8, una parte del código fuente que incluye la clase List [7]..

```
private void Acciones (){
        acciones = new List("Opciones", List.IMPLICIT);
        acciones.append("Seleccionar todas", null);
        acciones.append("Abrir puerta", null);
        acciones.append("Llamar ascensor", null);
        acciones.append("Encender luz", null);
        acciones.addCommand(Salir);
        acciones.setCommandListener(this);
        display.setCurrent(acciones);
    }
```

Figura 3.8 Código fuente para crear una List.

Sistema Bluetooth para el control de dispositivos.

En el método Acciones () se define la lista Opciones con los elementos que se pueden seleccionar, el texto que se debe mostrar, una imagen para cada opción y además se añaden comandos a la lista y se define la primera opción seleccionada por defecto. En la Figura 3.9 se puede observar el resultado.

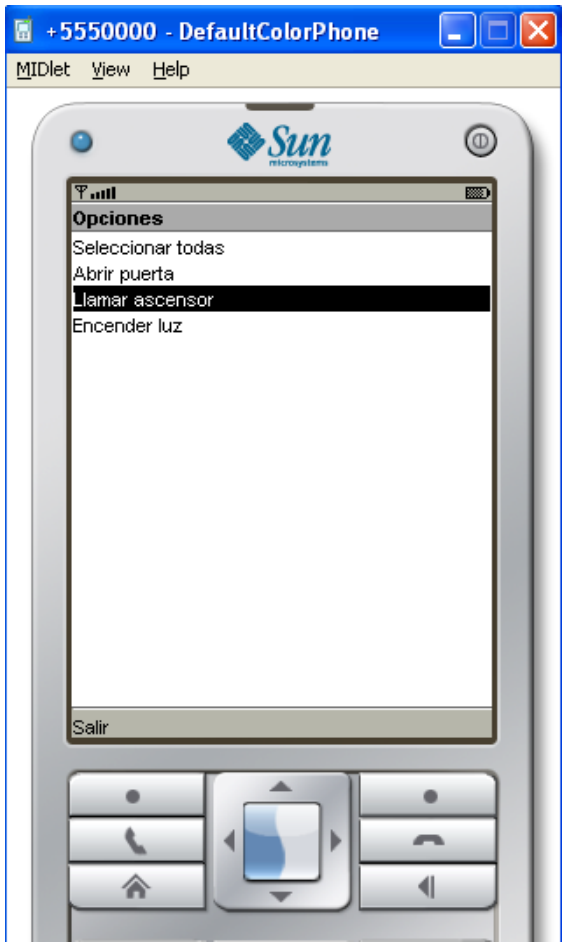


Figura 3.9 Lista Opciones para seleccionar un formulario

La clase TextBox permite introducir y editar texto a pantalla completa. Es como un pequeño editor de textos. Las limitaciones o tipos de texto aceptado pueden ser alguno de los siguientes: ANY, sin limitación; EMAILADDR, dirección de email; NUMERIC, sólo se permiten números; PASSWORD, los caracteres no serán visibles; PHONENUMBER, sólo número de teléfono; URL, sólo direcciones URL.

```
public Form get_form_cAcceso() {
    if (form_cAcceso == null) {
        form_cAcceso = new Form("Control de Acceso");
        TextField_clave = new TextField("Ingrese Clave", "",20, TextField.PASSWORD);
        form_cAcceso.setTicker(ticker);
        form_cAcceso.append(TextField_clave);
    }
         return form_cAcceso;
}
```

Figura 3.10 Código para insertar un Cuadro de Texto.

Sistema Bluetooth para el control de dispositivos.

El método get_form_cAcceso() define un formulario que incluye un cuadro de texto, TextField_clave, de tipo PASSWORD. En la Figura 3.11 se observa dicho formulario y el cuadro de texto con el texto oculto.

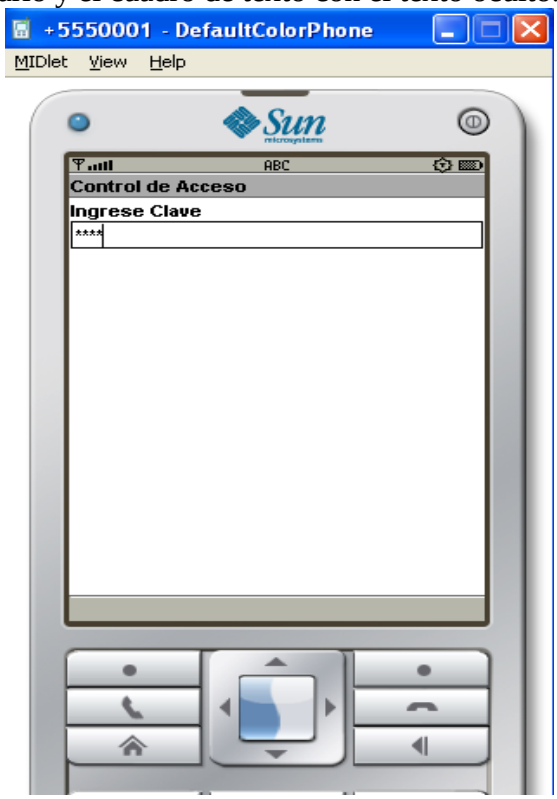


Figura 3.11 Formulario que incluye un cuadro de texto

La clase Form es un elemento de tipo contenedor, es decir, es capaz de contener una serie de elementos visuales con los que se construyen las interfaces más elaboradas. Los elementos que se podrían añadir a un formulario son [7].:

a. StringIItem
b. lmageItem
c. TextField
d. DateField
e. ChoiceGroup
f. Gauge

En las Figuras 3.8 y 3.10 se puede apreciar ejemplos de métodos que definen formularios.

La clase Form es capaz de manejar objetos derivados de la clase Item, que representa a un elemento visual que no ocupará toda la pantalla, sino que formará parte de la interfaz de usuario junto con otros elementos.

Hay métodos de la clase Form que permiten añadir, eliminar y modificar elementos del formulario, son las siguientes:

a. append() añade al formulario un elemento.
b. delete() elimina un elemento del formulario.
c. insert() inserta un elemento en una posición indicada

Sistema Bluetooth para el control de dispositivos.

3.2.2 PROGRAMACIÓN DE LA APLICACIÓN DE COMUNICACIÓN

Un MlDlet puede establecer diversos tipos de conexiones: *Sockets, http, https, datagramas, bluetooth* y otras[7]. [XIII]..

En una comunicación Bluetooth existe un dispositivo que ofrece un servicio (servidor) y otros dispositivos que acceden a él (clientes). En este caso se está programando la parte del cliente y la parte del servidor está ya implementada en el módulo Bluetooth que ofrece el servicio de puerto serie, SPP.

El módulo Bluetooth, como servidor, deberá hacer las siguientes operaciones:

a. Crear una conexión servidora
b. Especificar los atributos de servicio
c. Aceptar las conexiones clientes

La aplicación Bluetooth debe realizar las siguientes funciones:

a. Búsqueda de dispositivos. La aplicación realizará una búsqueda de los dispositivos Bluetooth a su alcance que estén en modo conectable.

b. Búsqueda de servicios. La aplicación realizará una búsqueda de servicios por cada dispositivo encontrado.

c. Establecimiento de la conexión. Una vez encontrado un dispositivo que ofrece el servicio deseado se podrá realizar la conexión.

d. Comunicación. Ya establecida la conexión es posible leer y escribir sobre ésta.

Estas funciones se revisarán a continuación analizando el código que permite llevarlas a cabo.

3.2.2.1 Búsqueda de Dispositivos

3.2.2.1.1 BCC (Bluetooth Control Center)

El BCC es un conjunto de capacidades que permiten al usuario o al fabricante resolver conflictos entre las aplicaciones previniendo que una aplicación pueda perjudicar a otra. Gracias al BCC, los dispositivos que implementen el API de Bluetooth pueden permitir que múltiples aplicaciones se ejecuten simultáneamente [1].

El BCC o pila Bluetooth, puede ser una aplicación nativa, una aplicación en un API separado, o sencillamente un grupo de parámetros fijados por el proveedor que no pueden ser cambiados por el usuario.

El BCC define las características de seguridad. Se encarga de manejar el modo de seguridad que la pila debe usar y mantendrá las listas de dispositivos seguros. El API de Sistema Bluetooth para el control de dispositivos.

Bluetooth permite a la aplicación especificar sus requerimientos de autenticación y encriptación.

La pila Bluetooth es la responsable de controlar el dispositivo Bluetooth, por lo que es necesario inicializarla antes de iniciar el proceso de conexión. La especificación deja la implementación del BCC a los fabricantes, y cada uno maneja la inicialización de una manera diferente.

3.2.2.1.2 Habilitación dispositivo local

Los objetos Bluetooth esenciales en una conexión son LocalDevice y RemoteDevice. La clase LocalDevice provee acceso y control del dispositivo local. Las clases DeviceClass y BluetoothStateException dan soporte a la clase LocalDevice. La clase RemoteDevice representa un dispositivo remoto y provee métodos para obtener información de dicho dispositivo remoto[1].

Clase javax.bluetooth.LocalDevice

Esta clase provee acceso y control sobre el dispositivo local Bluetooth. Está diseñada para cumplir con los requerimientos del Generic Access Profile, GAP, definidos para Bluetooth.

Un objeto LocalDevice representa al dispositivo local. Este objeto será el punto de partida de cualquier operación que se pueda llevar a cabo con este API. Alguna información de interés que se obtiene de este objeto a través de sus métodos es, por ejemplo, la dirección Bluetooth de nuestro dispositivo, el apodo o "friendlyName". A través de este objeto también se puede obtener y establecer el modo de conectividad: la forma en que nuestro dispositivo está o no visible para otros dispositivos usando el método setDiscoverable()[1].

Los posibles valores que admite el método setDiscoverable() están definidos en la clase DiscoveryAgent como campos estáticos. El valor que se usa para esta clase es GIAC, General/Unlimited Inquiry Access Code[1].

En la Figura 3.14, se observa el método comenzar() y el uso de algunos métodos de la clase localDevice para habilitar el Bluetooth en el dispositivo local.

```
private void comenzar() {
        LocalDevice localDevice = null;
        try {
                localDevice = LocalDevice.getLocalDevice();
                localDevice.setDiscoverable(DiscoveryAgent.GIAC);
                discoveryAgent = localDevice.getDiscoveryAgent();
                String dir_local = localDevice.getBluetoothAddress();
                Inicio.append("dispositivo local:"+dir_local);
        } catch(Exception e) {
                e.printStackTrace();
                Alert alert = new Alert("Error", "No se puede hacer uso de Bluetooth", null,
```

Sistema Bluetooth para el control de dispositivos.

```
AlertType.ERROR);
            Display.getDisplay(this).setCurrent(alert);
      }
}
```

Figura 3.14 Función comenzar()

Dado que los dispositivos inalámbricos son móviles, necesitan un mecanismo que permita encontrar, conectar, y obtener información sobre las características de dichos dispositivos, todas estas tareas son realizadas por el API de Bluetooth. La búsqueda de dispositivos y servicios son tareas que solamente realizarán los dispositivos clientes, en este caso, el teléfono móvil.

3.2.2.1.3 Descubrimiento de Dispositivos

A través de cualquier aplicación se puede encontrar dispositivos y crear una lista de todos ellos, para esto se usa startInquiry(). Este método requiere que la aplicación tenga especificado un “listener”, una interfaz que es notificada cuando un nuevo dispositivo es encontrado después de haber lanzado un proceso de búsqueda. Esta interfaz es javax.bluetooth.DiscoveryListener[1].

```
private void buscar() {
    dispositivosEncontrados= new Vector();
    etiquetasDospositivos= new Vector();
        try {
            discoveryAgent.startInquiry(discoveryAgent.GIAC, this);
        } catch(BluetoothStateException e) {
            e.printStackTrace();
            Alert alert = new Alert("AlertError","No se pudo comenzar la
            busqueda",null, AlertType.ERROR);
            Display.getDisplay(this).setCurrent(alert);
        }
}
```

Figura 3.15 Función Buscar()

La interfaz javax.bluetooth.DiscoveryAgent provee métodos para descubrir dispositivos y servicios. En este caso, para comenzar una nueva búsqueda de dispositivos es llamado el método startInquiry(), como se puede apreciar en la Figura 3.16, este método requiere dos argumentos. El primer argumento es un entero que especifica el modo de conectividad definido para la búsqueda, este valor deberá ser DiscoveryAgent.GIAC, para tener un acceso ilimitado. El segundo argumento es un objeto que implemente DiscoveryListener. A través de este último objeto serán notificados los dispositivos que se vayan descubriendo. Para cancelar la búsqueda se debe usar el método cancelInquiry() de la misma interfaz javax.bluetooth.DiscoveryAgent[1].

La interfaz DiscoveryListener tiene los siguientes métodos usados para el descubrimiento de dispositivos:

a. public void deviceDiscovered(RemoteDevice rd, DeviceClass c)

b. public void inquiryCompleted(int c)

Cada vez que se descubre un dispositivo se llama al método deviceDiscovered(RemoteDevice rd, DeviceClass c). Se pasan dos argumentos. El primero es un objeto de la clase RemoteDevice que representa el dispositivo encontrado. El segundo argumento permitirá determinar el tipo de dispositivo encontrado[1].

```
public void deviceDiscovered(RemoteDevice remoteDevice, DeviceClass deviceClass) {
    String address = remoteDevice.getBluetoothAddress();
    String friendlyName = null;
    try {
        friendlyName = remoteDevice.getFriendlyName(true);
    } catch(IOException e) { }
        String device = null;
        if(friendlyName == null) { device = address;
    } else { device = "("friendlyName" : "+address+")";
    }
    etiquetasDospositivos.addElement(device);
    dispositivosEncontrados.addElement(remoteDevice);
    Display.getDisplay(this).setCurrent(list_dispositivos);
}
```

Figura 3.16 Método deviceDiscovered()

En la Figura 3.16 se observa el código que permite obtener el friendlyName y la dirección de los dispositivos hallados en una búsqueda para presentarlos como elementos de una Lista llamada list_dispositivos.

El método inquiryCompleted(int c) es llamado cuando la búsqueda de dispositivos ha finalizado pasa un argumento entero indicando el motivo de la finalización. Este argumento podrá tomar los valores[1]:

DiscoveryListener.INQUIRY_COMPLETED si la búsqueda ha concluido con normalidad, DiscoveryListener.INQUIRY_ERROR si se ha producido un error en el proceso de búsqueda, o DiscoveryListener.INQUIRY_TERMINATED si la búsqueda fue cancelada. Para cada caso se ha definido una acción, se puede revisar en la Figura 3.17.

```
    public void inquiryCompleted(int i) {
        switch(i) {
    case DiscoveryListener.INQUIRY_COMPLETED:
    System.out.println("Busqueda de dispositivos concluida con normalidad");
    break;
    case DiscoveryListener.INQUIRY_TERMINATED:
    System.out.println("Busqueda de dispositivos cancelada");
    break;
    case DiscoveryListener.INQUIRY_ERROR:
    System.out.println("Busqueda de dispositivos finalizada debido a un error");
    break;
        }
```

```
            Display.getDisplay(this).setCurrent(list_dispositivos);
        }
```

Figura 3.17 Función inquiryCompleted(int i)

Clase javax.bluetooth.RemoteDevice

Esta clase representa al dispositivo Bluetooth remoto, el dispositivo con el que se podría realizar la conexión Bluetooth. De ella se obtiene la información básica acerca de un dispositivo remoto, su dirección Bluetooth y su friendlyName, entre otros, usando los métodos que se muestran en la Tabla 3.2, a continuación[1]..

MÉTODO	RESUMEN
authenticate()	Intenta autenticar el RemoteDevice.
authorize()	Determina si el RemoteDevice esta habilitado para acceder al servicio local provisto por la conexión.
encrypt()	Intenta activar o desactivar la encriptación para una conexión existente.
equals()	Determina si dos RemoteDevices son iguales.
getBluetoothAddress()	Devuelve la dirección Bluetooth del dispositivo.
getFriendlyName()	Retorna el nombre del dispositivo.
getRemoteDevice()	Retorna el dispositivo Bluetooth remoto de una conexión SPP, L2CAP u OBEX.
hashCode()	Calcula el código Hash para este objeto
isAuthenticated()	Determina si el RemoteDevice ha sido autenticado.
isAuthorized()	Determina si el RemoteDevice ha sido autorizado previamente por el BCC del dispositivo local para intercambiar datos.
isEncrypted()	Determina si el intercambio de datos con el
RemoteDevice	está bien encriptada.
isTrustedDevice()	Determina si el dispositivo es de confianza según el BCC.

Tabla 3.2 Métodos admitidos por la clase RemoreDevice

Al llamar al método getLocalDevice() se puede producir una excepción del tipo BluetoothStateException, cuando un dispositivo no puede atender una petición que normalmente atendería, probablemente debido a algún problema propio del equipo. Esto significa que no se pudo inicializar el sistema Bluetooth[1]..

La excepción BluetoothStateException extiende de java.io.IOException y no añade ningún método adicional.

El método getDeviceClass() devuelve un objeto de tipo DeviceClass. Este tipo de objeto describe el tipo de dispositivo, se puede saber, por ejemplo, si se trata de un teléfono o de un ordenador.

Sistema Bluetooth para el control de dispositivos.

3.2.2.2 Búsqueda de Servicios

En un área de cobertura se pueden encontrar otros dispositivos Bluetooth a más del módulo usado en este proyecto. Para identificar un dispositivo Bluetooth se puede aprovechar alguna característica que lo hace diferente de los demás, por ejemplo, su friendlyName o su Bluetooth Address[1]..

3.2.2.2.1 Selección del Dispositivo de Control

En el programa se lleva a cabo una búsqueda de dispositivos y muestra un lista de friedlyname encontrados si aparece: “BT_2010”, lo seleccionamos. En la Figura 3.18 se observa la parte del código del programa donde se crea la lista de dispositivos de friendlyName de cada dispositivo encontrado.

```
public void deviceDiscovered(RemoteDevice arg0, DeviceClass arg1) {
    lista = new List("Dispositivos", List.IMPLICIT);
    device = arg0;
    try {
        lista.append(device.getFriendlyName(true, null);
    } catch (IOException ex) {
        Alert alerta = new Alert("ERROR","No encuentra dispositivos", null,
AlertType.ERROR);
        display.setCurrent(alerta);
        ex.printStackTrace();
    }
    lista.addCommand(buscar);
    lista.addCommand(acabar);
    lista.setCommandListener(this);
    display.setCurrent(lista);
  }
```

Figura 3.18 Función seleccionarDispositivo()

3.2.2.2.2 Descubrimiento de Servicios

Para realizar una búsqueda de servicios también se usa la clase DiscoveryAgent y se implementa la interfaz DiscoveryListener[1].

La clase DiscoveryAgent provee de métodos para buscar servicios en un dispositivo servidor Bluetooth e iniciar transacciones entre el dispositivo y el servicio.

Para comenzar la búsqueda se usa searchServices() de la clase DiscoveryAgent, que se verá con más detalle[1]:

Método searchServices(int[], UUID[ ], RemoteDevice, DiscoveryListener)

El primer argumento es un array de enteros con el que se especifican los atributos del servicio en los que se esta interesado, algo así como su ID de servicio. En la interfaz *ServiceRecord*, cada servicio es descrito a través de atributos que son identificados Sistema Bluetooth para el control de dispositivos.

numéricamente. El segundo argumento es un array de identificadores de servicio. Permite especificar los servicios en los que el cliente está interesado. La clase UUID (*universally unique identifier*) representa indentificadores únicos universales. Más adelante, en la Figura 3.20 se observará que el valor de UUID usado en el presente proyecto es "0x1101" que es el correspondiente para el Perfil de Puerto Serie, SPP. El tercer argumento es el dispositivo remoto sobre el que se va a realizar la búsqueda.

Por último argumento se pasará un objeto que implemente DiscoveryListener que será usado para notificar los eventos de búsqueda de servicios.

En la Figura 3.19, se puede ver estas definiciones dentro del código del programa, en donde se hace la búsqueda de servicios en el dispositivo remoto seleccionado.

```
public void deviceDiscovered(RemoteDevice remoteDevice, DeviceClass deviceClass) {
String name = BT_2010;
      try {
         name = remoteDevice.getFriendlyName(true);
      } catch (IOException ex) {
         ex.printStackTrace();
      }
         discoveryAgent.cancelInquiry(this);
         rd = remoteDevice;
         try {
            discoveryAgent.searchServices(null, SERVICIOS, remoteDevice, this);
         } catch (BluetoothStateException e) {
            e.printStackTrace();
         }
}
```

Figura 3.19 Función que seleccionar a través de los servicios

Los campos del método discoveryAgent.searchServices, en el cuadro anterior, están declarados de la siguiente manera[1]:

```
public static final UUID SERVICIO_CONTROL = new UUID(0x1101);
public static final UUID[] SERVICIOS = new UUID[]{SERVICIO_CONTROL };
public static final int[] ATRIBUTOS = null;
private RemoteDevice remoteDevice2 =null;
```

Figura 3.20 Valores del método searchService

Es posible que se desee hacer diversas búsquedas de servicios al mismo tiempo. El método searchServices() devolverá un entero que identificará la búsqueda. Este valor entero servirá para saber a qué búsqueda pertenecen los eventos servicesDiscovered() y serviceSearchCompleted().

Método serviceSearchCompleted(int transID, int respCode)

Este método es llamado cuando se finaliza un proceso de búsqueda. El primer argumento identifica el proceso de búsqueda, el valor devuelto al invocar el método Sistema Bluetooth para el control de dispositivos.

searchServices() de la clase DiscoveryAgent(). El segundo argumento indica el motivo de finalización de la búsqueda.

En la Figura 3.21 se observa esta función dentro del código general de la aplicación.

```
public void serviceSearchCompleted(int transID, int respCode) {
switch (respCode) {
   case
DiscoveryListener.SERVICE_SEARCH_COMPLETED:
System.out.println("Busqueda completada con normalidad");
   break;
   case
DiscoveryListener.SERVICE_SEARCH_TERMINATED:
System.out.println("Busqueda cancelada");
   break;
   case
DiscoveryListener.SERVICE_SEARCH_DEVICE_NOT_REACHABLE:
System.out.println("Dispositivo no alcanzable");
   break;
   case
DiscoveryListener.SERVICE_SEARCH_NO_RECORDS:
System.out.println("No se encontraron registros de servicio");
   break;
   case
DiscoveryListener.SERVICE_SEARCH_ERROR:
System.out.println("Error en la busqueda");
   break;
}
}
```

Figura 3.21 Función serviceSearchCompleted

Se puede cancelar un proceso de búsqueda de servicios llamando al método cancelServiceSearch() pasándole como argumento el identificador de proceso de búsqueda, que es el número entero devuelto cuando se comenzó la búsqueda con searchServices()[1].

Método servicesDiscovered(int transID, ServiceRecord[] sr)

Este método notifica que se han encontrado servicios. El primer argumento es un entero que es el que identifica el proceso de búsqueda. Este entero es el mismo que devolvió searchDevices() cuando se comenzó la búsqueda.

El segundo argumento es un array de objetos ServiceRecord. Un objeto ServiceRecord describe las características de un servicio Bluetooth mediante atributos, los cuales se identifican numéricamente, es decir, un servicio Bluetooth tiene una lista de pares identificador-valor que lo describen. El objeto ServiceRecord se encarga de almacenar estos pares.

Los identificadores son números enteros y los valores son objetos de tipo DataElement. Los objetos DataElement encapsulan los posibles tipos de datos mediante los cuales se pueden describir los servicios Bluetooth. Estos tipos de datos son: valor nulo, enteros de diferente longitud, arrays de bytes, URLs, UUIDs, booleanos, Strings de los tipos anteriores[1].

```
public void servicesDiscovered(int transID,ServiceRecord[] servRecord) {
   ServiceRecord service = null;
   service = servRecord[seleccion];
   url = service.getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYPT, false);
   enviar(url);
}
```

Figura 3.22 Función servicesDiscovered()

La clase javax.bluetooth.RemoteDevice contiene los métodos que pueden ser usados en cualquier momento para hacer una petición de cambio en las configuraciones de seguridad, o de averiguar la configuración actual de seguridad en la conexión.

3.2.2.3 Establecimiento de la Conexión

Una vez que la aplicación Bluetooth ya ha realizado la búsqueda de los dispositivos Bluetooth a su alcance, seleccionamos al módulo Bluetooth y buscará información de sus servicios. Después se establecerá la conexión haciendo uso de la Url obtenida y posteriormente se transmitirán los datos.

connection = (StreamConnection) Connector.open(Url); //Así se establece una conexión SPP a través de Bluetooth[1].

3.2.2.3.1 Comunicación Cliente Servidor

El paquete javax.bluetooth permite utilizar dos mecanismos de conexión: SPP y L2CAP. En este caso se usará el perfil de puerto serie. Mediante SPP se obtiene un DataInputStream y un DataOutputStream. Para abrir una conexión se hace uso de la clase javax.microedition.io.Connector. En concreto se usará el método estático open(), su versión más sencilla requiere un parámetro que es un String que contendrá la URL con los datos necesarios para realizar la conexión.

La URL, necesaria para realizar la conexión, se obtiene a través del método getConnectionURL() de un objeto ServiceRecord. Un objeto ServiceRecord representa un servicio, es decir, una vez encontrado el servicio deseado (un objeto ServiceRecord), él mismo proveerá la URL necesaria para conectarse a él [1].

Este método requiere dos argumentos, el primero de los cuales indica si se debe autenticar y/o cifrar la conexión. Los posibles valores de este primer argumento son:

a. ServiceRecord.NOAUTHENTICATE_NOENCRYPT: No se requiere ni autenticación ni cifrado.

b. ServiceRecord.AUTHENTICATE_NOENCRYPT: Se requiere autenticación, pero no cifrado.

c.ServiceRecord.AUTHENTICATE_ENCRYPT: Se requiere tanto autenticación como cifrado.

El segundo argumento del método getConnectionURL() es un booleano que especifica si nuestro dispositivo debe hacer de maestro (true) o bien no importa si es maestro o esclavo (false) [1].

```
url = service.getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYPT, false);
```

Este código puede ser revisado en su contexto dentro de la función servicesDiscovered(), mostrada en la Figura 3.22.

Una vez que se tiene la URL, se utiliza el método Connector.open() para realizar la conexión. Este método devuelve un objeto distinto según el tipo de protocolo usado. En el caso de un cliente SPP devolverá un StreamConnection.

A partir del StreamConnection se obtienen los flujos de entrada y de salida:

```
StreamConnection     con = (StreamConnection) Connector.open(url); //se abre la conexión
DataOutputStream    out = con.openDataOutputStream(); //flujo de datos de salida
DataInputStream       in = con.openDataInputStream(); // flujo de datos de entrada
```

Estas definiciones están incluidas como parte del método enviar(url), que se revisará más adelante en la Figura.

Además, para el control de conexión se implementa un nivel de seguridad básico por medio de un PIN(personal identification number), que consiste en un PIN para emparejar dos dispositivos Bluetooth. Para poder utilizarlo primero se habilita en el dispositivo Bluetooth con el comando:

“AT+ZV Bond [BD addr] [PIN]”

Donde [BD addr] es la BD Address del dispositivo remoto y donde [PIN] es el PIN (máximo 16 caracteres). A partir de este momento cada vez que se pretenda establecer una conexión con este dispositivo nos realizara la petición del PIN para autorizar la conexión.

Una vez establecida la conexión ya es posible leer y escribir en ella concretando la comunicación.

3.2.2.4 Transmisión de Datos

Una vez establecida la conexión Bluetooth, se enviará un comando en forma de una cadena de texto, que será recibida e interpretada por el módulo Bluetooth.Para enviar una cadena de texto sobre una conexión SPP, se procede de la siguiente manera:

```
private void enviar(url){
StreamConnection connection = null;
DataInputStream      in = null;
DataOutputStream  out = null;
      try {
 connection = (StreamConnection)
Connector.open(url);
in   = connection.openDataInputStream();
out = connection.openDataOutputStream();
out.writeUTF(comando);
out.flush();
out.close();
in.close();
connection.close();
        } catch(IOException e) {
e.printStackTrace();
        } finally{
try {
if(out != null)
connection.close();
        } catch(IOException e) {}
}
}
```

Figura 3.23 Método enviar()

Dependiendo de la opción que se escoja en el menú, la variable “comando” tomará un valor diferente, en la Tabla 3.24, se pueden ver los valores correspondientes para cada opción del menú.

Opciones:

Abrir Puerta	AT+ZV GPIOWrite 05 1
Encender la Luz	AT+ZV GPIOWrite 10 1
Llamar al ascensor	AT+ZV GPIOWrite 07 1[Data-sheet1].

```
private class AbrirPuerta extends Thread {
public AbrirPuerta() {}
public void run(){
        try {
           connection = (StreamConnection) Connector.open(Url);
           out = connection.openDataOutputStream();
           in = connection.openDataInputStream();
           out.write("AT+ZV GPIOWrite 10 1\n".getBytes());
           out.flush();
           wait(10000);
           out.write("AT+ZV GPIOWrite 10 0\n".getBytes());
```

```
                    out.flush();
                    in.close();
                    out.close();
                    connection.close();
                } catch (InterruptedException ex) {
                    ex.printStackTrace();
                } catch (IOException ex) {
                    ex.printStackTrace();
                }
            Acciones();
        }
    }
```

Figura 3.24 Función AbrirPuerta()

La función AbrirPuerta() se ejecuta después de haber seleccionado una opción, de la lista de opciones posibles. En esta función también se definen elementos de datos de entrada y datos de salida, DataInputStream y DataOutputStream, porque la comunicación Bluetooth se realiza en dos vías. Se transmite el texto en modo String "AT+ZV GPIOWrite 10 1 \n" sobre la conexión establecida y luego se cierra la conexión para que el medio quede liberado y se pueda repetir el proceso.

3.2.3 PREPARACIÓN DE LA APLICACIÓN

En la fase de Compilación se generó el archivo *.Jar* asociado al archivo *.java* y se creó el directorio del proyecto que contiene toda la información que será usada para crear un MIDlet, incluyendo el archivo de código fuente, recursos, imágenes, y un descriptor del mismo. Después de compilar y preverificar, el MIDlet pasa a la fase de Depuración y Ejecución que se debe realizar en un simulador.

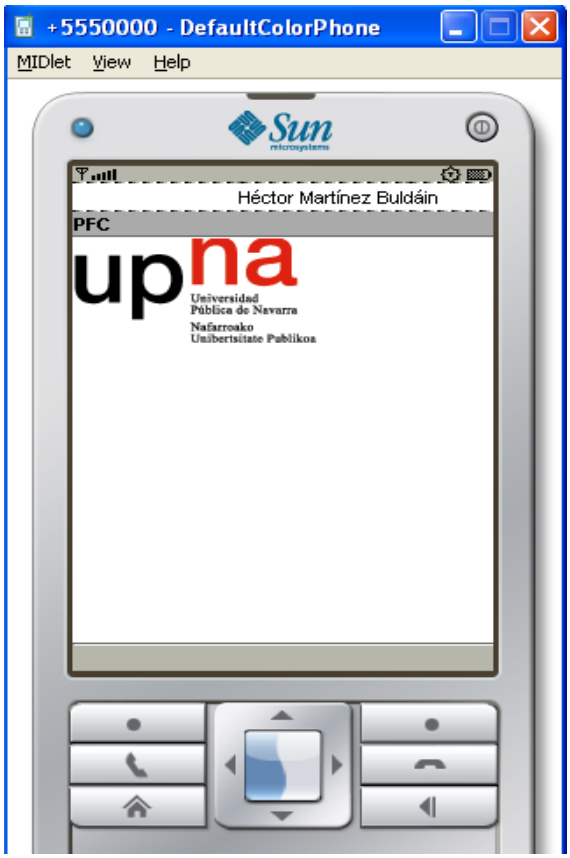


Figura 3.25 La aplicación ejecutándose en el simulador

Cuando ya se han realizado todas las pruebas y simulaciones en el computador, la aplicación puede ser transmitida al móvil, para hacerlo es necesario que pase por el proceso de empaquetamiento, aquí se prepara el MIDlet para que pueda ser descargado sobre el móvil.

3.2.3.1 Instalación de la Aplicación en el móvil

El proceso de empaquetamiento genera los archivos JAD, JAR en la carpeta *dist* del proyecto cuando se ha trabajado con NetBeans.

El MIDlet implementado es una aplicación como muchas que se pueden instalar normalmente en los teléfonos móviles, similares a los juegos Java que están bastante difundidos actualmente.

La instalación consiste en copiar los archivos JAR al teléfono móvil para que puedan ser ejecutados. Hay varias formas de hacerlo dependiendo de la marca y modelo del dispositivo. Se puede transferir como cualquier otra aplicación o archivo, usando infrarrojos, Bluetooth o un cable de datos desde un computador.

Existen muchas aplicaciones útiles para administración de archivos e instalación de aplicaciones en dispositivos móviles según la marca y modelo del teléfono móvil. Por ejemplo, para teléfonos Nokia se puede usar Nokia PC Suite, para Sony Ericsson también hay una versión de PC Suite, para Motorola se puede usar P2ktools, entre otros.

Sistema Bluetooth para el control de dispositivos.

CAPÍTULO 4. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA BLUETOOTH

En este capitulo se resume el diseño y la construcción de las interfaces de hardware para controlar las luces y el acceso a un portal desde el teléfono, es decir, los circuitos de control para los dispositivos eléctricos (cerradura eléctrica, luz, ascensor).

La parte de Hardware consiste en un circuito electrónico que incluye el módulo Bluetooth y las interfaces necesarias para realizar el control de dispositivos eléctricos.

Hasta este punto se ha desarrollado e instalado la aplicación Java en el teléfono móvil. El dispositivo móvil esta listo para conectarse vía Bluetooth con el módulo de control a través de una conexión puerto serie.

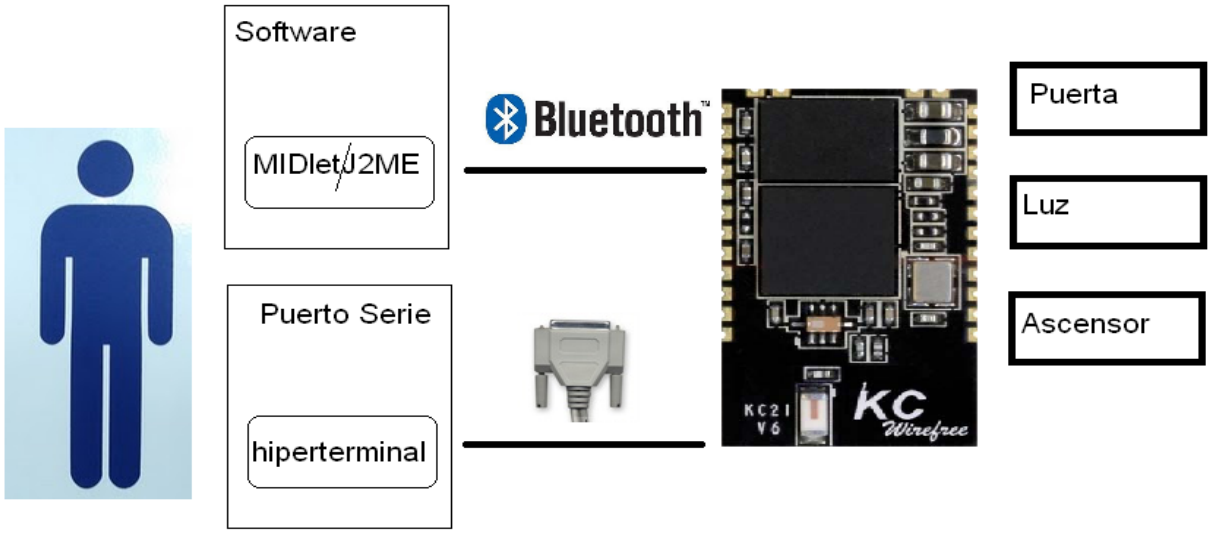


Figura 4.1. Diagrama del Proyecto

Este proyecto se puede utilizar para aplicaciones de control sencillo dentro de una vivienda: encendido de luces y apertura de puertas. Pero además se puede controlar el encendido y apagado de cualquier otro artefacto eléctrico que funcione con 220V, por ejemplo: una lámpara, un ventilador, aire acondicionado, pequeños motores, etc.

La parte de hardware es la encargada de aceptar la conexión Bluetooth solicitada por el dispositivo móvil y una vez establecida la conexión, se recibirán las instrucciones para realizar las acciones de control correspondientes.

El circuito electrónico se divide en tres partes: Circuitos de Alimentación, Circuito adaptador del Módulo Bluetooth y Circuitos Acopladores de Potencia.

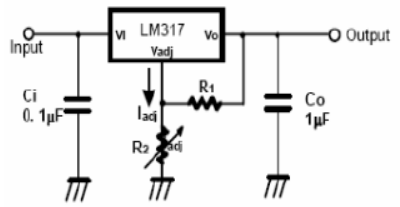
El diagrama del circuito se realizó con la ayuda del programa Pcad 2006. Una vez probado el circuito, el diseño del circuito impreso se realizó con la ayuda del mismo programa en una fresadora de circuitos impresos.

4.1 CIRCUITO DE ALIMENTACIÓN

El elemento más importante del proyecto es el chip Bluetooth KC-21. Según las especificaciones del fabricante necesita un voltaje de polarización de 3.3V y utiliza niveles lógicos TTL. Este módulo nos permite la posibilidad de establecer una comunicación directa con un ordenador a través de su puerto serie, para ello es necesario implementar un circuito adaptador de señales RS232 a TTL y viceversa, para ello se utiliza el chip integrado MAX232. Para su funcionamiento, este integrado se debe polarizar con 5V, para conseguir este voltaje fijo se utiliza un regulador de voltaje como el L7805CV. El voltaje de entrada para este regulador esta designado entre 7V y 25V [Data-sheet1].

Con los 5V que proporciona el regulador se consigue posteriormente un voltaje fijo de 3.3V para polarizar el KC-21. Para conseguir dicho voltaje se puede usar cualquier regulador ajustable, en este caso se está usando el LM317[Data-sheet7].

Configuración usada es la que el fabricante recomienda para una aplicación típica, de tal forma que para un Vo=3.3V los valores son los siguientes. Según la referencia del fabricante:



$$V_o = 1.25V\left(1+\frac{R_2}{R_1}\right)+I_{adj}\cdot R_2$$

Ci = 0.1uF se podría obviar, este capacitor es requerido cuando el regulador está a una distancia considerable del filtro de la fuente de poder.

Co = 1uF no es necesario para la estabilidad del regulador pero mejora la respuesta transitoria.

Mientras la Iadj sea menor de 100uA, el error asociado al voltaje de salida Iadj R2 es casi insignificante en la mayoría de aplicaciones.

Luego, para un Vo=3.3V y asumiendo que la Iadj=0, se tiene:

$$V_o = 1.25V\left(1+\frac{R_8}{R_7}\right)+I_{adj}\cdot R_8$$

$$3.3 = 1.2\left(1+\frac{R_8}{R_7}\right)+I_{adj}\cdot R_8$$

$$\frac{3.3}{1.2}-1=\frac{R_8}{R_7}$$

$$\frac{R_8}{R_7}=1.64$$

$$R_8 = 1.64\cdot R_7$$

$$\text{Si se asume } R_7 = 1K\Omega$$

$$R_8 = 1.64K\Omega$$

Para poder conseguir un valor de Vo exacto se debe usar la R8 variable y el circuito se calibrara cuando ya este armado. Por lo tanto, los valores de resistencias son[Data-sheet5]:

$$R_7 = 1K\Omega \qquad R_8 = 10K\Omega \text{ variable}$$

En la Figura 4.2 se observa el circuito con los valores calculados.

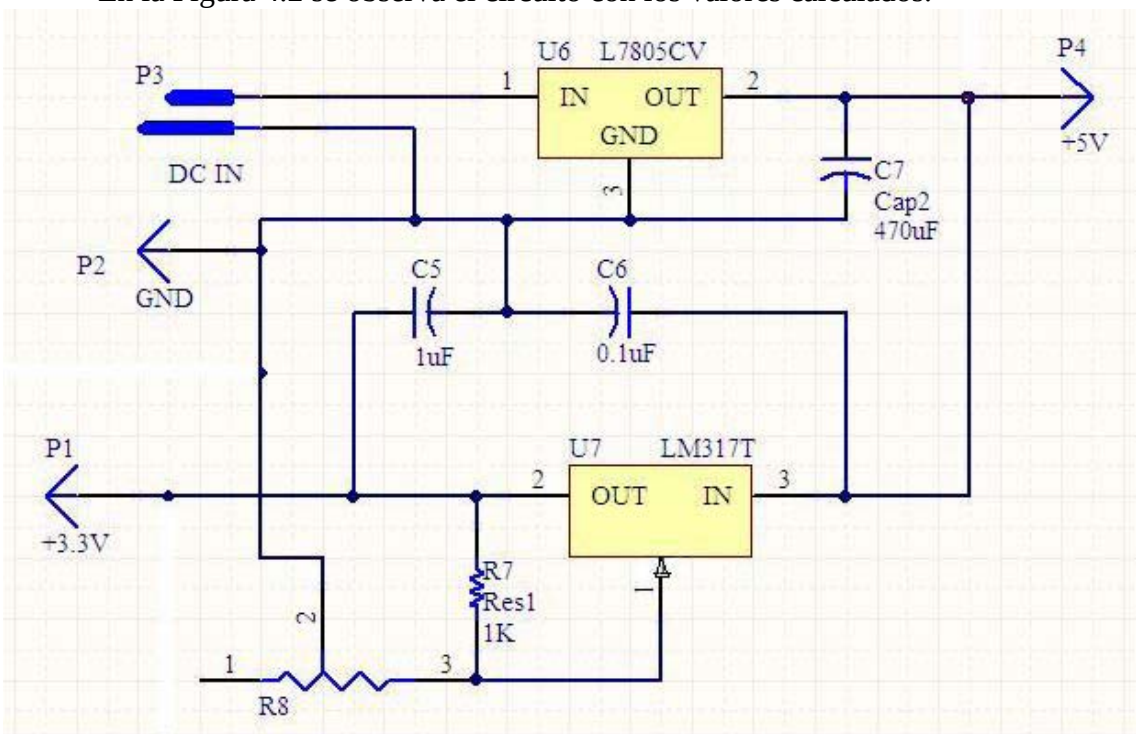


Figura 4.2. Diagrama del circuito de alimentación para 5V y 3.3V

4.2 CIRCUITO ADAPTADOR DEL MÓDULO BLUETOOTH

El módulo Bluetooth KC-21 es el elemento principal del proyecto. Es uno de los módulos más usados actualmente por su flexibilidad.

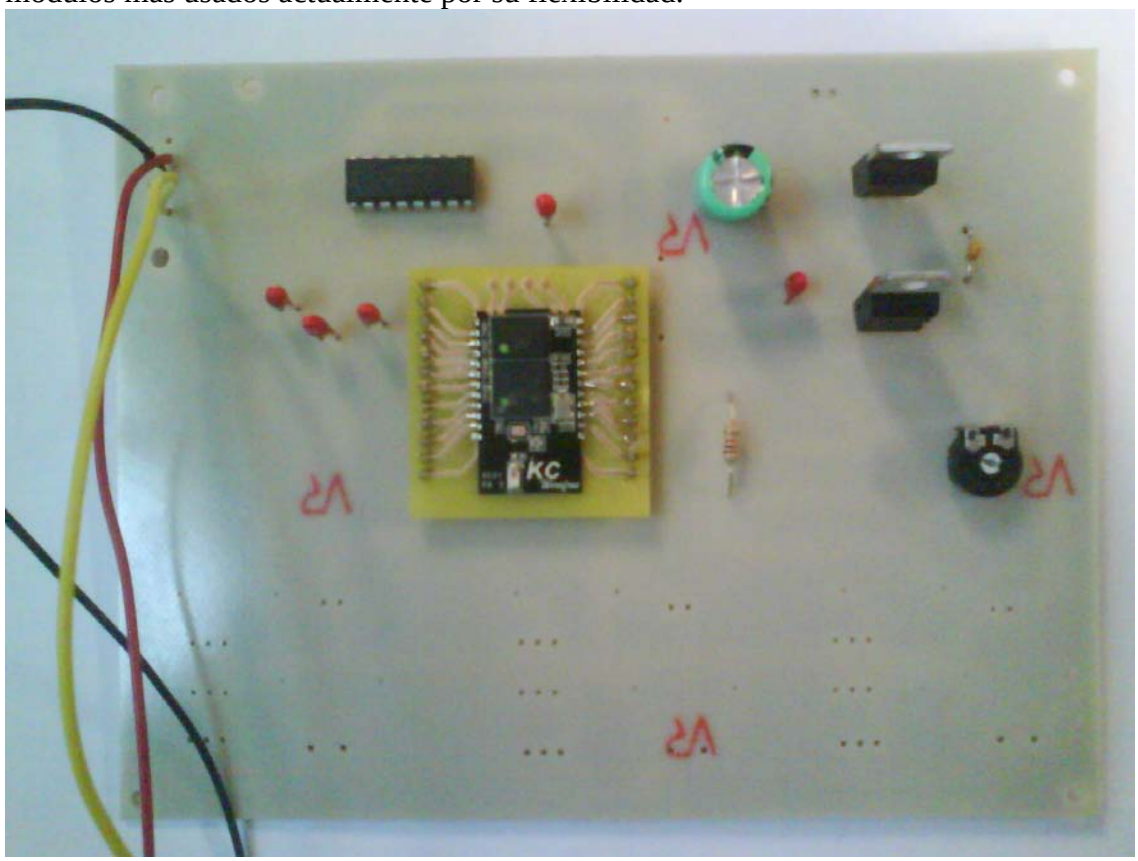


Figura 4.3. Módulo KC-21 en la placa del circuito

El módulo KC-21 incluye 14 líneas(GPIO) configurables como entradas o salidas y ofrece altas velocidades en la comunicación serie. Además, como ya se revisó en el Capítulo 2, posee las siguientes características:

a. Sistema de radio Bluetooth con licencia.
b. Perfil de Puerto Serie
c. Bluetooth clase 2
d. Estándar Bluetooth v1.2
e. Microprocesador ARM7 de 48MHz
f. 8Mb de memoria flash
g. Antena integrada al módulo.
h. Seguridad con encriptación a 128 bits
i. Distancia de hasta 20m con línea de vista.
j. Conjunto de comandos AT para configuración interna.
k. Viene en una presentación de 28 pines.

Este módulo puede ser utilizado para reemplazar conexiones de cable a través de puerto serie, equipos de adquisición de datos, control de maquinaria, sensores y monitoreo, control robótico, etc [Data-sheet1].

Dentro de este proyecto, el módulo Bluetooth KC-21 permite establecer una comunicación directa con el usuario a través del puerto serie del ordenador. Dado que para transmisión y recepción con el puerto serie del ordenador se utiliza señales RS232, mientras el KC-21 utiliza señales TTL. El integrado MAX232 transforma las señales RS232 a TTL para transmisión y recepción.

La configuración que se utiliza es la recomendada por el fabricante: C1, C2, C3, C4 tienen un valor de 1uF. Los terminales de recepción y transmisión (pin 13 y pin 14) se conectan al ordenador usando un cable con un conector DB9. En el ordenador viene el conector DB9 macho, por lo que para esta conexión el cable serie usado tendrá un conector DB9 hembra en un extremo y un conector simple de 4 pines en el otro extremo, que ira conectado al circuito. La configuración de los terminales de los conectores DB9 es la siguiente[Data-sheet4]:

La información asociada a cada uno de los pines es la siguiente:

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

En la Figura 4.4 se puede ver la identificación de pines en los conectores DB9.

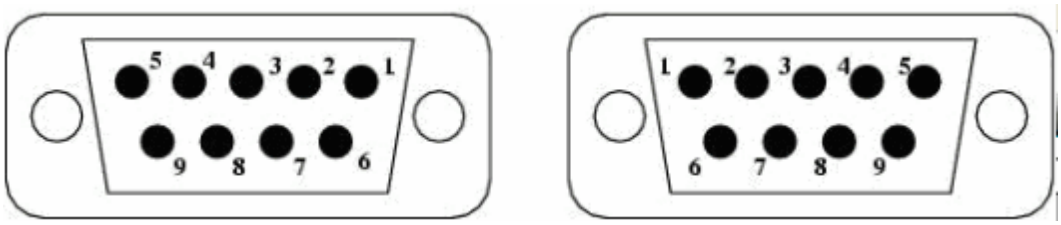


Figura 4.4. Conectores DB9 hembra y macho

4.2.1 CONFIGURACIÓN DE PINES PARA EL CABLE SERIE

El cable puede ser de cualquier tipo, la conexión de los pines es muy importante para asegurar la comunicación. De los nueve pines solamente se usaran tres: transmisión, recepción y tierra.

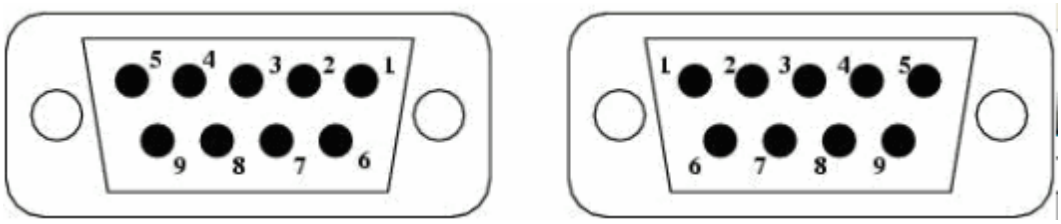


Figura 4.5. Esquema del cable serie

El chip MAX 232

Este chip permite adaptar los niveles RS232 y TTL, permitiendo conectar un PC con un microcontrolador. Sólo es necesario este chip y 4 condensadores electrolíticos. El esquema es el siguiente[Data-sheet4]:

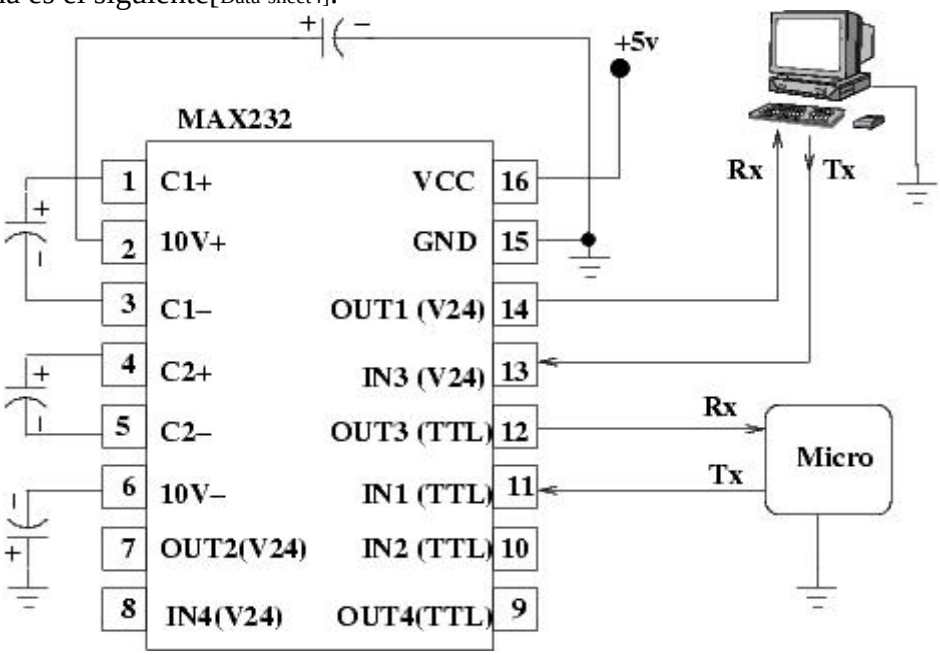


Figura 4.6. Circuito acoplador del MAX 232

Los terminales de transmisión y recepción (pin 11 y pin12) se conectan al módulo KC-21 entregándole señales TTL. De esta manera esta garantizada la comunicación entre el usuario y el módulo Bluetooth. El diagrama de la conexión entre el MAX232 y el KC-21 se puede observar en la Figura 4.7

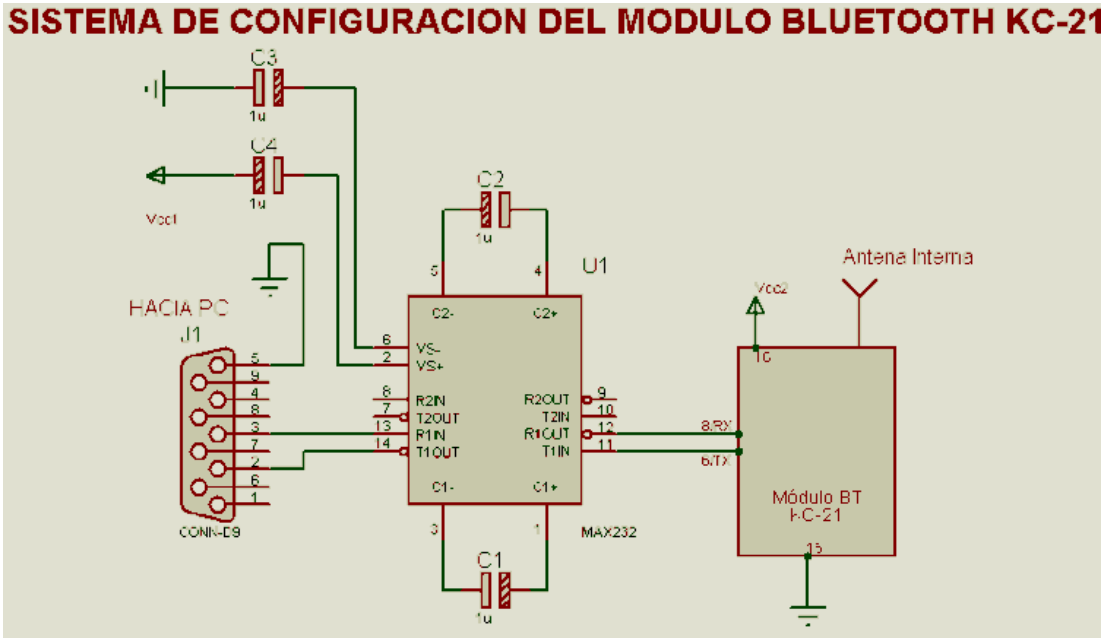


Figura 4.7. Circuito acoplador del Módulo Bluetooth

El módulo Bluetooth KC-21 utiliza para su configuración interna comandos AT, que son comandos proporcionados por el fabricante para cambiar los parámetros del módulo. Estos comandos pueden ser enviados a través del puerto serie o a través de la interfaz Bluetooth.

El módulo Bluetooth tiene asignados varios pines IO de propósito general (GPIO) que pueden ser usados como entradas o salidas. Cada uno de estos pines podrían controlar un dispositivo diferente. En este proyecto se utilizarán 3 de estos terminales como salidas: uno para control de acceso, otro para control de iluminación y un último para el ascensor. Utilizando los comandos AT se cambiarán los estados de los terminales GPIO y se emplearán estos estados para controlar los dispositivos eléctricos.

4.3 CIRCUITOS ACOPLADORES DE POTENCIA

Cada GPIO del módulo Bluetooth se conectará a un circuito acoplador de potencia diferente. Cada Circuito Acoplador está compuesto básicamente por un optotriac y un triac. El optotriac empleado en este proyecto es el MOC3021. El triac puede ser un triac genérico, de 8A a 600V, en este caso se esta usando el BTB08.

El optotriac es un triac controlado a través de la luz que le emite un diodo LED integrado. Al tener este LED encendido, el triac conduce y cuando este LED esta apagado no conduce. De esta manera se independiza la parte de control y la parte de potencia, es decir, el circuito de control solo estará conectado a un LED, que se activará con voltajes pequeños DC, y aislado del alto voltaje AC que utiliza la carga. El funcionamiento de la carga estará dado prácticamente por el estado del Optotriac.

El triac es un dispositivo semiconductor de tres pines que permite controlar el paso de corriente en dos sentidos, entre dos terminales (ánodos), usando el tercer pin (compuerta).

Sistema Bluetooth para el control de dispositivos.

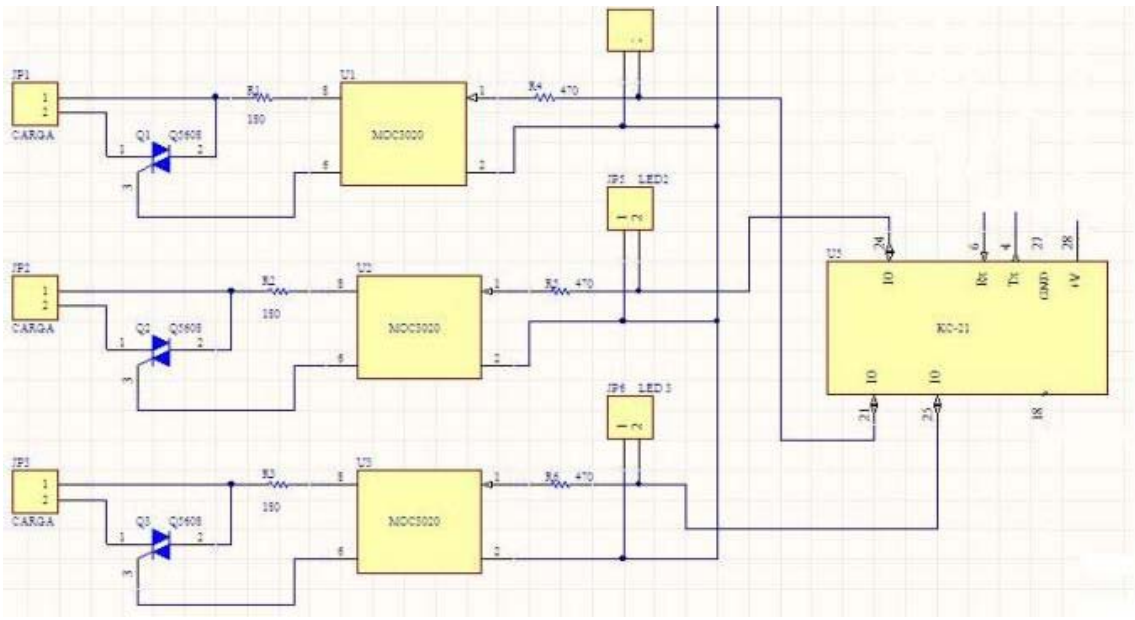


Figura 4.8. Diagrama de los circuitos de control de potencia.

Para los valores para las resistencias R1, R2 y R3, que se muestran en la Figura 4.7, utilizó la recomendación del fabricante. El valor es de 180Ω. Las resistencias R4, R5 y R6 se calcularon de la siguiente manera:

Según las especificaciones del fabricante del MOC3020, las salidas IO manejan un voltaje de VOL=0,4V para nivel bajo y VOH=2.4V para nivel alto, en ambas condiciones IOL= IOH=2mA y el máximo valor que puede tomar las I=3.1mA.

El voltaje de entrada VF de este optotriac puede estar alrededor de 1.2V típicamente, según el grafico que se encuentra en el Datasheet en el Anexo_6. Para un valor de I=2mA se tendrá un VF=1,07 V y para un valor de I=3.1mA se tendrá un VF=1,09 V. Por lo tanto, para ambas condiciones se tiene:

$$I = 2mA \quad V_F \approx 1.07V$$

$$R = \frac{2.4V - 1.07V}{2mA} = 0.665K\Omega$$

$$I = 3.1mA \quad V_F \approx 1.09V$$

$$R = \frac{2.4V - 1.09V}{3.1mA} = 0.422K\Omega$$

Entonces los valores de resistencia a la entrada del MOC3020 deben estar entre 422Ω y 665Ω. Se escogió R4=R5=R6=470 Ω, como también se muestra en la Figura 4.8

Teniendo ya listo el Software del móvil (la aplicación Java Micro Edition) y la interfaz de Hardware (el módulo Bluetooth adaptado a los circuitos acopladores de potencia), el sistema de control está completo. En el siguiente capítulo se explicará su

funcionamiento usando un teléfono móvil para controlar simultáneamente una lámpara y una cerradura eléctrica y un ascensor.

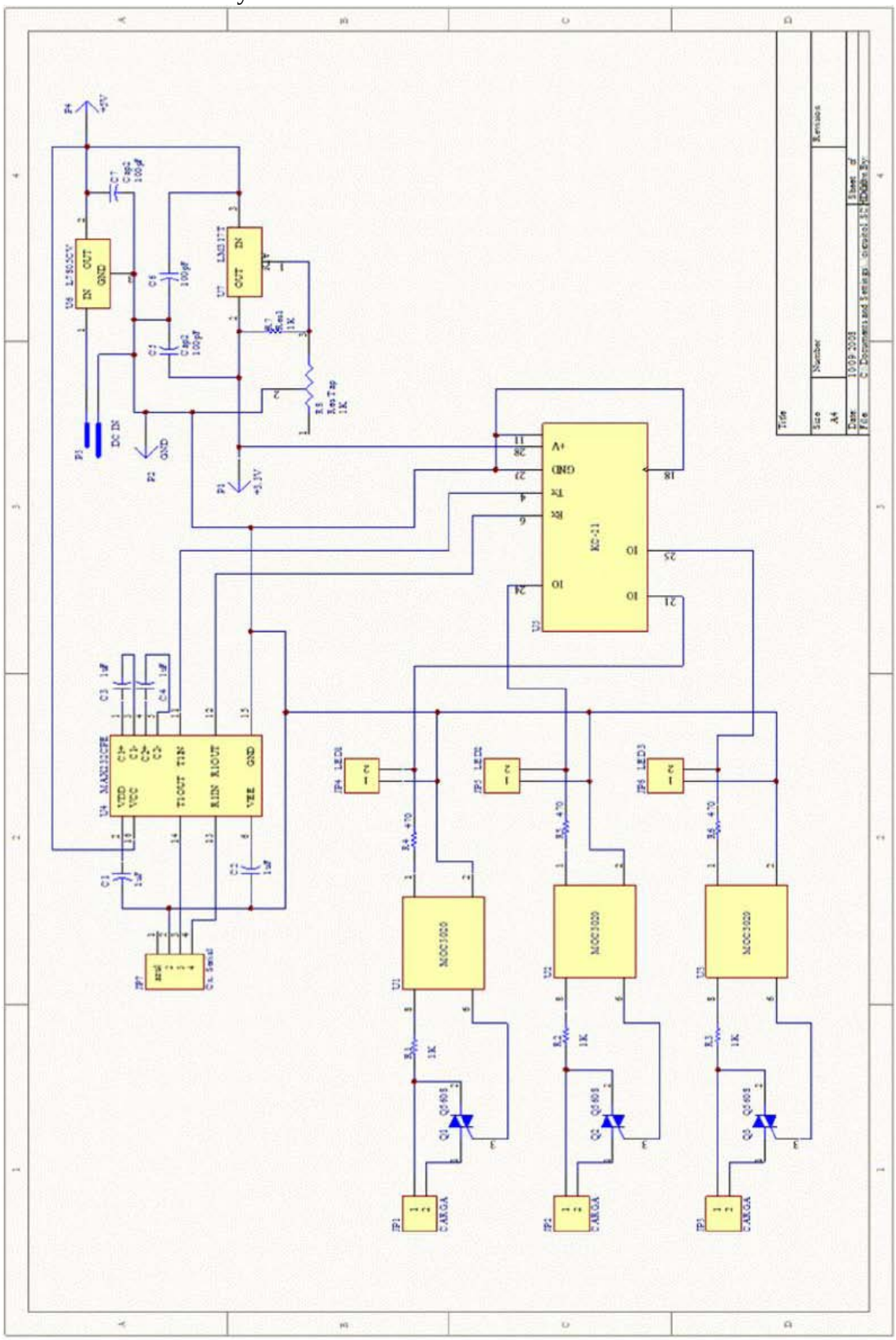


Figura 4.9. Diagrama del circuito completo

CAPÍTULO 5. CONFIGURACIÓN DEL SISTEMA

5.1 CONFIGURACIÓN DEL MÓDULO

5.1.1 Configuración Inicial

La configuración inicial del Módulo KC-21 consiste en establecer los parámetros de comunicación Puerto Serie del Hyperterminal de Windows para poder conectarse a éste, y de este modo configurar un nombre con el cual sea identificado en una búsqueda de dispositivos

Para realizar la conexión con el KC-21 usando el puerto serie desde un ordenador se debe tener en cuenta que sus parámetros de conexión deben ser los mismos que en el Hyperterminal. Estos parámetros, por defecto, son: velocidad 115200, 1 bit de parada, no paridad, 8 bits de datos, control de flujo habilitado por hardware.

Una vez hecha la conexión, se puede configurar el módulo usando códigos AT que el fabricante provee y que están detallados en el Anexo_3. Así por ejemplo, se cambió la velocidad de transmisión de la conexión SPP escribiendo la siguiente línea:

AT+ZV ChangeBaud 9600

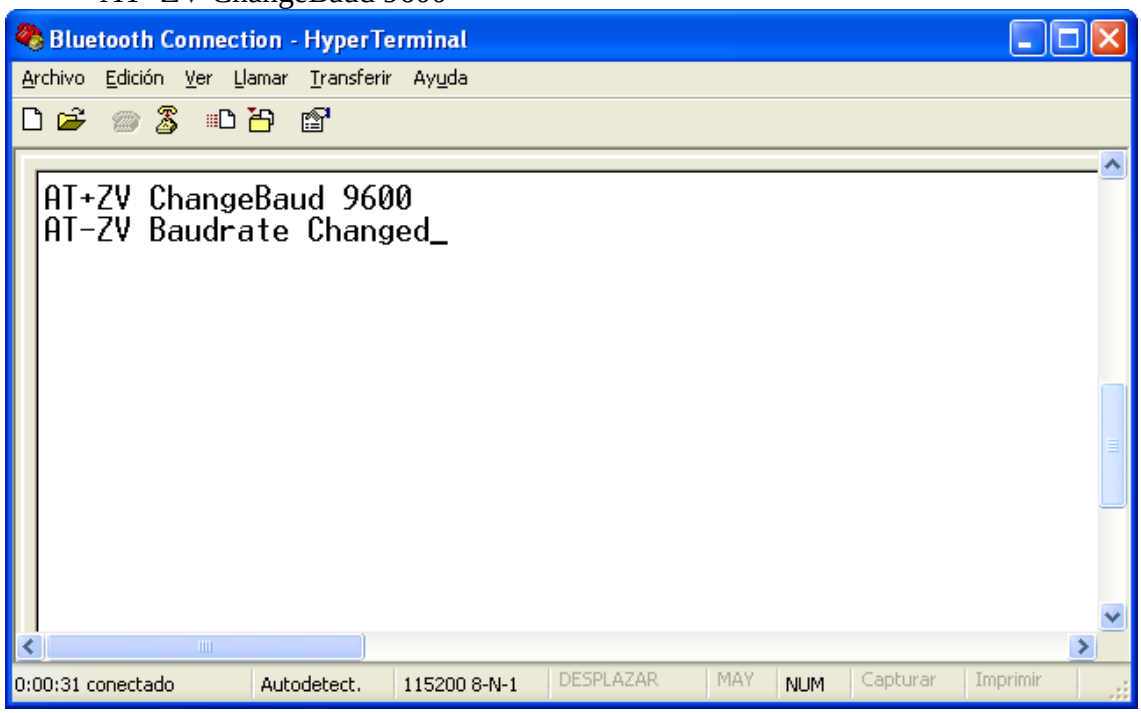


Figura 5.1 Velocidad de transmisión

Desde ese momento, la velocidad de transmisión SPP del módulo KC-21 cambia y a partir de ahora es de 9600 bps[Data-sheet2].

En el futuro, para la comunicación SPP se deberá usar la configuración del Hyperterminal con los valores predeterminados, como se ve en la Figura 5.2.

Figura 5.2 Propiedades de puerto serie

Para cambiar el nombre del dispositivo o friendlyName del módulo KC-21, se usó la siguiente línea:

AT+ZV DefaultLocalName BT_2010

La característica de recibir comandos remotos, llamada Remote Command Mode viene incluida en el módulo KC-21 y se la activa con el siguiente comando:

AT+ZV RemoteCommand e

Como se revisó en el Capítulo 4, las órdenes o comandos que se envían desde el cliente hasta el módulo, son cadenas de texto, para el módulo KC-21 son comandos AT similares a los que se usaron para cambiar la velocidad o para cambiar su nombre. Por lo tanto, una vez hecha la comunicación Bluetooth, también se puede enviar cualquier comando por este medio de la misma manera que se hace por el puerto serie[Data-sheet2].

Estas configuraciones no se borran aunque el módulo sea reiniciado, por lo tanto, cuando un dispositivo móvil haga una búsqueda el módulo aparecerá como “BT_2010”; la conexión SPP se realizará con los valores predeterminados del Hyperterminal y estará habilitado la recepción de comandos AT a través de una conexión Bluetooth.

5.1.2 Configuración de los terminales del módulo

Para realizar el control de los dispositivos se hizo uso de los llamados Terminales de Propósito General (General Purpose Input-Output), con que cuenta este módulo Bluetooth. Se están usando tres de los terminales disponibles: GPIO05, GPIO07 y GPIO10.

Se puede usar cualquier terminal GPIO, pero se escogieron éstos porque la posición que tienen en el módulo facilitó el diseño del circuito.

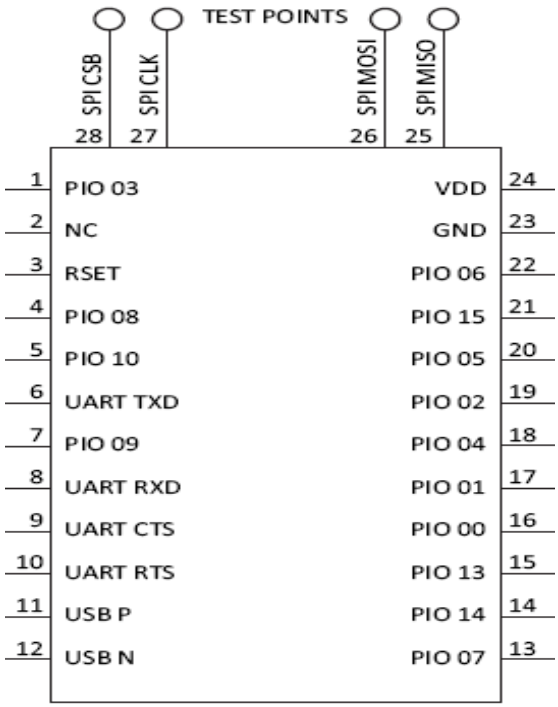


Figura 5.3 Distribución de pines del módulo KC-21

Los GPIOs pueden ser usados como entradas o como salidas, cuando están configurados como salidas pueden ser configurados con nivel alto o nivel bajo.

Las condiciones iniciales predefinidas de los GPIOs evitarán conflictos en el caso de un eventual reseteo del circuito, ya que de esta manera se asegura un estado fijo y predefinido al iniciar.

Las condiciones de los GPIOs se muestran en la Tabla 5.1.

Pin	Función	Tipo	Descripción
1	PIO_03	I/O	Programmable Input/Output
4	PIO_08	I/O	Programmable Input/Output
5	PIO_10	I/O	Programmable Input/Output[Class 1 TX Enable]
7	PIO_09	I/O	Programmable Input/Output[Class 1 TX Enable]
13	PIO_07	I/O	Programmable Input/Output [ADC, CLK]
14	PIO_14	I/O	Programmable Input/Output [ADC, CLK]
15	PIO_13	I/O	Programmable Input/Output
16	PIO_00	I/O	Programmable Input/Output [Connection Indicator]
17	PIO_01	I/O	Programmable Input/Output [Power Indicator]
18	PIO_04	I/O	Programmable Input/Output [Activity Indicator]
19	PIO_02	I/O	Programmable Input/Output [RXD Passthrough]
20	PIO_05	I/O	Programmable Input/Output [RXD Passthrough]
21	PIO_15	I/O	Programmable Input/Output [TXD Passthrough]
22	PIO_06	I/O	Programmable Input/Output [RTS Passthrough]

Tabla 5.1 Condiciones iniciales de los GPIOs

En este caso los GPIOs serán configurados como salidas, de este modo se pueden establecer en estado alto o bajo y así controlar a los dispositivos conectados[Data-sheet1].

El comando GPIOConfig es usado para configurar el terminal como entrada o salida.

Sintaxis: AT+ZV GPIOConfig [GPIO 0-15]["i" para entrada y "o" para salida]

Respuesta Terminal: AT-ZV GPIOConfigDone

En este caso, para tener los terminales configurados como salidas, se enviarán desde el teléfono móvil los comandos de la siguiente manera:

AT+ZV GPIOConfig 05 o
AT+ZV GPIOConfig 07 o
AT+ZV GPIOConfig 10 o

Una vez configurado el terminal, con el comando GPIOWrite se puede cambiar el estado de un terminal a alto o bajo según la necesidad[Data-sheet3].

Sintaxis AT+ZV GPIOWrite [GPIO 0-15] [Setting "1" para alto o "0" para bajo]

Respuesta Terminal AT-ZV GPIOWriteDone

Con estos comandos, se pueden controlar los dispositivos eléctricos conectados al módulo de control como se verá más adelante en las pruebas realizadas.

5.2 INSTALACIÓN FÍSICA

Este proyecto puede ser usado con cualquier dispositivo eléctrico. Para el caso del Control de Iluminación la lámpara debe poder ser encendida manualmente o vía Bluetooth. Para ello se recomienda usar un circuito eléctrico conmutado.

En la Figura 5.4, se muestra el diagrama de conexión recomendada con dos conmutadores.

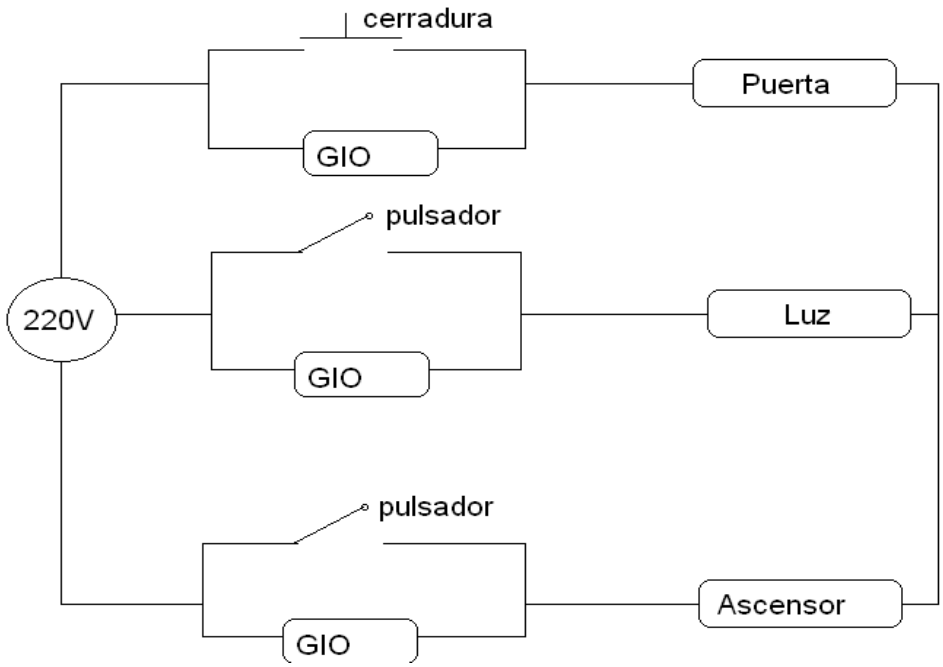


Figura 5.4 Diagrama de conexión.

Para el caso de la cerradura electrónica, la conexión del circuito de control Bluetooth actuará como el pulsador del portero automático que se usa generalmente con una cerradura eléctrica, para el caso de la luz el relé activado por Bluetooth actuará como el pulsador para encender la luz y lo mismo para el caso del pulsador del ascensor.

5.2.1 ALIMENTACIÓN

Para conseguir el voltaje de alimentación se usará un adaptador regulado a 6V, aunque se podría usar un voltaje de hasta 12V. El proyecto puede estar instalado en cualquier rincón de la casa y desde allí se conectará al dispositivo a controlar. Los dispositivos eléctricos deben funcionar con 220V.

Es muy importante que el Módulo Bluetooth siempre esté encendido para que pueda recibir las peticiones de parte de los clientes y sea capaz de realizar las funciones de control. Lo más aconsejable es usar un UPS, más aún si los dispositivos que se controlen también necesiten protección.

5.3 DETALLE DE COSTOS

Cantidad	Elemento	Precio (€)
1	Módulo Kc-21	24€
1	Zócalo	2€
3	Triacs	1.5€
3	Optotriacs	2.4€
1	Acoplador Max232	1.2€
12	Condensadores y resistencias	5€
3	Leds	1€
2	Gastos de envío	70€
10(horas)	Diseño de la placa(20€/hora)	200€
	Subtotal	307.1€
100(horas)	Investigación y elaboración(20€/hora)	2000€
	TOTAL	2301.7€

Tabla 5.2 Coste del proyecto

Una vez concluido el proyecto, se ha realizado un resumen referencial de los gastos.

El Módulo KC-21 es el elemento más importante y más costoso ya que ha sido enviado desde EE.UU. y hay que tener en cuenta los gastos de envío.

Los circuitos acopladores, que incluyen principalmente un triac y un optotriac, se usan uno por cada GPIO en uso. En el caso de usar más terminales GPIO los costos también aumentarán; pero a cambio se tendrán más dispositivos eléctricos controlados.

También se ha valorado el tiempo empleado en el diseño de la placa con el programa Pcad 2006. Así como el tiempo destinado a la investigación, documentación e implementación de todas las aplicaciones necesarias par su funcionamiento.

5.4 PRUEBAS Y RESULTADOS DEL SISTEMA

5.4.1 CONTROL DE DISPOSITIVOS VÍA CONEXIÓN SPP

Como ya se mencionó, usando el puerto serie también se pueden enviar los comandos al módulo de control. Se ha procedido a realizar esta conexión. La comunicación se realizó normalmente y el envío de comandos produjo los resultados esperados: la respuesta fue casi instantánea.

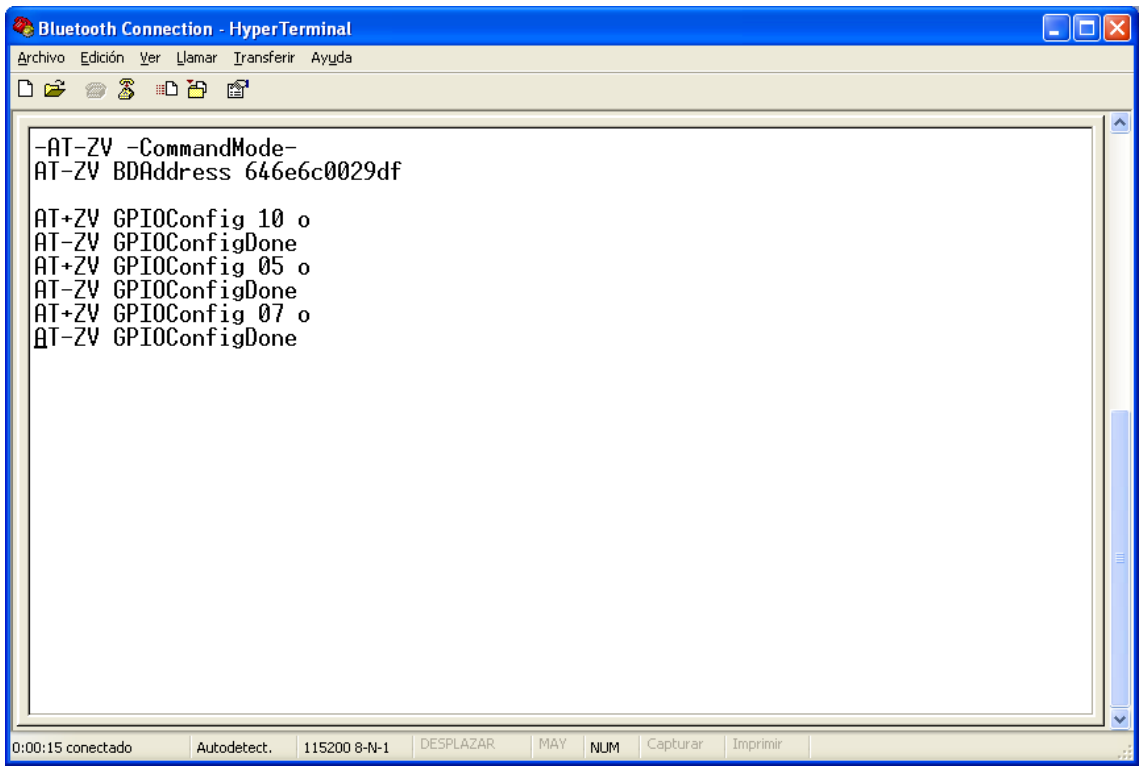


Figura 5.5 Pantalla del Hyperterminal que muestra comandos AT

En la Figura 5.5 se observa la pantalla del Hyperterminal con los comandos enviados vía puerto serie y sus respectivas respuestas. Los comandos mostrados corresponden a los que se usan para el control del módulo Bluetooth.

5.4.2 CONTROL DE DISPOSITIVOS VÍA CONEXIÓN BLUETOOTH

5.4.2.1 Pruebas con el móvil Sony- Ericsson Z 750i

Para realizar las pruebas de funcionamiento se usó el teléfono móvil Sony- Ericsson Z 750i, un móvil de gama media que cuenta con las características necesarias. En la Figura 5.5 se puede observar el modelo. Para administración de archivos e instalación de aplicaciones, utilizo una conexión Bluetooth con el ordenador. Cuando se tiene una aplicación, se conecta el teléfono con ordenador, y se trasfiere el archivo *BT_2010.JAR*.

Figura 5.6 Móvil Sony- Ericsson Z 750i

Una vez se transmite la aplicación en el teléfono, aparece con el nombre BT_2010. Se abre la aplicación se instala y se comienza a utilizar.

Cuando se abre la aplicación BT_2010 en el móvil, la aplicación pide confirmar la activación de Bluetooth. Una vez autorizado, aparece en la pantalla principal la opción “Buscar”. Cuando se selecciona esta opción, automáticamente el teléfono empezó una búsqueda a su alrededor y puede encontrar al Módulo Bluetooth, este procedimiento es transparente para el usuario. El proceso tarda alrededor de 8s hasta que se completa la búsqueda y se pueden mostrar los dispositivos encontrados. La lista de dispositivos encontrados se muestra por pantalla, al ser una lista implícita, se pueden escoger, al escoger el dispositivo “BT_2010” aparecen las opciones:

"Seleccionar todas"
“Abrir puerta"
"Llamar ascensor"
"Encender luz"

La primera activa las tres opciones de manera consecutiva, y las otras de la lista activan solo la opción a la que hace referencia.

Al seleccionar “Abrir puerta" se envía el comando:

“Abrir puerta"	AT+ZV GPIOWrite 05 1

Al seleccionar "Llamar ascensor" se envía el comando:

"Llamar ascensor"	AT+ZV GPIOWrite 07 1

Al seleccionar "Encender luz" se envía el comando:

"Encender luz"	AT+ZV GPIOWrite 10 1

5.4.3 LISTADO DE BÚSQUEDA

En este formulario aparece la lista con todos los dispositivos encontrados en la búsqueda realizada, incluido “BT_2010”, el módulo de control. Si en la lista no aparece, no se puede acceder al menú de control. Es necesario seleccionar en el menú la opción que permita hacer una nueva búsqueda hasta que aparezca en la lista y poder continuar.

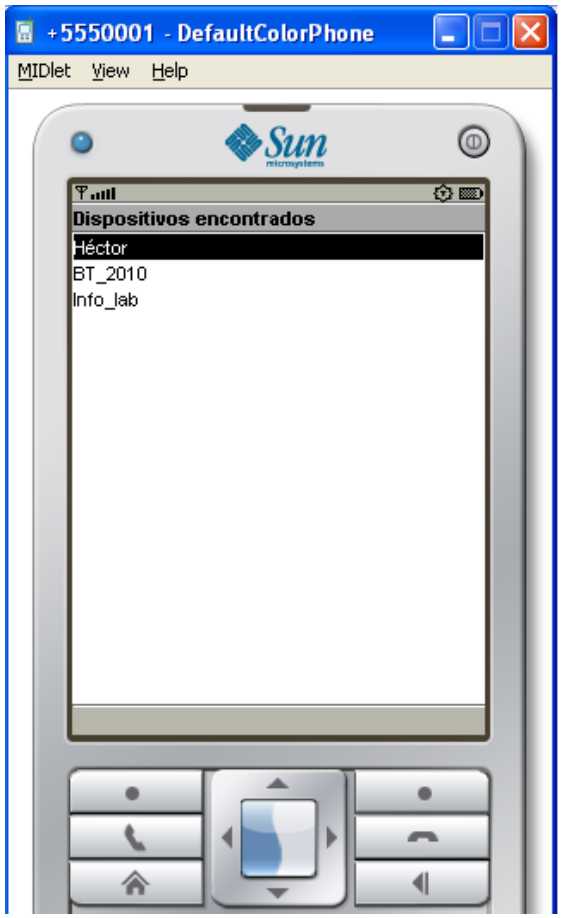


Figura 5.7 Lista de dispositivos encontrados

Se pueden enviar los comandos, las veces que se desee, inclusive alternando entre Control de Acceso y Control de Iluminación.

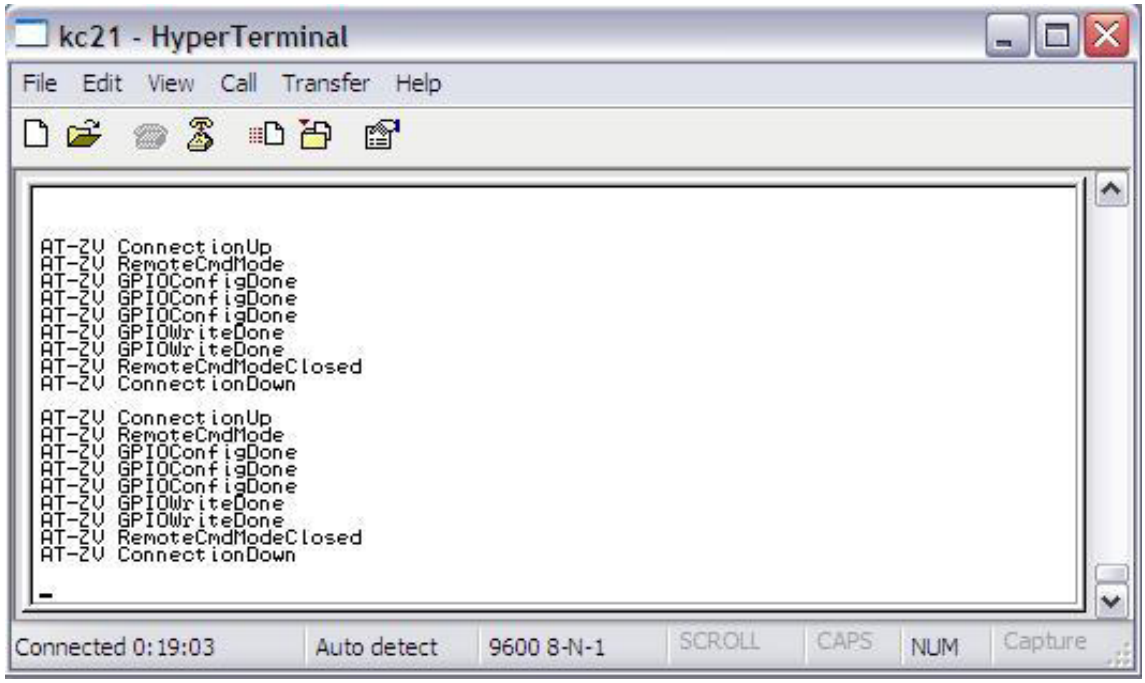


Figura 5.8 Respuesta a comandos ejecutados correctamente

Se puede usar más de un teléfono móvil a la vez. Los comandos se ejecutaron en el mismo orden en que se enviaron, respetando el tiempo de respuesta señalado anteriormente.

Al hacer pruebas se ha aumentado progresivamente la distancia del dispositivo móvil al Módulo de Control sin presentar alguna variación considerable en los tiempos de búsqueda y de envío. Se pudo alcanzar hasta 10m sin obstáculos. Si la distancia aumenta la búsqueda se ralentiza y muchas veces el módulo de control, “BT_2010”, no se detecta.

También se ha hecho pruebas de conexión con una puerta de metal en medio, la distancia máxima menor de 8 m, dentro de los cuales el módulo respondió con normalidad.

Usando como obstáculo una pared común de ladrillo se obtuvo los mismos resultados, una distancia máxima para la transmisión está alrededor de 8 m. Se concluye obviamente que a mayor distancia menor efectividad en la transmisión por lo que la distancia máxima de transmisión disminuye si hay obstáculos.

Teniendo en cuenta que la frecuencia de trabajo de Bluetooth, 2,4GHz, es también la que utilizan otros aparatos comunes para una casa como teléfonos inalámbricos domésticos, microondas, equipos Wireless, WiFi, etc. Se ha probado en situaciones similares a las que se pueden dar en un portal, con varios móviles y redes WiFi y los resultados fueron similares a los obtenidos en pruebas interiores, esto es, no se observaron fallos por interferencia.

CAPÍTULO 6. CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

- Este proyecto posibilita dar otro uso a la tecnología Bluetooth integrada en los teléfonos móviles. Por otro lado demuestra que la Plataforma J2ME que incorporan los móviles se puede utilizar para controlar un sistema Bluetooth.

- Se ha conseguido desarrollar un sistema Bluetooth sencillo y a la vez muy útil, mucho más para personas con algún tipo de discapacidad o que por algún motivo no pueden utilizar los controles convencionales dentro de una vivienda.

- Actualmente existen muchas tecnologías para control remoto tanto a nivel doméstico como industrial, pero están diseñadas para dar servicio o respuesta a necesidades concretas y se han desarrollado de manera. La tecnología Bluetooth es un ejemplo del avance acelerado en este campo para integrar las telecomunicaciones globalmente y proveer acceso a más personas cada día, es decir una misma tecnología puede dar respuesta a muchas necesidades en un único dispositivo, el teléfono móvil.

- La tecnología Bluetooth es muy habitual entre los teléfonos móviles, hoy en día todos los modelos la incluyen por lo que el campo de aplicación del proyecto sigue creciendo. Por esta razón, en comparación con otras tecnologías inalámbricas usadas en Domótica, como ZigBee e incluso WiFi, se puede concluir que el utilizar un teléfono móvil como dispositivo de control es muy ventajoso, puesto que ya tiene incluida esta tecnología.

- La Máquina Virtual Java viene incluida en los teléfonos móviles desde su fabricación. Ésta debe contar con soporte para Bluetooth (JSR-82) para que la aplicación pueda ejecutarse sin problemas. Dado que cada vez más móviles incorporan esta tecnología, se puede concluir que, cada día más usuarios pueden utilizar este tipo de proyectos.

- La colocación del módulo se puede realizar de manera que no afecte a las instalaciones eléctricas convencionales, por su tamaño reducido. Gracias a sus funciones se puede manipular los dispositivos eléctricos de la vivienda simultáneamente tanto manualmente como vía Bluetooth. Por lo tanto, se puede adaptar cualquier dispositivo tradicional para que sea controlado desde un teléfono móvil.

- La comunicación implementada en este proyecto se realiza de manera simplificada, en una sola vía. El teléfono no recibe respuestas de parte del módulo Bluetooth sobre el estado de los dispositivos eléctricos o del cumplimiento de la orden enviada. Solamente se puede ver la respuesta a un comando viendo si la orden es atendida observando los dispositivos eléctricos y su funcionamiento

- El módulo de control solo puede controlar dispositivos en dos estados, generalmente encendido y apagado, esto limita el tipo de dispositivos eléctricos que puedan ser controlados.

- Las condiciones iniciales de los GPIOs evitan tener conflictos al reiniciar el Módulo Bluetooth, ya que aseguran un estado único y fijo. Se concluye que en caso de un eventual reinicio, el módulo siempre volverá a un estado predefinido y conocido por el usuario, de esta manera se pueden tomar las debidas precauciones.

- El alcance máximo, según el fabricante, es de 20m y la distancia máxima para un radio Clase 2, según la especificación de Bluetooth, es de aproximadamente 10m. Se ha probado que la distancia real está alrededor de los 10m en condiciones normales y hasta 7m atravesando grandes objetos. Por lo tanto se puede concluir que esta aplicación servirá para las acciones que se explican en el proyecto.

- La frecuencia de trabajo del sistema es 2,4GHz, la misma que es usada por microonda, teléfonos inalámbricos domésticos y sistemas IEEE802.11. Se pudo comprobar que la tecnología Bluetooth no se ve afectada por las interferencias con esas otras tecnologías, debido al sistema mejorado de salto de frecuencia, AFH.

- Respecto a la seguridad de la que disponemos en este caso no es muy alta ya que el modulo Bluetooth esta visible a cualquier usuario que rastree con su móvil cerca del mismo, y con ciertos conocimientos sería sencillo activar los dispositivos. El modulo Kc21 incorpora posibilidades de aumentar la seguridad pero se pierde comodidad para el usuario.

- El funcionamiento del sistema es satisfactorio y se han cumplido los objetivos del proyecto. Por lo tanto, se puede concluir, que Bluetooth es una buena alternativa para este tipo de aplicaciones por su capacidad de funcionamiento en equipos móviles de reducido tamaño y bajo consumo de potencia, por su inmunidad al ruido, su estabilidad y seguridad.

6.2 RECOMENDACIONES Y LINEAS FUTURAS

Sistema Bluetooth para el control de dispositivos.

- Se recomienda el uso del lenguaje Java, para la creación de aplicaciones para móviles, por la versatilidad y la gran cantidad de equipos que lo soportan.

- Este proyecto puede ser referencia y guía para quienes deseen continuar desarrollando este tipo de aplicaciones. Por ejemplo, se puede mejorar este sistema para que funcione como un daemon; o hacer alguna modificación para que pueda manejar más niveles de control.

- El módulo usado tiene un sistema de radio Bluetooth de Clase 2, diseñado para ser usado a distancias máximas de 10 m, para aplicaciones de control que necesiten una distancia mayor se recomienda utilizar un módulo que trabaje con un radio de clase 1, este cubrirá distancias de hasta 100 m.

- Para el diseño de sistemas Bluetooth se recomienda utilizar módulos que cuenten con una interfaz SPP, de esta manera se podrán ver los datos recibidos usando el Hyperterminal.

- Se recomienda instalar el módulo de control en un lugar centrado en el portal para que pueda cubrir un área lo más grande posible, para ser detectado conforme nos acerquemos al portal.

- Para realizar aplicaciones para Palm, WM, o Android que utilicen la tecnología Bluetooth con que cuentan estos dispositivos, se recomienda utilizar otras herramientas basadas en Microsoft Visual C#, .Net, o Android.

- El Sistema Operativo en cada marca de teléfono móvil puede ser diferente por lo que la presentación de la aplicación en uno o en otro modelo también puede cambiar, se podría modificar la aplicación para un modelo en particular tomando en cuenta el tamaño de su pantalla, su sistema operativo, disposición de botones, y otros recursos propios con que cuente realizando llamadas al sistema para adaptarse a cada caso.

- Para la construcción del circuito impreso, se recomienda ubicar los elementos de manera que se puedan evitar daños en caso de golpes o mal uso de los mismos, así como también cuidar el proceso de ensamblaje para evitar contactos abiertos o cortocircuitos.

- El módulo kc21 podría servir para muchas otras aplicaciones como por ejemplo en la domótica ya que al incorporar 14 pines I/O que pueden configurarse tanto como entradas como salidas de modo que algunos pueden conectarse a sensores y el resto a relees para domotizar una vivienda o un local y tener acceso a las lecturas de los sensores y la los relees desde el móvil.

- Sobre este mismo proyecto se pueden realizar pequeñas modificaciones que nos permitan obtener otras aplicaciones, como por ejemplo configurando el dispositivo Bluetooth como non-discoverable para garantizar la seguridad y poder utilizarlo para locales comerciales o en viviendas unifamiliares.

- En definitiva el Bluetooth y J2ME se pueden utilizar para cualquier comunicación diaria y rutinaria en distancias cortas, como podría ser lecturas de contadores, puntos de información, control de productos etc.

- En un futuro el móvil podría convertirse en un mando a distancia universal en el ámbito de la vivienda, siempre y cuando todos los electrodomésticos incorporen un dispositivo Bluetooth.

BIBLIOGRAFÍA

LIBROS:

[1] "BLUETOOTH FOR JAVA" de Bruce Hopkins and Ranjith Antony

[2] "JAVA 2. MANUAL DE USUARIO Y TUTORIAL" de Agustín Froufe.

[3] "JUEGOS EN JAVA" de Joel Fan/Eric Ries/Calin Tenitchi

[4] "J2ME. MANUAL DE USUARIO Y TUTORIAL" de Froufe, A/Jorge, P.

[5] "WIRELESS JAVA WITH J2ME" de Michael Morrison.

[6] "PROGRAMACIÓN DE JUEGOS PARA MÓVILES CON J2ME" de Alberto García Serrano,

[7] "JAVA A TOPE: J2ME (JAVA 2 MICRO EDITION)" de Sergio Gálvez Rojas y Lucas Ortega Díaz

REFERENCIA ELECTRÓNICA:

[I] "JAVA APIS FOR BLUETOOTH WIRELESS TECHNOLOGY", en http://www.j2medev.com/api/btapi/index.html

[II] "LISTADO DE EQUIPOS QUE CUMPLEN CON LAS CARACTERÍSTICAS JAVABLUETOOTH" en http://java.sun.com/j2me/

[III] "EL ESTÁNDAR DE COMUNICACIONES POR INFRARROJOS IRDA" en http://www.monografias.com/trabajos24/estandar-comunicaciones-irda/estandarcomunicaciones-irda.shtml

[IV] "DESCRIPCIÓN GENERAL DEL FUNCIONAMIENTO" en http://www.bluetooth.com/Spanish/Technology/Pages/default.aspx

[V] "BLUETOOTH ADAPTIVE FREQUENCY HOPPING AND SCHEDULING" en http://www.antd.nist.gov/pubs/golmie_milcom03.pdf

[VI] "BANDA BASE BLUETOOTH" en http://www.freebsd.org/doc/es_ES.ISO8859-1/books/handbook/network-bluetooth.html

[VII] "COMPARE CON OTRAS TECNOLOGÍAS" en http://www.bluetooth.com/Spanish/Technology/Works/Pages/Compare.aspx

[VIII] "ARTÍCULOS SOBRE J2ME" http://developers.sun.com/techtopics/mobility/midp/reference/techart/index.html

[IX] "Códigos J2ME de ejemplo" en http://developers.sun.com/techtopics/mobility/midp/samples/

[X] "NOTICIAS RELATIVAS A J2ME" en http://www.developer.com/java/j2me/

[XI] "PÁGINA SOBRE JUEGOS PARA MÓVILES" en http://www.midlet-eview.com/

[XII] "TUTORIALES Y ARTÍCULOS SOBRE J2ME" en http://www.palowireless.com/java/tutorials.asp

[XIII] "SUN DELIVERS NEXT VERSION OF THE JAVA PLATFORM" y "BUILDING AND STRENGTHENING THE JAVA BRAND" en http://www.sun.com

DATA SHEETS

[Data-sheet1] KC-21
[Data-sheet 2] kcSerial_Getting Started Guide
[Data-sheet 3] kcSerial_RefGuide.pdf
[Data-sheet 4] MAX232 Data sheet
[Data-sheet 5] BTB08 Data sheet
[Data-sheet 6] L7805 Data sheet
[Data-sheet 7] MOC3020 Datasheet
[Data-sheet 8] LM317 Data sheet

SISTEMA BLUETOOTH PARA EL CONTROL DE DISPOSITIVOS

Héctor Martínez Buldáin

Bluetooth

COMPARACIÓN CON OTRAS TECNOLOGÍAS

- Bluetooth e Infrarrojo

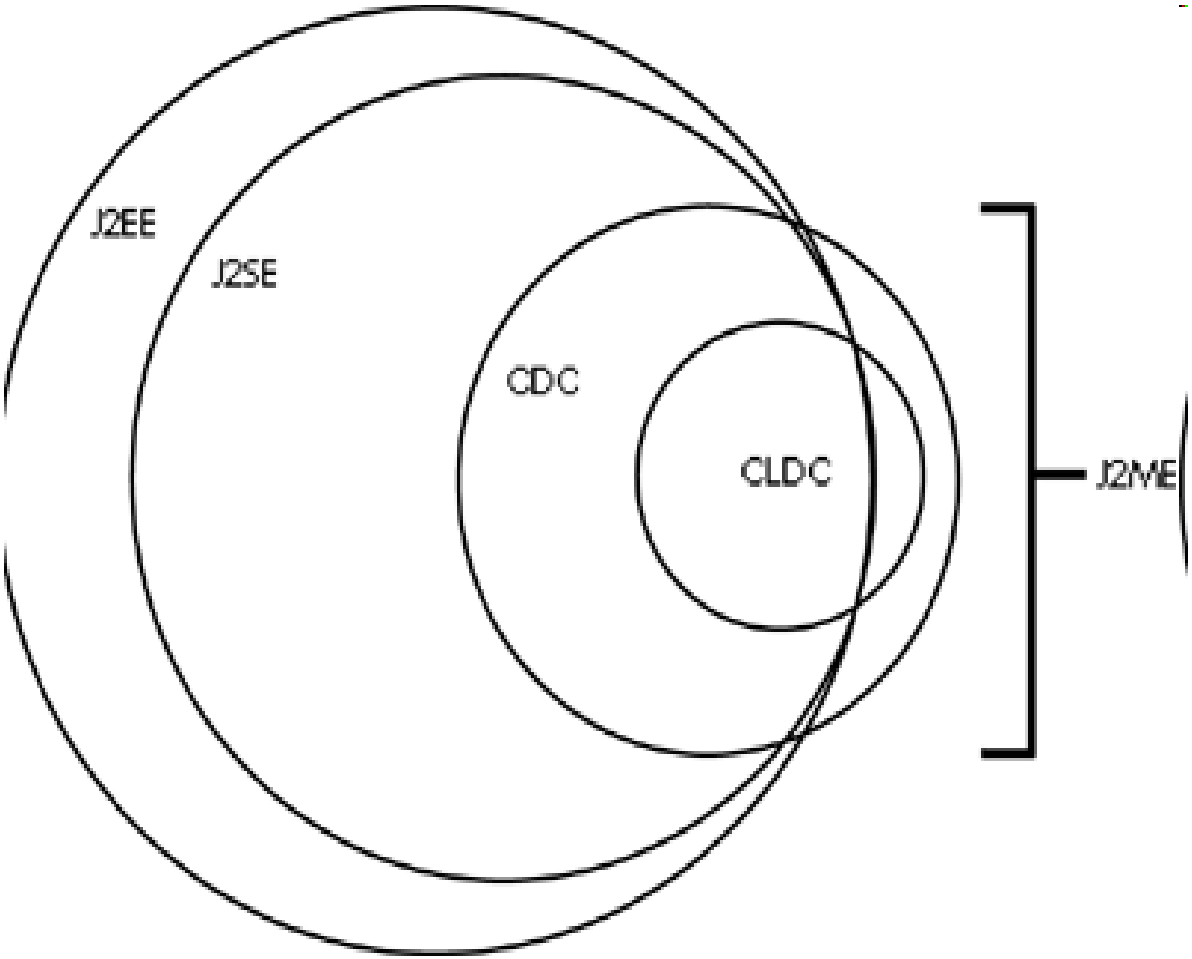
- Bluetooth y WiFi

- Bluetooth y ZigBee

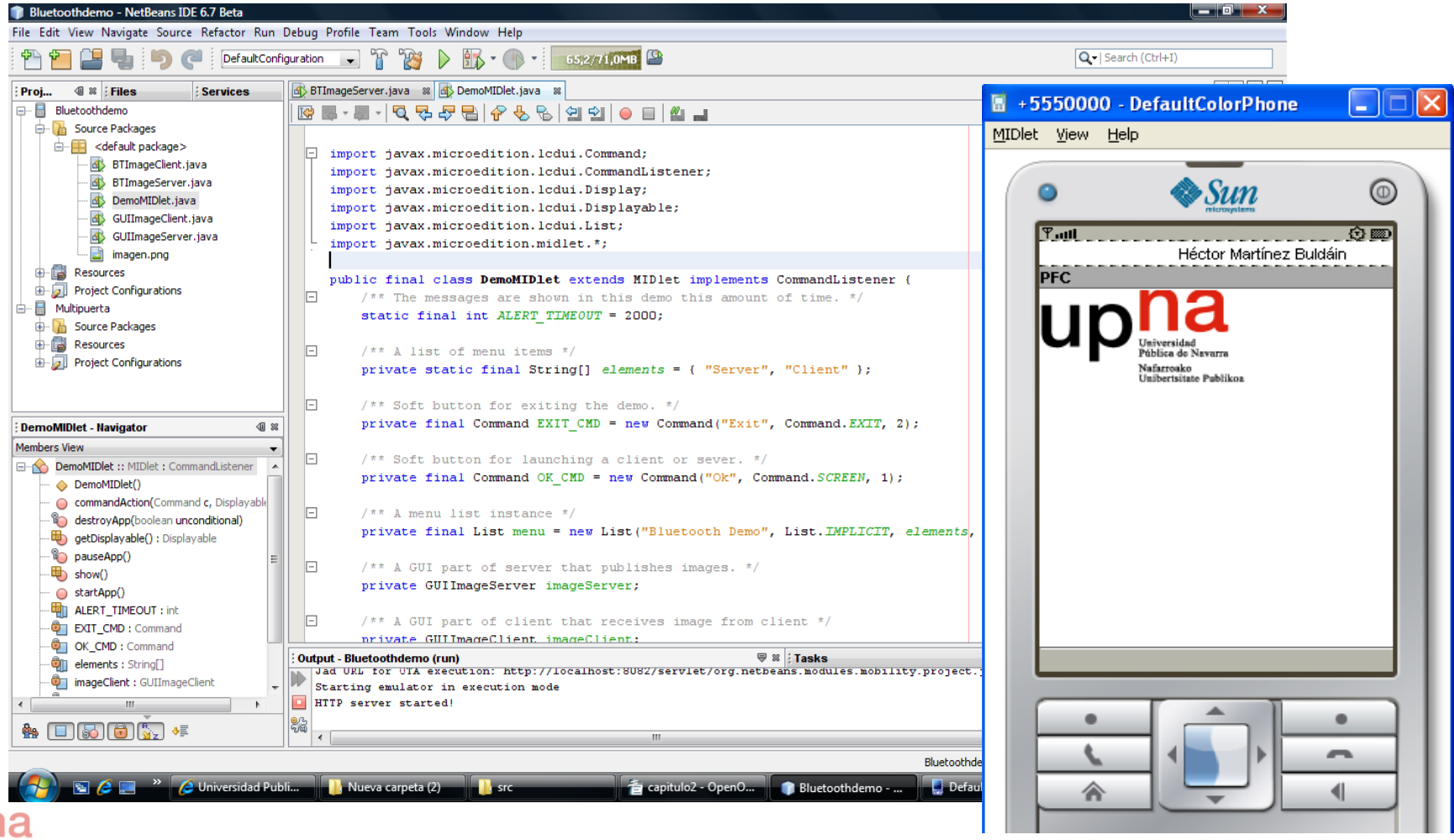
-

J2ME

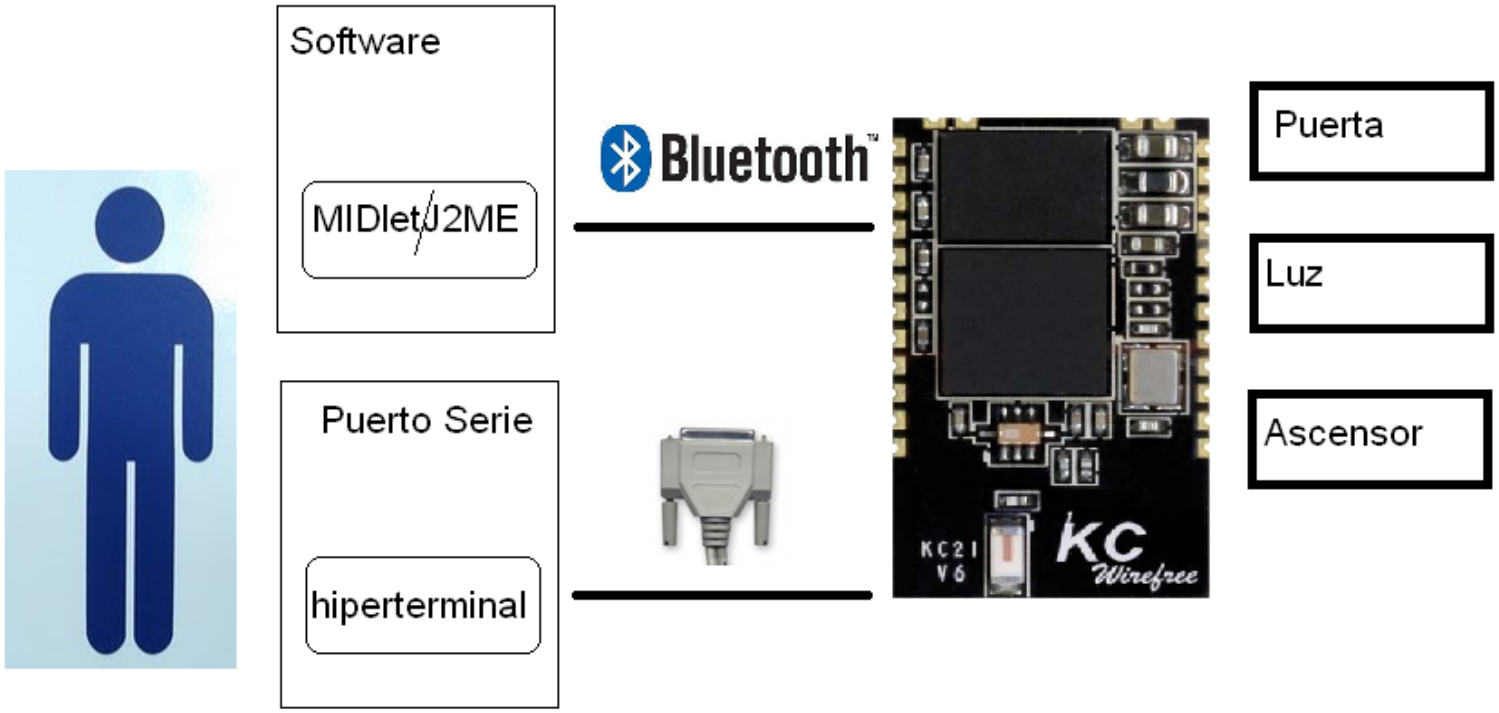
J2ME

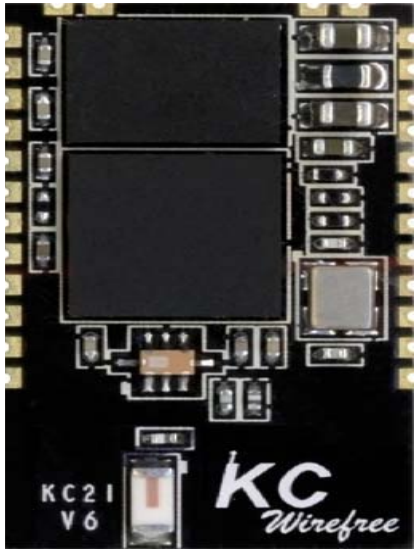


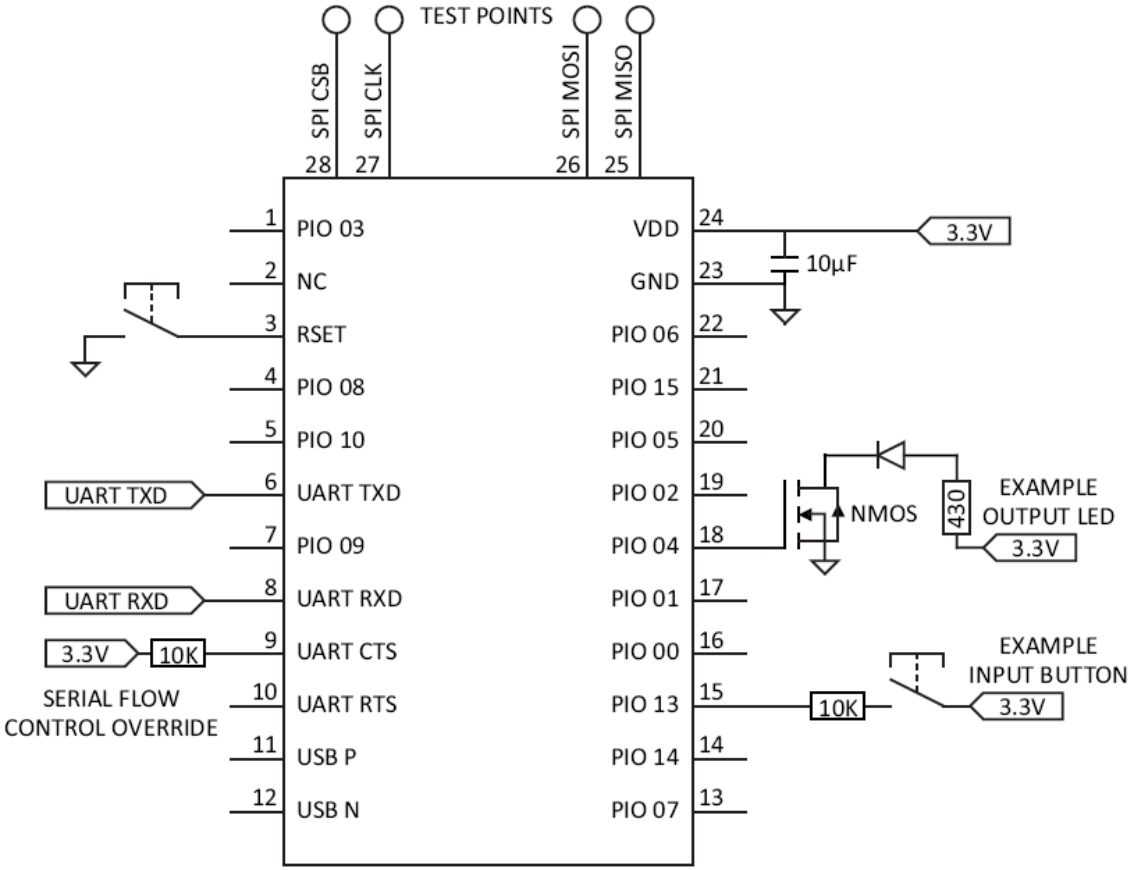
Plataforma NetBeans



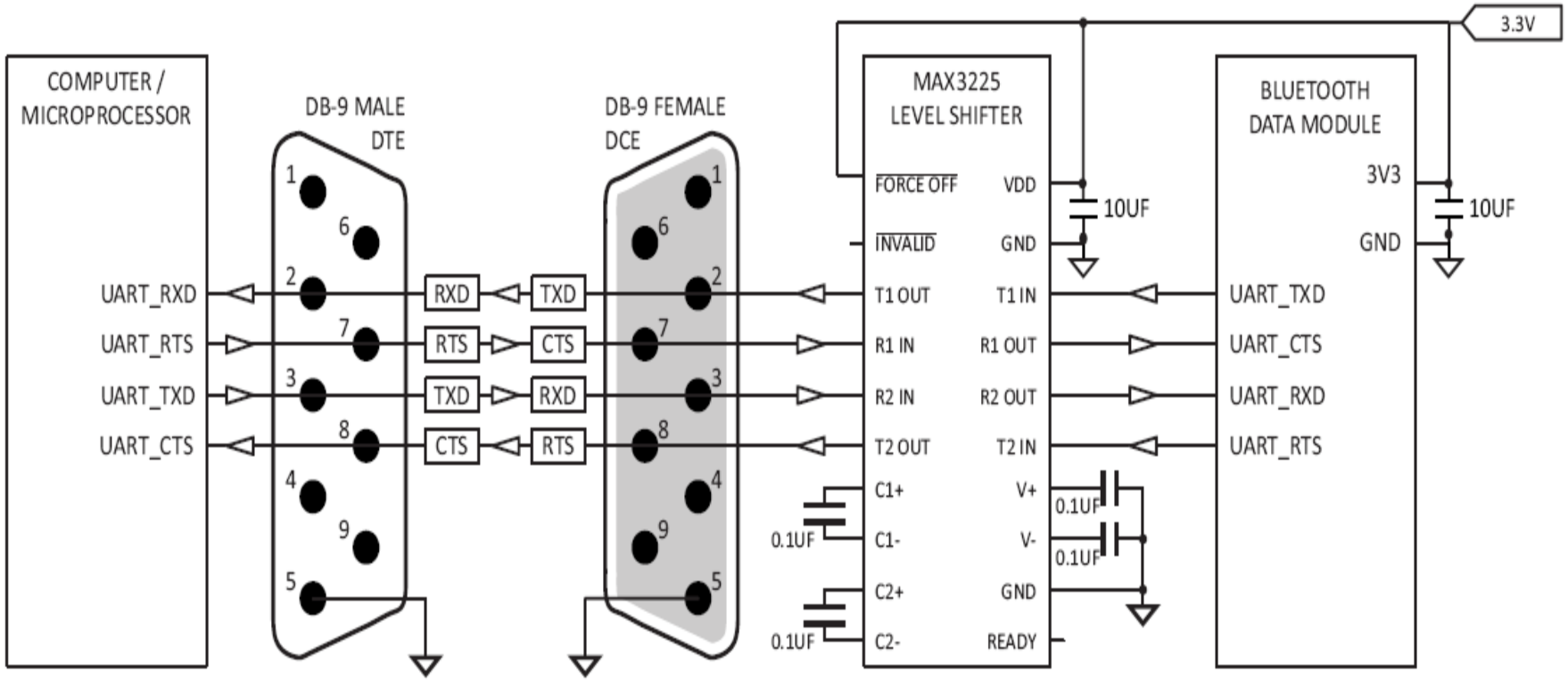
Control de dispositivos con el móvil







Conexión SSP



UART connection with level shifting

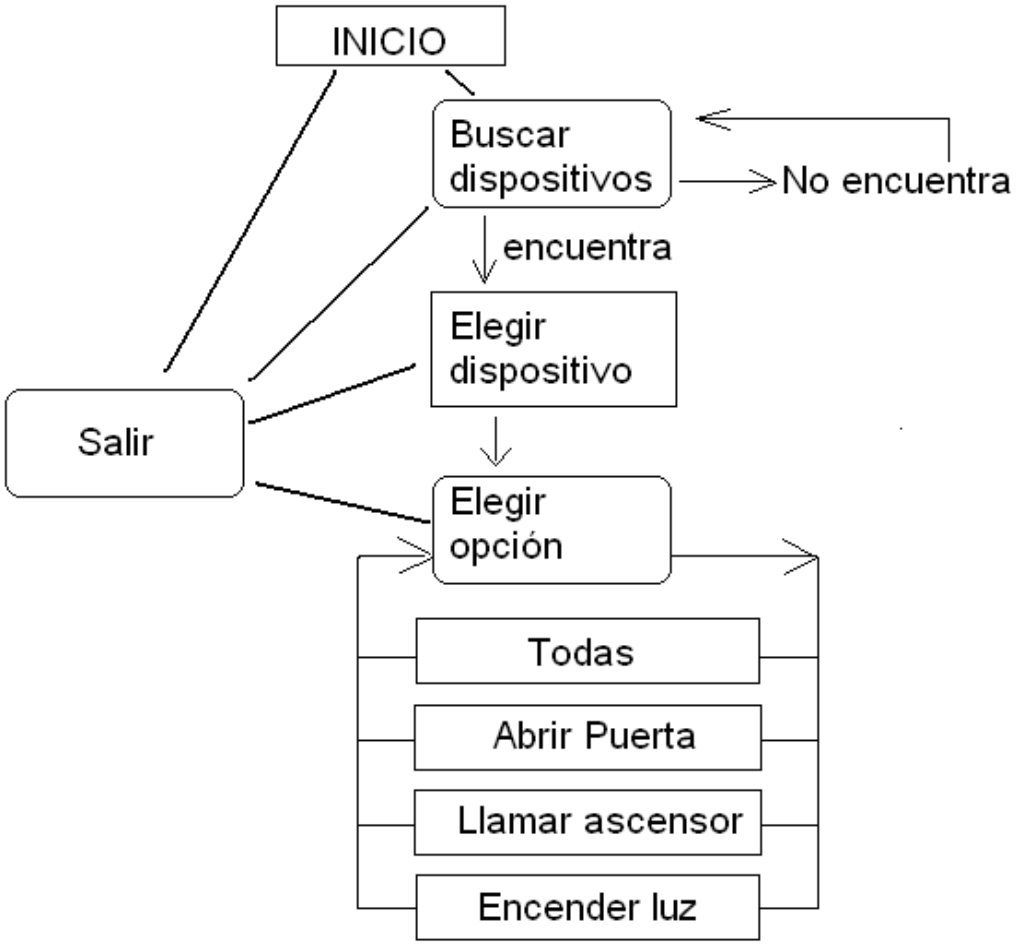
Comandos AT

- AT+KC GPIOConfig
- AT+KC GPIORead
- AT+KC GPIOWrite
- AT+KC RemoteCommand
- AT+KC Reset
- AT+KC SPPConnect
- AT+ZV EnableBond [BD addr] [PIN]

APLICACIÓN BLUETOOTH/J2ME.

- API JSR 82
- -Active LocalDevice
- -Device Discovery
- -Service Discovery
- -SPP

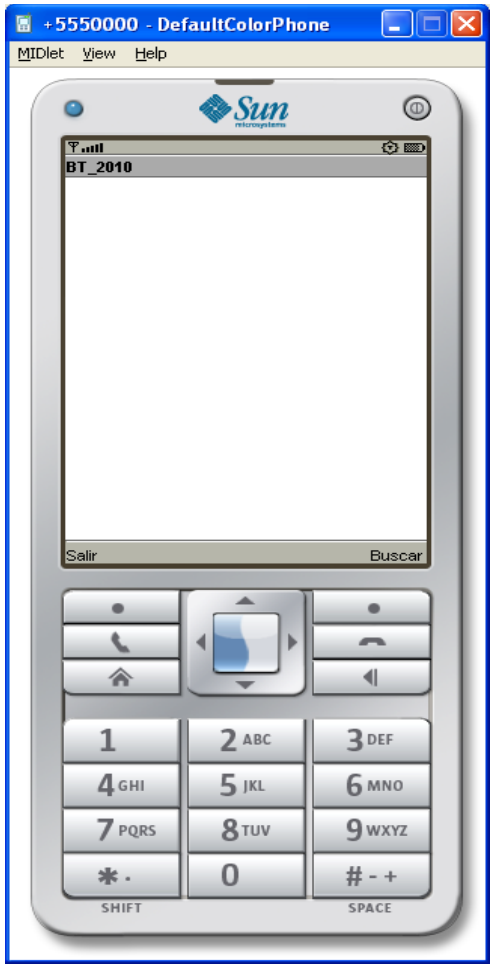
Diagrama

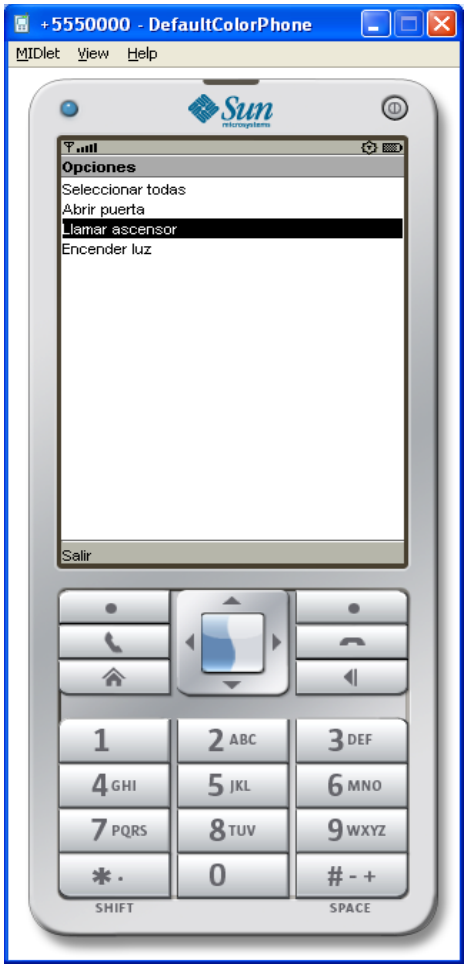


Conexion

```
private class AbrirPuerta extends Thread {
public AbrirPuerta() {}
public void run(){
        try {
            connection = (StreamConnection) Connector.open(Url);
            out = connection.openDataOutputStream();
            in = connection.openDataInputStream();
            out.write("AT+ZV GPIOWrite 10 1\n".getBytes());
            out.flush();
            wait(10000);
            out.write("AT+ZV GPIOWrite 10 0\n".getBytes());
            out.flush();
            in.close();
            out.close();
            connection.close();
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
      Acciones();
   }
}
```

Simulación





Seguridad

- GIAC – LIAC
- Discoverable – Nondiscoverable

LINEAS FUTURAS

SISTEMA BLUETOOTH PARA EL CONTROL DE DISPOSITIVOS

Héctor Martínez Buldáin