DISEÑO E IMPLEMENTACIÓN DE UN ENTORNO DE CONTROL PARA UN SISTEMA DOMÓTICO CON JAVA J2ME

BRAYAN ALEXANDER BELEÑO GARCÍA
JULIO CÉSAR CHALELA BONILLA

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA ELECTRONICA
NEIVA - HUILA
2012

DISEÑO E IMPLEMENTACIÓN DE UN ENTORNO DE CONTROL PARA UN SISTEMA DOMÓTICO CON JAVA J2ME

BRAYAN ALEXANDER BELEÑO GARCÍA
JULIO CÉSAR CHALELA BONILLA

Proyecto de grado presentado como requisito para optar al título de Ingeniero Electrónico

Director:
JOHAN JULIAN MOLINA MOSQUERA
Ingeniero Electrónico

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA ELECTRONICA
NEIVA - HUILA
2012

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____
Firma del director del proyecto

_____
Firma del primer jurado

_____
Firma del segundo jurado

Neiva, 25 de Mayo de 2012.

Dedicado a Dios,
Por la vida, salud y alegría para afrontar cada día;
A mis padres y a mi hermana,
Por su apoyo incondicional,
Ejemplo y orientación en todo momento;
A mi familia y amigos,
Por todos los momentos y alegrías compartidas.

BRAYAN ALEXANDER BELEÑO GARCIA

Este trabajo de grado está dedicado a Dios,
Por brindarme la oportunidad y la dicha de la vida;
A mis Padres, por ser un ejemplo de superación;
A mi novia Jessica y sus padres,
Quienes me han brindado su apoyo incondicional;
A mi familia y amigos.

JULIO CESAR CHALELA BONILLA

AGRADECIMIENTOS

El presente proyecto es un esfuerzo en el cual, directa o indirectamente, participaron varias personas leyendo, opinando, corrigiendo, teniéndonos paciencia, dándonos ánimo, acompañándonos en los momentos de crisis y en los momentos de felicidad.

Agradecemos al ingeniero Johan Julián Molina por haber confiado en nosotros, por la paciencia y por la dirección de este trabajo.

Gracias también a nuestros queridos amigos y compañeros, que nos apoyaron y nos permitieron entrar en su vida durante estos casi seis años de convivir dentro y fuera del salón de clase.

A los profesores que durante la carrera, nos guiaron y compartieron sus conocimientos en las diferentes asignaturas.

CONTENIDO

pág.

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

LISTA DE ANEXOS

pág.

GLOSARIO

Actuador: dispositivo capaz de transformar energía hidráulica, neumática o eléctrica en la activación de un proceso con la finalidad de generar un efecto sobre un proceso automatizado.

API (Application Programming Interface): interface provista por los creadores del lenguaje Java, y que da a los programadores los medios para desarrollar aplicaciones en Java.

Atributos: características de los diversos tipos de clases que posee Java.

Bluetooth: especificación industrial para Redes Inalámbricas de Area Personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz.

Bus: sistema de comunicaciones que transfiere datos entre los componentes de un circuito eléctrico. Está formado por cables o pistas en un circuito impreso, dispositivos como resistores y condensadores además de circuitos integrados.

CCS: Software basado en lenguaje C para programar firmware para microcontroladores.

CDC (Connected Device Configuration): conjunto mínimo de APIs que definen 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 de mesa.

Clase: agrupación de datos (variables o campos) y de funciones (métodos) que operan sobre estos mismos datos.

CLDC (Connected Limited Device Configuration): conjunto mínimo de APIs que definen las capacidades básicas que deben tener dispositivos con recursos limitados como teléfonos móviles, buscapersonas y los principales asistentes digitales personales.

Cliente: aplicación que permite establecer una conexión para el intercambio de información con un servidor.

Control On-Off: modo de control en el cual la salida del controlador solo permanecerá en dos estados opuestos.

Controlador: dispositivo electrónico encargado de inspeccionar uno o más procesos.

Datagrama: estructura interna de un paquete de datos. Estos paquetes de datos se transfieren en una conexión.

Domótica: integración de la tecnología en el diseño inteligente de un recinto cerrado.

EAGLE (Easily Applicable Graphical Layout Editor): editor para automatización de diseño electrónico y captura esquemática, diseño de tarjeta de circuito impreso, y autotrazador de circuitos eléctricos y electrónicos.

Flow Design: herramienta visual que permite configurar una aplicación hecha en Netbeans de manera gráfica.

GAP (Generic Access Profile): define el uso de las capas bajas de la pila de protocolos Bluetooth, incluyendo dispositivo de la funcionalidad de gestión.

Gestor de aplicaciones AMS (Application Manager System): programa encargado de manejar la descarga y ciclo de vida de las *aplicaciones* creadas e instaladas bajo la tecnología J2ME.

HTTP (Hypertext Transfer Protocol): método más común de intercambio de información. Este protocolo opera a través de solicitudes y respuestas, entre un "cliente" y un "servidor".

IDE (integrated development environment): tipo de programa que facilitan el desarrollo de aplicaciones para los dispositivos móviles.

Identificador: nombre que identifica a una variable, a un método o función miembro, a una clase. Cada lenguaje tiene ciertas reglas para componer los identificadores.

J2EE (Java to Enterprise Edition): plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java con arquitectura de N capas distribuidas y que se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

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

J2SE (Java to Standard Edition): colección de APIs del lenguaje de programación Java útiles para muchos programas de la Plataforma Java. La Plataforma Java 2, Enterprise Edition incluye todas las clases en el Java SE, además de algunas de las cuales son útiles para programas que se ejecutan en servidores sobre workstations.

JAD (JAva Decompiler): tipo de archivo que describe los contenidos de los archivos de tipo .jar.

JAR (Java ARchive): tipo de archivo que permite ejecutar aplicaciones escritas en lenguaje Java. Los archivos JAR están comprimidos con el formato ZIP y cambiada su extensión a .jar.

JSR (Java Specification Request): documentos formales que describen las especificaciones y tecnologías propuestas para que sean añadidas a la plataforma Java.

Maquina Virtual Java (Java Virtual Machine): programa ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el bytecode Java), el cual es generado por el compilador del lenguaje Java.

Métodos: definen un grupo de instrucciones que son separadas, que definen un comportamiento y que requieren en algunos casos de valores para su proceso.

Midlet: programa en lenguaje de programación Java para dispositivos embebidos, más específicamente para la máquina virtual Java MicroEdition. Generalmente son juegos y aplicaciones que corren en un teléfono móvil. Está desarrollada bajo la especificación MIDP.

MIDP (Mobile Information Device Profile): versión de J2ME integrada en el hardware de celulares relativamente modernos que permite el uso de programas java denominados MIDlets, tales como juegos, aplicaciones o todo tipo de software.

NetBeans: entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java.

PIC (Peripheral Interface Controller): familia de microcontroladores fabricados por Microchip Technology Inc. y derivados del PIC1650, originalmente desarrollado por la división de microelectrónica de General Instrument.

PCB (Printed Circuit Board): se utiliza para soportar mecánicamente y conectar eléctricamente componentes electrónicos que usan pistas trazadas o caminos conductores de hojas de cobre laminadas sobre un sustrato no conductor.

Protocolo de Comunicaciones: reglas de comunicación que permiten el flujo de información entre equipos que manejan lenguajes distintos.

Servidor: aplicación que ofrece un servicio por un puerto especificado a diversos usuarios para el intercambio de información.

Sockets: formas en las que se puede interconectar 2 o más programas mediante el uso de internet. En java se utilizan para poder crear conexiones utilizando básicamente una IP/hostname y un puerto para establecer la conexión.

Sistema Embebido: sistema de computación diseñado para realizar una o algunas pocas funciones dedicadas frecuentemente en un sistema de computación en tiempo real.

SPP (Serial Port Profile): permite que dispositivos Bluetooth realicen emulación de RS232 (o similar). El escenario cubierto por este perfil trata con aplicaciones comerciales que utilizan Bluetooth como un sustituto del cable, utilizando una capa de abstracción que representa un puerto serie virtual.

Symbian: sistema operativo que fue producto de la alianza de varias empresas de telefonía móvil, entre las que se encuentran Nokia, Sony Ericsson, Psion, Samsung, Siemens, Arima, Benq, Fujitsu, Lenovo, LG, Motorola, Mitsubishi Electric, Panasonic, Sharp, etc. Sus orígenes provienen de su antepasado EPOC32, utilizado en PDA's y Handhelds de PSION.

Transceptor: dispositivo que realiza dentro de una misma caja o chasis, funciones tanto de trasmisión como de recepción, utilizando componentes de circuito comunes para ambas funciones.

URL (Uniform Resource Locator): referencia (una dirección) a un recurso de Internet.

UUID (Universally Unique Identifier): número único identificador de información, de 16 bytes (128 bits), usado en el desarrollo de software.

ABSTRACT

The main purpose of this project is to facilitate the management of common electrical devices in the home via a wireless connection established between a wireless phone and a module that controls the operation of these devices by commands received via Bluetooth.

This project can be controlled: the lighting, the opening of the doors, sprinklers. The system is capable of allowing the user to control via Bluetooth only.

To establish the connection, the cellphone must have Bluetooth technology. Once connected, the cellphone sends commands that are received by the Bluetooth module and this in turn transmitted to the control module to perform a specific action.

For the mobile phone is able to establish the Bluetooth connection and send commands, it was necessary to create an application to the cellphone. The application was developed on Java J2ME, a Java sub-platform created specifically for mobile devices and low memory capacity.

The Bluetooth module used at the data reception is the KC-5100, it has terminals remotely programmable. Taking advantage of this feature, the Java application sends over a Bluetooth connection, commands that will do it that these terminals will transmit the information to the PIC which will allow to the pin to take a specific value. Thus, the value of the PIC's terminal will determine the state of the electrical device.

Having implemented the entire system, it is found that Bluetooth communication is a very useful tool in applications related with domotic. It is possible to give it an additional use to the Bluetooth technology, including in mobile phones and the Java language is a good alternative to create applications for mobile phones.

Keywords: Domotic, actuators, communication buses, Java J2ME.

RESUMEN

El objeto de este proyecto es facilitar el manejo de dispositivos eléctricos comunes en el hogar a través de una conexión inalámbrica creada entre un teléfono celular y un módulo que controlará el funcionamiento de dichos dispositivos mediante comandos recibidos vía Bluetooth.

Con este proyecto se puede controlar: la iluminación, la abertura de las puertas, los aspersores, etc. El sistema es capaz de permitir al usuario hacer el control vía Bluetooth únicamente.

Para poder establecer la conexión, el celular debe contar con tecnología Bluetooth. Una vez realizada la conexión, el celular envía comandos que son recibidos por el módulo bluetooth y este a su vez los transmite al modulo de control para realizar una acción específica.

Para que el teléfono móvil sea capaz de establecer la conexión Bluetooth y enviar comandos, fue necesario crear una aplicación para el celular. La aplicación fue desarrollada sobre Java J2ME, una sub-plataforma de Java creada especialmente para dispositivos móviles y de baja capacidad de memoria.

El módulo Bluetooth que se usa en la parte de recepción de datos es el KC-5100; éste cuenta con terminales programables de forma remota. Aprovechando esta característica, la aplicación Java envía, sobre la conexión Bluetooth, comandos que harán que dichos terminales transmitan la información al PIC el cual permitirá que los pines tomen un valor específico. De esta manera, el valor que tenga el terminal del PIC determinará el estado del dispositivo eléctrico.

Una vez implementado el sistema completo, se comprueba que la comunicación Bluetooth es una herramienta muy útil en aplicaciones relacionadas con domótica; que es posible darle un uso adicional a la tecnología Bluetooth, incluida en teléfonos celulares, y que el lenguaje Java es una buena alternativa para crear aplicaciones para teléfonos móviles.

Palabras claves: Domótica, actuadores, PIC, buses de comunicación, Java J2ME.

INTRODUCCIÓN

Cuando se habla de Domótica, se hace referencia a la automatización y control de aparatos y sistemas de instalaciones eléctricas, electrónicas, mecánicas, hidráulicas, etc. de forma centralizada, descentralizada y/o remota. Ya que esta rama de la ciencia va ligada de manera muy estrecha al mejoramiento de la calidad de vida de las personas, los objetivos principales de esta son: ahorro energético, aumento del confort, mejora en la seguridad y comunicación integrada de los servicios de la vivienda.

En el momento en que se implementa una solución domótica, esta puede variar desde un único dispositivo que realiza una sola acción, hasta amplios sistemas que controlan prácticamente todas las instalaciones de una vivienda. Para ello, generalmente se utilizan los siguientes dispositivos: controladores, actuadores, sensores, buses de comunicación y la interface con el usuario. Cabe destacar que todos los dispositivos del sistema domótico no tienen que estar físicamente separados, sino que varias funcionalidades pueden estar combinadas en un equipo.

Teniendo en cuenta que los sistemas domóticos deben ser de fácil manejo por parte del usuario, la tecnología Java nos permitirá crear una interface de comunicación entre el dispositivo móvil y la tarjeta electrónica, que servirá como hardware de prueba para el control de la sección domótica del proyecto.

1. PLANTEAMIENTO DEL PROBLEMA

1.1 DESCRIPCIÓN

En estos momentos, el país cuenta con muy poco desarrollo tecnológico electrónico orientado hacia áreas de aplicación comercial, como lo es la Domótica, y cabe anotar que existen muy pocas empresas con la solidez suficiente para ofrecer los servicios y beneficios que este tipo de proyectos implican[1].

En la actualidad, las empresas que brindan estos servicios lo hacen de tal manera que ellas mismas proporcionan a sus clientes el hardware y software necesario para el correcto funcionamiento de los sistemas adquiridos, por ende, cada empresa puede manejar distintos tipos de integración para su tecnología. En base a lo anterior, existe la necesidad de desarrollar un software con un programa que sea de distribución gratuita, con gran capacidad de procesamiento y confiabilidad, con facilidad de integrarlo en cualquier dispositivo móvil que cuente con soporte Java y que sea capaz de controlar el hardware diseñado para tal fin; dando como resultado un ahorro desde el punto de vista económico con respecto a otros proyectos realizados bajo estos sistemas, así como también innovación e investigación sobre éstos mismos.

[1] INVESTIGACIÓN PARA el desarrollo de software en Domótica orientado al sector doméstico y empresarial de Colombia. [Anónimo]. [En Línea]. [Citado el 12 de Marzo de 2012]. <Disponible en http://es.scribd.com/doc/30715431/INVESTIGACION-PARA-EL-DESARROLLO-DE-SOFTWARE-EN-DOMOTICA-ORIENTADO-AL-SECTOR-DOMESTICO-Y-EMPRESARIAL-DE-COLOMBIA>

2. OBJETIVOS

2.1 OBJETIVO GENERAL

- Diseñar e implementar un entorno de control para un sistema domótico con Java J2ME, creando una infraestructura de información basada en ambiente Java, usando protocolo de comunicación 802.15 (Bluetooth).

2.2 OBJETIVOS ESPECÍFICOS

- Diseñar e implementar una interface de control interactiva que será instalada en el teléfono móvil, empleando para ello tecnología Java J2ME de manera que sea de fácil manejo para el usuario.

- Diseñar e implementar un sistema de comunicación inalámbrico con protocolo 802.15 (Bluetooth) para salida y entrada de datos entre el dispositivo móvil y el hardware de prueba.

- Diseñar e implementar una tarjeta electrónica que servirá como hardware de prueba para el control de la sección domótica del proyecto y la implementación del mismo.

- Realizar un control On-Off de los dispositivos y elementos utilizados en la tarjeta electrónica como interruptores digitales, led's y actuadores.

3. MARCO TEÓRICO

3.1 DOMÓTICA

El concepto domótica[2] se refiere a la automatización y control (encendido / apagado, apertura / cierre y regulación) de aparatos y sistemas de instalaciones eléctricas y electrotécnicos (iluminación, climatización, persianas y toldos, puertas y ventanas motorizados, el riego, etc.) de forma centralizada y/o remota. El objetivo del uso de la domótica es el aumento del confort, el ahorro energético y la mejora de la seguridad personal y patrimonial en la vivienda.

3.1.1 Dispositivos

Una solución de tipo domótico puede variar desde un único dispositivo, que realiza una sola acción, hasta amplios sistemas que controlan prácticamente todas las instalaciones dentro de la vivienda. En la Figura 1 se observan los distintos dispositivos de los sistemas de domótica, y se pueden clasificar en los siguientes grupos:

- **Controlador:** Los controladores son los dispositivos que gestionan el sistema según la programación y la información que reciben. Puede haber un solo controlador, o varios distribuidos por el sistema.

- **Actuador:** El actuador es un dispositivo capaz de ejecutar y/o recibir una orden del controlador y realizar una acción sobre un aparato o sistema (encendido/apagado, subida/bajada, apertura/cierre, etc.).

- **Sensor:** El sensor es el dispositivo que monitoriza el entorno captando información que transmite al sistema (sensores de agua, gas, humo, temperatura, viento, humedad, lluvia, iluminación, etc.).

- **Bus:** El bus es el medio de transmisión que transporta la información entre los distintos dispositivos por un cableado propio, por la redes de otros sistemas (eléctrica, telefónica, de datos) o de forma inalámbrica.

- **Interface:** Los interfaces refieren a los dispositivos (pantallas, móvil, Internet, conectores) y los formatos (binario, audio) en que se muestra la información del sistema para los usuarios (u otros sistemas) y donde los mismos pueden interactuar con éste.

[2] CASADOMO, El Portal Del Edificio y Hogar Digital. Domótica. [En Línea]. Marzo 2012. [Citado el 13 de Marzo de 2012]. <Disponible en http://www.casadomo.com/noticiasDetalle.aspx?c=14 >

Figura 1. Ejemplo de dispositivos de sistemas domóticos

Fuente: http://www.casadomo.com/noticiasDetalle.aspx?c=14

3.2 TELEFONO MÓVIL

El teléfono móvil es un dispositivo inalámbrico electrónico que permite tener acceso a la red de telefonía celular o móvil. Se denomina celular en la mayoría de países latinoamericanos debido a las antenas repetidoras que conforman la red, cada una de las cuales es una célula, si bien existen redes telefónicas móviles satelitales. Su principal característica es su portabilidad, que permite comunicarse desde casi cualquier lugar. Aunque su principal función es la comunicación de voz, como el teléfono convencional[3].

En los últimos años, han adquirido funcionalidades que van mucho más allá que limitarse llamar o enviar mensajes de texto, y tienen muchas funciones de otros dispositivos (fotos, agenda, reloj despertador, calculadora, reproductores etc.).

[3] INSTITUTO TECNOLOGICO DE CANARIAS. Telefonía Móvil. Marzo 2012- [Citado el 13 de Marzo de 2012]. <Disponible en http://www.itccanarias.org/web/difusion/como_funciona/movil/index.jsp?lang=es>

3.3 BLUETOOTH

Bluetooth[4] es una especificación industrial para Redes Inalámbricas de Área Personal (WPANs) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia (2,4 GHz).

Ésta especificación esta recogida por el grupo de trabajo 802.15.1 del IEEE[5]. Los principales objetivos que se pretenden conseguir con esta norma son:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar cables y conectores entre éstos.

Los dispositivos que con mayor frecuencia utilizan esta tecnología pertenecen a sectores de las telecomunicaciones y la informática personal -teléfonos móviles, computadoras portátiles, PCs, impresoras o cámaras digitales-.

3.3.1 Usos y aplicaciones

Bluetooth está diseñado para dispositivos de bajo consumo, con una cobertura baja, y basados en transceptores de bajo costo. Gracias a este protocolo, los dispositivos que lo implementan pueden comunicarse entre ellos cuando se encuentran dentro de su alcance.

Las comunicaciones se realizan por radiofrecuencia de forma que los dispositivos no tienen que estar alineados y pueden incluso estar en habitaciones separadas si la potencia de transmisión lo permite. Estos dispositivos -como se indica en la Tabla 1- se clasifican en "clases" en referencia a su potencia de transmisión, siendo totalmente compatibles los dispositivos de una clase con los de las otras.

Tabla 1. Clase de dispositivos Bluetooth

TIPO	Potencia máxima Permitida (mW)	Potencia máxima Permitida (dBm)	Rango (Aproximado)
CLASE 1	100 mW	20 dBm	~100 metros
CLASE 2	2.5 mW	4 dBm	~25 metros
CLASE 3	1 mW	0 dBm	~1 metro

[4] BLUEZONA, Especialista Bluetooth. Bluetooth. [En Línea]. 2008. [Citado el 13 de Marzo de 2012]. <Disponible en http://www.bluezona.com/que-es-bluetooth/>

[5] IEEE XPLORE, Digital Library. IEEE Standard for Information Technology – Telecomunications and Information Exchange. [En Línea]. 2002. [Citado el 13 de Marzo de 2012]. <Disponible en http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7932>

En la Tabla 2 se observa la clasificación de los dispositivos con Bluetooth según su ancho de banda:

Tabla 2. Ancho de banda según versión Bluetooth

VERSIÓN	ANCHO DE BANDA
Versión 1.2	1 Mbit/s
Versión 2.0 + EDR	3 Mbit/s
Versión 3.0 + HS	24 Mbit/s

3.3.2 Comparación con otras tecnologías

3.3.2.1 Bluetooth e infrarrojo

- El infrarrojo requiere comunicación lineal entre transmisor y receptor, lo que hace imprescindible la línea de vista para su efectiva transmisión.

- Las frecuencias de la banda del infrarrojo no permiten penetración a través de paredes, ventaja que tiene la radiofrecuencia que opera Bluetooth.

- La comunicación con IrDA siempre será uno a uno -ventaja desde el punto de vista de la seguridad- mientras que con Bluetooth existe la posibilidad de manejar aplicaciones multipunto.

- Bluetooth permite la generación de redes.

- La velocidad de transmisión de datos son más elevadas en la tecnología IrDA, alcanza hasta 16Mbps. Bluetooth puede alcanzar máximo 3Mbps.

- 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.

- Desde el punto de vista de las interferencias, IrDA es afectado por la luz fluorescente y solar, mientras Bluetooth trata de no ser afectado por otros sistemas que también usan la frecuencia de 2.4GHz.

3.3.2.2 Bluetooth y Wi-fi

- Wi-fi es capaz de alcanzar una velocidad de 54 Mbps, ofrece seguridad tanto por encriptación -mediante sistemas de claves-, como por cifrado de clave dinámico, es decir, la clave está cambiando constantemente.

- Una red Wi-fi requiere de una configuración tanto en hardware como en software mientras que 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 Wi-fi, el costo y el consumo de energía también es mucho más alto.

- Los dispositivos Bluetooth puede acceder a los recursos de una LAN. Para esto, el punto de acceso de LAN debe proporcionar las capacidades de conversiones protocolares[6].

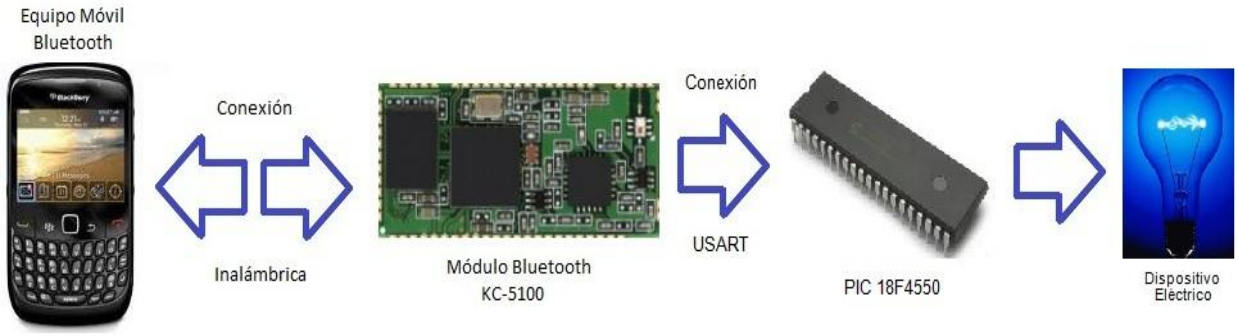
3.4 SISTEMA BLUETOOTH PARA CONTROL DE DISPOSITIVOS

Para realizar control de dispositivos eléctricos usando Bluetooth, es necesario que éstos cuenten con ésta tecnología, o bien sean adaptados a otros que lo estén. Para el control del acceso (por ejemplo: una puerta eléctrica) y de iluminación (por ejemplo: una lámpara) de una habitación, se puede emplear módulos Bluetooth que les proporcionen esta característica.

El sistema implementado -véase Figura 2- constará de un dispositivo móvil (teléfono celular), un módulo Bluetooth, un microcontrolador PIC y un dispositivo eléctrico común. Dentro del teléfono se implementaría 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 clase 1, un pequeño procesador y una memoria flash. El módulo puede ser programado para interactuar con el microcontrolador y este a su vez con el dispositivo que se desea controlar. Una vez establecida la conexión, el equipo móvil será capaz de enviar comandos que controlarán el dispositivo eléctrico a través del módulo Bluetooth.

[6] FROUFE QUINTAS, Agustín, JORGE CÁRDENAS, Patricia. J2ME Java 2 Micro Edition: Manual de Usuario y Tutorial, Primera Edición. Alfaomega, México, 2004.

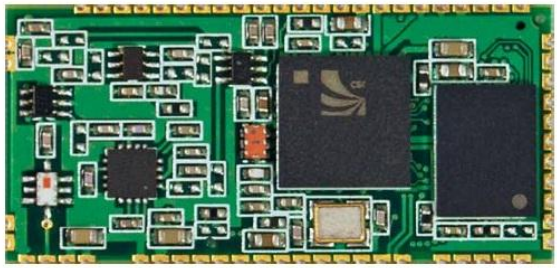
Figura 2. Esquema simple del sistema de control



3.5 MODULO BLUETOOTH KC-5100

Un módulo Bluetooth tiene una parte de Software y una de Hardware. Por ejemplo, en el módulo KC-5100[7] -mostrado en la Figura 3- que será usado en este proyecto, la parte de hardware está compuesta por: un dispositivo de radio, encargado de modular y transmitir la señal; y un controlador digital. El controlador digital está 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 3. Módulo Bluetooth KC-5100



33.2mm x 15.8 mm x 1.8 mm

Fuente: http://www.kcwirefree.com/docs/KC5100_Datasheet.pdf

[7] KC WIREFREE, We Make Bluetooth Work For You. [En Línea]. [Citado el 13 de Marzo de 2012]. <Disponible en http://kcwirefree.com/docs/KC5100_Datasheet.pdf >

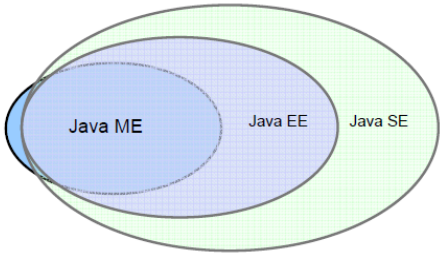
3.6 JAVA

Java[8] es un lenguaje de programación orientado a objetos que 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.

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 4.

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

Figura 4. Diagrama de entornos de la plataforma Java



Fuente: http://academica-e.unavarra.es/bitstream/handle/2454/2167/577244.pdf?sequence=1

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

- **Java SE (J2SE), Java Standard Edition**, 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.

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

- **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.

[8] GALVEZ ROJAS, Sergio, ORTEGA DÍAZ, Lucas. Java A Tope: Java 2 Micro Edition J2ME. Edición Electrónica, Universidad de Málaga. España, 2003.

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

3.7 JAVA J2ME

J2ME[9] 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.

3.7.1 Aplication Programming Interfaces APIs

Un API Java es una Interfaz de Programación de Aplicaciones provista por los creadores del lenguaje Java, mediante un proceso conocido como JCP -Java Community Process-. Dicho proceso conlleva el uso de documentos conocidos como *Java Specification Request* (JSR), los cuales describen las especificaciones y tecnologías propuestas para añadirse en la plataforma, y que da a los programadores los medios para desarrollar aplicaciones Java.

3.7.2 Maquinas virtuales para Java J2ME

Una máquina virtual de Java (JVM) es un programa encargado de: interpretar código intermedio 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. Ésta puede ser de varios tipos, entre los más usados se tiene:

- **KVM**: Es una máquina virtual pequeña. 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 (dispositivos móviles).

- **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.

[9] GALVEZ ROJAS, Sergio, ORTEGA DÍAZ, Lucas. Java A Tope: Java 2 Micro Edition J2ME. Edición Electrónica, Universidad de Málaga. España, 2003.

3.7.3 Configuraciones Java J2ME

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

- Características soportadas por lenguaje de programación Java.
- Características soportadas por la Máquina Virtual Java.
- Bibliotecas básicas de Java soportadas.

Existen dos configuraciones dentro de Java ME:

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

Éstas definen características incluidas por el lenguaje Java, funcionalidades de la máquina virtual Java (CVM), las APIs usadas por CDLC necesarias para el desarrollo de aplicaciones, subconjuntos de librerías de J2SE, requerimientos de hardware de dispositivos, E/S, acceso a redes, seguridad, etc.

3.7.3.1 Configuración de Dispositivos con Conexión CDC

La CDC funciona en dispositivos con algunas capacidades de memoria y procesamiento (equipos de TV por Internet, decodificadores de TV digital, electrodomésticos, sistemas de navegación, etc.). La CDC usa una CVM tomando en cuenta las limitaciones en la interfaz gráfica y en memoria del equipo.

3.7.3.2 Configuración de dispositivos limitados con conexión CLDC

CLDC es una especificación general para un amplio abanico de dispositivos, que se caracterizan por sus limitaciones en cuanto a capacidad gráfica, cómputo y memoria. Algunas de estas restricciones vienen dadas por el uso de la KVM, necesaria al trabajar con la CLDC debido a su pequeño tamaño.

Los dispositivos que usan CLDC se caracterizan por:

- Disponer como mínimo de 128 Kb de memoria para la Máquina Virtual.
- Tener un procesador de 16 o 32 bits con al menos 25 MHz de velocidad.
- Ofrecer bajo consumo, debido a que estos dispositivos trabajan con suministro de energía limitado, normalmente baterías.
- Tener conexión a algún tipo de red, normalmente sin cable, con conexión intermitente y ancho de banda limitado (unos 9600 bps).

3.7.3.3 Librerías incluidas en el 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 -véase Tabla 3- es menor que en CDC, esto debido al tipo de dispositivos que usan esta aplicación y su alcance.

Tabla 3. Librerías incluidas en CLDC

PAQUETE CLDC	DESCRIPCION
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.

3.7.4 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.

JSR-82 permite, entre otras, las siguientes operaciones:

- Registro de servicios.
- Descubrimiento de dispositivos y servicios.
- Establecer conexiones entre dispositivos.
- Usar dichas conexiones para mandar y recibir datos
- Manejar y controlar las conexiones de comunicación.
- Ofrecer seguridad a dichas actividades.

3.7.4.1 Paquetes incluidos en el JSR-82

Actualmente el JSR-82 consta de dos paquetes independientes:

- **javax.bluetooth:** Son 13 clases e interfaces necesarios para establecer una comunicación inalámbrica usando el protocolo Bluetooth.
- **javax.obex:** Son 8 clases necesarias para enviar objetos entre dos dispositivos, independientemente del mecanismo de transporte usado entre ellos.

3.8 NetBeans IDE 7.0.1

NetBeans -véase Figura 5- 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[10]. NetBeans IDE 7.0.1 es un módulo usado por la plataforma NetBeans para desarrollar aplicaciones móviles, e incluye las librerías necesarias y la opción de edición del código fuente de manera interactiva para implementar aplicaciones para dispositivos móviles.

Figura 5. NetBeans IDE 7.0.1

Fuente: http://www.netbeans.com/kb/docs/java/quickstart.html

[10] DOMINGUEZ-DORADO, Manuel. NetBeans IDE 4.1. La alternativa a Eclipse, Todo Programación. Nº 13. Iberprensa, Madrid, España, 2005.

4. DESARROLLO DE LA APLICACIÓN

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 J2ME se debe pasar por varias fases desde la edición del código hasta poder tener listo el MIDlet que se instalará en el dispositivo móvil. Estas fases de desarrollo de un MIDlet se resumen a continuación.

4.1 MIDLET

MIDlet es un programa en lenguaje de programación Java para dispositivos embebidos, más específicamente para la JVM. Está desarrollada bajo la especificación MIDP (Perfil para Información de Dispositivo Móvil).

Un MlDlet tiene que ejecutarse en un entorno muy concreto e implementar una serie de métodos que ofrece la especificación MIDP. Además, puede estar en tres estados diferentes: en ejecución, en pausa o finalizado.

Dentro de la aplicación, la interface de usuario interactuará con la interface de comunicación, la cual establecerá la conexión Bluetooth, para alcanzar el objetivo de control de este proyecto.

4.2 DISEÑO DE LA INTERFACE DE USUARIO

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

4.3 PROGRAMACIÓN DE LA INTERFACE DE USUARIO

La aplicación Java ME estará sustentada principalmente en dos APIs, por un lado CLDC que hereda algunas de las clases de J2SE, y MIDP que añade nuevas clases que permitirán crear interfaces de usuario[11].

[11] GALVEZ ROJAS, Sergio, ORTEGA DÍAZ, Lucas. Java A Tope: Java 2 Micro Edition J2ME. Edición Electrónica, Universidad de Málaga. España, 2003.

Las clases más importantes de J2SE que ofrece CLDC son las siguientes:

- java.lang
- java.util
- java.io

Además MIDP añade los siguientes paquetes:

- javax.microedition.midlet
- javax.microedition.lcdui
- javax.microedition.io
- javax.microedition.rms

El paquete **javax.microedition.midlet**, es el más importante de todos. Sólo contiene a la clase MlDlet, 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.

4.3.1 ELEMENTOS DE LA INTERFACE DE USUARIO

Command es un elemento que permite interactuar con el usuario y le permite introducir comandos. En la Tabla 4, se muestran los tipos de comandos disponibles en J2ME para cada pantalla:

Tabla 4. 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.

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 accederá mediante un menú. Dentro de la aplicación, el resultado de los diversos comandos se aprecia en la Figura 6.

Figura 6. Comandos dentro de un formulario

4.3.1.1 Tipos de clases

La clase Screen, permite crear las interfaces gráficas de alto nivel. Un objeto creado con 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.

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. La diferencia entre uno y otro tipo de alerta es básicamente el tipo de sonido o efecto que produce el dispositivo.

La clase List, permite crear listas de elementos seleccionables. Los posibles 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.

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 alguna 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.

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 interfaces más elaboradas. Los elementos que se podrían añadir a un formulario son:

- StringIItem
- lmageItem
- TextField
- DateField
- ChoiceGroup
- Gauge

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.

4.4 FLOW DESIGN DE NetBeans

El software utilizado para la realización del programa, también tiene la posibilidad de crear la interfaz de usuario a través de la opción Flow Design, en donde se agregan listas, alertas, formularios, comandos, etc. de manera interactiva mientras el código fuente se va modificando automáticamente. Por ejemplo, para incluir un formulario en la aplicación solo es necesario arrastrarlo hasta la hoja de trabajo.

En la Figura 7, se ve la pantalla de edición interactiva con el diseño de la aplicación completa, en el que se aprecia la forma de enlazar clases mediante comandos en ellas.

El código generado por Flow Design aparece resaltado en color gris, como se muestra en la Figura 8, y no se puede modificar directamente, sino únicamente usando esta misma herramienta.

Flow Design facilita la creación de la interfaz de usuario. Es beneficioso ir supervisando visualmente cada elemento incluido en una pantalla y estar seguro de que el resultado es satisfactorio paso a paso. Esta herramienta es de gran ayuda, aunque crea muchas líneas de código que algunas veces no son indispensables.

La programación de la parte de comunicación no se puede realizar con esta herramienta, solamente se puede realizar mediante codificación directa como se indica en la sección 4.5.

Figura 7. Vista de la aplicación usando Flow Design

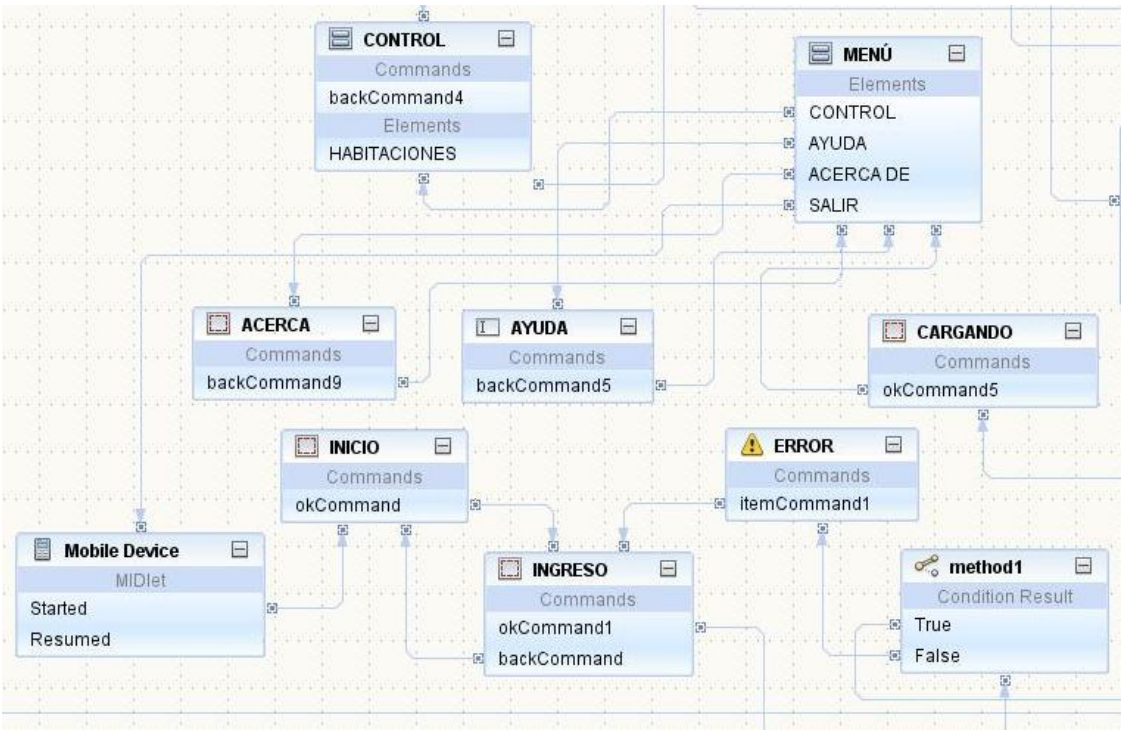
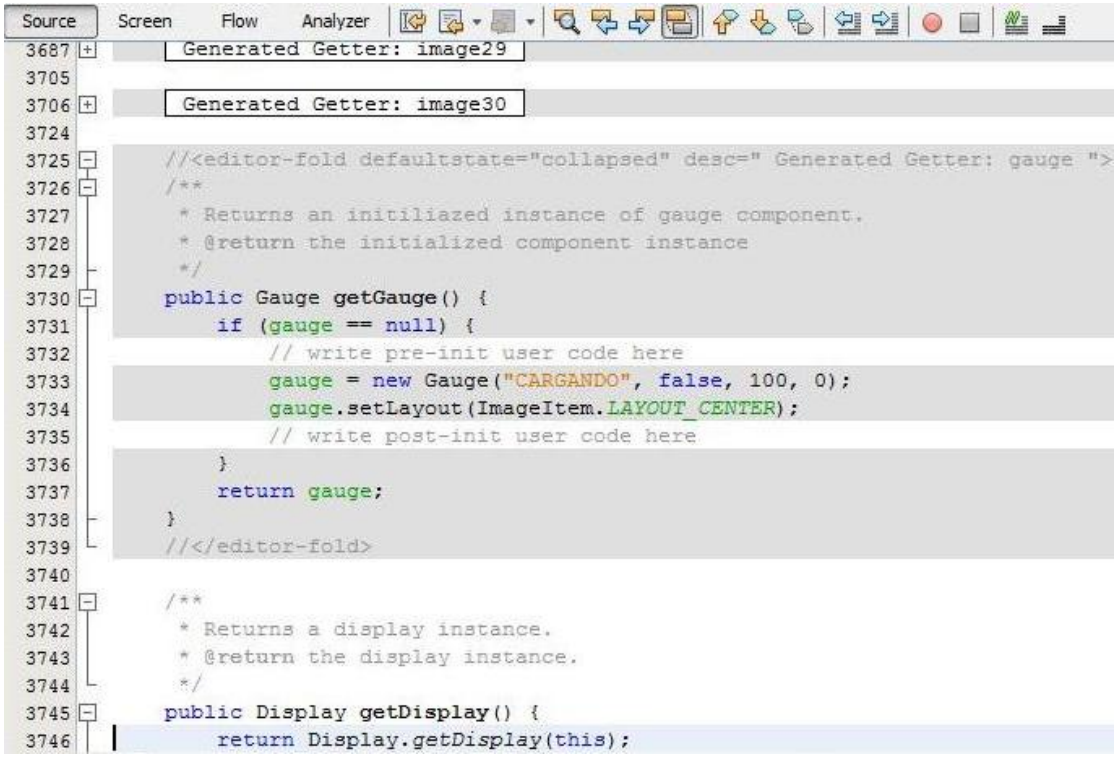


Figura 8. Vista del código generado por Flow Design



4.5 PROGRAMACIÓN DE LA INTERFACE DE COMUNICACIÓN

Un MlDlet puede establecer diversos tipos de conexiones: Sockets, http, https, datagramas, bluetooth, entre otras.

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 serial, SPP.

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

- Crear una conexión servidora
- Especificar los atributos de servicio
- Aceptar las conexiones clientes

La aplicación Bluetooth deberá realizar las siguientes funciones:

- **Búsqueda de dispositivos.** La aplicación realizará una búsqueda de los dispositivos Bluetooth a su alcance que estén en modo conectable.
- **Búsqueda de servicios.** La aplicación realizará una búsqueda de servicios por cada dispositivo encontrado.
- **Establecimiento de la conexión.** Una vez encontrado un dispositivo que ofrece el servicio deseado se podrá realizar la conexión.
- **Comunicación.** Ya establecida la conexión es posible leer y escribir sobre ésta.

4.5.1 Búsqueda de dispositivos

4.5.1.1 BCC (Bluetooth Control Center)

El BCC[12] es un conjunto de capacidades que permiten al usuario o al fabricante resolver peticiones conflictivas de 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.

El BCC define las características de seguridad. Se encargará de manejar el modo de seguridad que la pila debe usar y mantendrá las listas de dispositivos seguros. El API de Bluetooth permite a la aplicación especificar sus requerimientos de autenticación y encriptación.

[12] OVERVIEW (JSR 82 Bluetooth API and OBEX API). Package javax.bluetooth. [En línea]. 2006. [Citado el 14 de Marzo de 2012]. <Disponible en http://docs.oracle.com/javame/config/cldc/opt-pkgs/api/bluetooth/jsr082/index.html >

La librería 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.

4.5.1.2 Habilitación del dispositivo local

Los objetos Bluetooth esenciales en una conexión son *LocalDevice* y *RemoteDevice*. La clase LocalDevice provee acceso y control del dispositivo local. Está diseñada para cumplir con los requerimientos del Generic Access Profile, GAP, definidos para Bluetooth.

La clase RemoteDevice representa un dispositivo remoto y provee métodos para obtener información de dicho dispositivo remoto.

De este objeto se obtiene alguna información de interés, como 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()*.

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.

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 celular.

4.5.1.3 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 interface que es notificada cuando un nuevo dispositivo es encontrado después de haber lanzado un proceso de búsqueda. Esta interface es **javax.bluetooth.DiscoveryListener.**

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

- public void deviceDiscovered(RemoteDevice rd, DeviceClass c)
- public void inquiryCompleted(int c)

Cada vez que se descubre un dispositivo se llama al método *deviceDiscovered(RemoteDevice rd, DeviceClass c)* y 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.

El método *inquiryCompleted(int c)* es llamado cuando la búsqueda de dispositivos ha finalizado y pasa un argumento entero indicando el motivo de la finalización.

La clase **javax.bluetooth.RemoteDevice**, 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.

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.

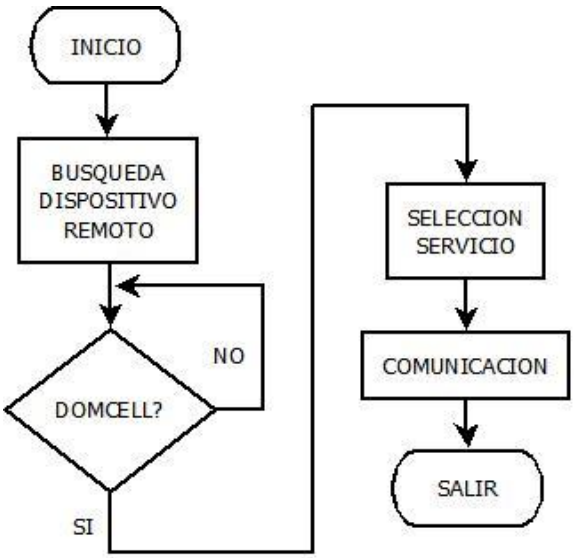
4.5.2 Búsqueda de servicios

En un área de cobertura se pueden encontrar otros dispositivos Bluetooth además 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**.

4.5.2.1 Selección del dispositivo de control

En el programa se lleva a cabo una comparación entre el Friendlyname del dispositivo encontrado y el que se espera: “DOMCELL”, si concuerda, se aceptará la comunicación con este dispositivo, se realizará la búsqueda de servicios y posteriormente se enviará la información de control. En la Figura 9 se observa el diagrama de flujo donde se realiza la comparación del FriendlyName de cada dispositivo encontrado con el esperado.

Figura 9. Detección y selección de dispositivos



4.5.2.2 Descubrimiento de servicios

Para realizar una búsqueda de servicios también se usa la clase *DiscoveryAgent* y se implementa la interfaz *DiscoveryListener*. 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:

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

El primer argumento es un arreglo de enteros con el que se especifica los atributos de servicio que interesan. El segundo argumento es un arreglo de identificadores de servicio. Permite especificar los servicios en los que el cliente está interesado. La clase UUID (Universally Unique Identifier) representa identificadores únicos universales. Cabe resaltar que el valor de UUID usado en el presente proyecto es “0x1101” que es el correspondiente para el Perfil de Puerto Serial, 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.

- **Método servicesDiscovered (int transID, ServiceRecord[ ] servRecord)**

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 arreglo 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.

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

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().

4.5.3 Establecimiento de la conexión

Una vez que la aplicación Bluetooth ya ha realizado la búsqueda de los dispositivos Bluetooth a su alcance, seleccionará al módulo Bluetooth y buscará información de sus servicios. Luego se establecerá la conexión y posteriormente se transmitirán los datos.

4.5.3.1 Comunicación cliente-servidor

El paquete javax.bluetooth permite usar dos mecanismos de conexión: SPP y L2CAP. En este caso se usará el SPP -Perfil de Puerto Serial-. Mediante SPP se obtiene un *DataInputStream* y un *DataOutputStream*.

Para abrir una conexión se usará 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 la clase 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.

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:

- **ServiceRecord.NOAUTHENTICATE_NOENCRYPT:** No se requiere ni autenticación ni cifrado.
- **ServiceRecord.AUTHENTICATE_NOENCRYPT:** Requiere autenticación, pero no cifrado.
- **ServiceRecord.AUTHENTICATE_ENCRYPT:** 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).

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

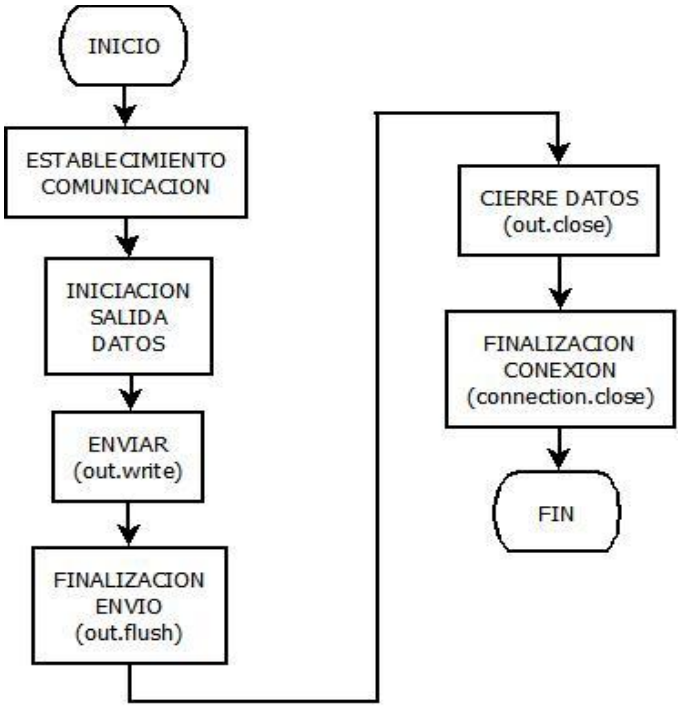
Una vez que se tiene la URL, se utiliza el método *Connector.open()* para realizar la conexión. Este método devuelve un objeto distinto según el tipo de protocolo usado. En el caso de un cliente SPP devolverá un *StreamConnection*. A partir del *StreamConnection* se obtienen los flujos de entrada y de salida:

```
StreamConnection con = (StreamConnection) Connector.open(url);
DataOutputStream out = con.openDataOutputStream();
DataInputStream in = con.openDataInputStream();
```

4.5.3.2 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. En la Figura 10, se muestra el diagrama de flujo del proceso para enviar una cadena de texto sobre una conexión SPP.

Figura 10. Envío de datos



El comando a transmitir está compuesto por un encabezado y una trama de datos, lo anterior para diferenciarla de cualquier otra información que pueda llegar al módulo.

El encabezado estará definido por el símbolo #, identificando el inicio de la trama de datos, la cual puede ser un carácter tipo letra minúscula (**a** - **z**) o de tipo numérico decimal (**0** - **9**), activando según el tipo de carácter, una función específica; todo lo anterior de la siguiente forma:

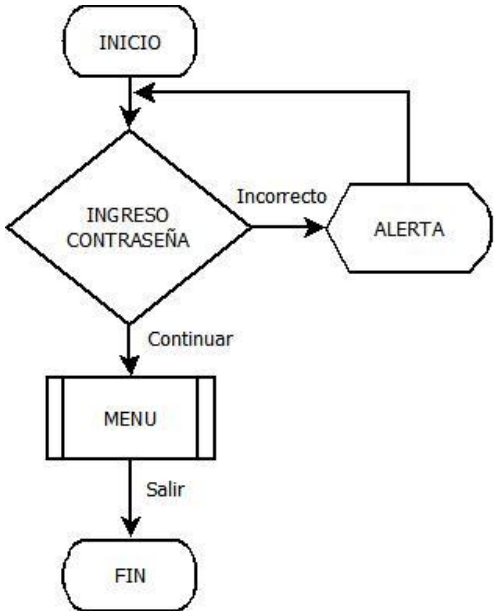
Encabezado ⟸ **#0** ⟹ Dato

Encabezado ⟸ **#a** ⟹ Dato

4.5.3.3 Control de acceso

En la Figura 11, se muestra el diagrama de flujo de la implementación del control de acceso, con un nivel de seguridad establecido por medio de una contraseña, consistente en un cuadro de texto en el que el usuario ingresará una cadena de texto alfanumérico para que sea comparada con la que se tiene guardada: “3456”.

Figura 11. Diagrama de flujo de inicio y acceso al programa



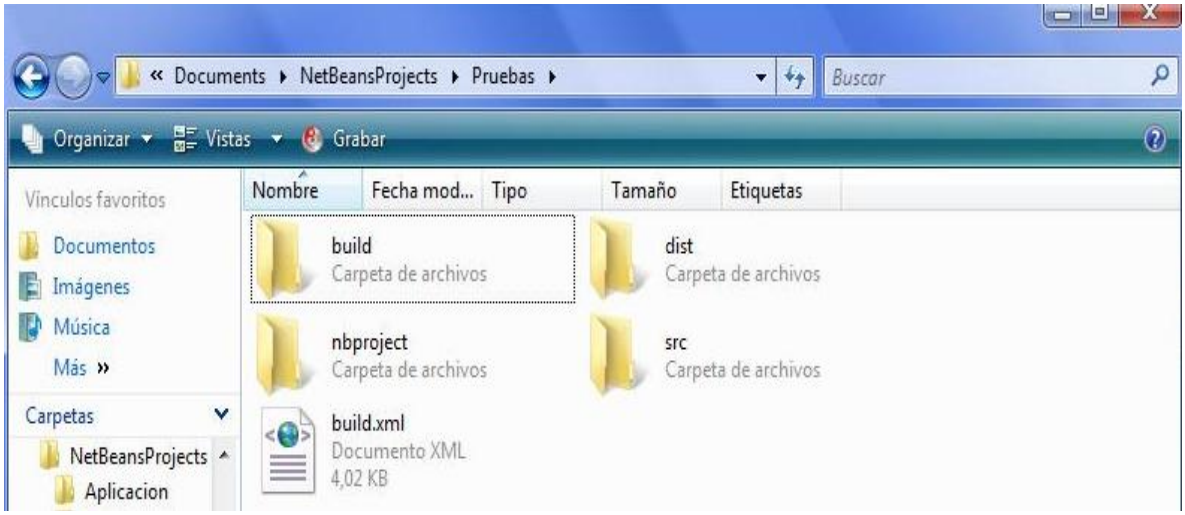
4.6 COMPILACIÓN

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

Para usar cualquiera de estas herramientas, es imprescindible tener instalada una Máquina Virtual Java en el computador, usada por cualquier compilador del lenguaje Java.

Sin importar el compilador que se use, las aplicaciones Java estarán organizadas en proyectos, y el resultado de un proyecto será el MIDlet. Los proyectos están organizados con una estructura de directorios donde se puede encontrar los archivos generados, como se observa en la Figura 12.

Figura 12. Imagen directorio del proyecto



En la Tabla 5, se resume el contenido de las carpetas generadas en un proyecto.

Tabla 5. Estructura de directorios de un proyecto Java

CARPETA	DESCRIPCIÓN
bin	Aquí se guardan los archivos JAD y JAR cuando el proyecto es empaquetado. Esta carpeta contiene también el archivo de manifiesto.
classes	Esta carpeta es usada por el compilador para almacenar los archivos de clases .class.
lib	Aquí se guardan las librerías usadas en el proyecto.
res	Las imágenes, sonido y más recursos van en esta carpeta. Luego estos son empaquetados dentro del archivo JAR.
src	En esta carpeta se encuentran los códigos fuente, en archivos .java.
tmpclasses	En esta carpeta, el compilador crea archivos temporales usados para la simulación y depuración.
tmpsrc	En esta carpeta, el compilador crea archivos temporales usados para la simulación y depuración.

4.7 DEPURACIÓN Y EJECUCIÓN

Una vez que se ha completado el proceso de compilación, el MIDlet se puede ejecutar en un simulador de teléfonos móviles, como el mostrado en la Figura 13.

Figura 13. Modelos de teléfono para las simulaciones

Fuente: http://celularesblog.com/tag/celular-nokia

En fases de prueba se deberá ejecutar el MIDlet sobre un simulador. Se puede hacer a través del comando Run con el botón F5 en el software NetBeans.

4.8 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 que contiene las clases (.*class*) que ha generado la compilación de nuestro programa, y también 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. El Gestor de Aplicaciones puede usar este archivo para obtener información útil sobre el MIDlet. Este archivo define una serie de atributos, y además es opcional, y por ende no es necesario copiarlo al dispositivo móvil en el momento de la instalación del programa.

4.9 INSTALACIÓN DE LA APLICACIÓN EN EL CELULAR

El proceso de empaquetamiento genera los archivos JAD, JAR y manifiesto 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 celulares, similares a los juegos Java que están bastante difundidos actualmente.

La instalación consiste en copiar los archivos JAR al teléfono celular 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.

5. DISEÑO DEL HARDWARE DE CONTROL

En ésta sección se resume el diseño y la construcción de un circuito electrónico encargado de realizar el control de las distintas funciones de un hogar desde el teléfono, es decir, el circuito de control para los dispositivos eléctricos.

Este hardware consiste en un circuito electrónico que incluye el módulo Bluetooth, un módulo de control (microcontrolador) y las interfaces necesarias para realizar el control de dispositivos eléctricos. Éste hardware es el encargado de aceptar la conexión Bluetooth solicitada por el dispositivo móvil, y una vez establecida la conexión, recibir las instrucciones a transmitir para realizar las acciones de control requeridas.

Hasta el momento, se ha desarrollado e instalado la aplicación Java en el teléfono celular, con lo que el dispositivo móvil está listo para conectarse vía Bluetooth con el módulo de control.

El circuito electrónico se divide en tres partes:

- Circuitos de Alimentación.
- Circuito adaptador de Módulo Bluetooth.
- Módulo de Control (Microcontrolador).

5.1. CIRCUITOS DE ALIMENTACIÓN

Este proyecto se centra sobre el Módulo Bluetooth KC-5100, el cual requiere un voltaje de alimentación de 3.3V y maneja niveles lógicos TTL[13]. Además, necesita la implementación de un circuito adaptador de niveles RS232 a 3.3V, debido a que este módulo permite comunicación directa con un computador a través de puerto serial, para lo cual se utiliza el MAX3225[14].

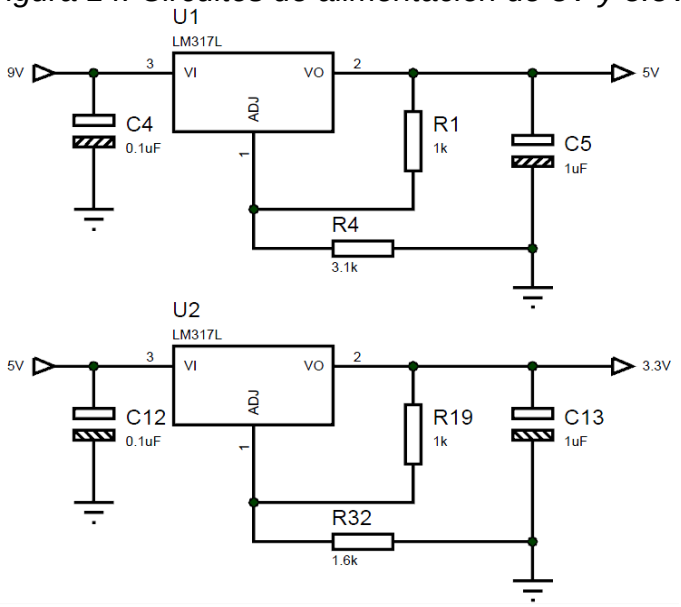
El modulo de control se basa en el uso del PIC 18F4550, que requiere 5V de voltaje de alimentación, así como también lo requiere la interface de control. De acuerdo a esto, se realizó el diseño del circuito de alimentación -véase Figura14-, usando dos reguladores de voltaje ajustable LM317, cada uno designado para entregar 5V y 3.3V respectivamente, y con base en la información del fabricante[15].

[13] KC WIREFREE, We Make Bluetooth Work For You. [En Línea]. [Citado el 14 de Marzo de 2012]. <Disponible en http://kcwirefree.com/docs/KC5100_Datasheet.pdf >

[14] MAXIM, Innovation Delivered. [En Línea].[Citado el 14 de Marzo de 2012].<Disponible en http://datasheets.maxim-ic.com/en/ds/MAX3224-MAX3245.pdf>

[15] FAIRCHILD SEMICONDUCTOR, Solutions For Your Success. [En Línea]. [Citado el 14 de Marzo de 2012]. <Disponible en http://www.fairchildsemi.com/ds/LM/LM317.pdf>

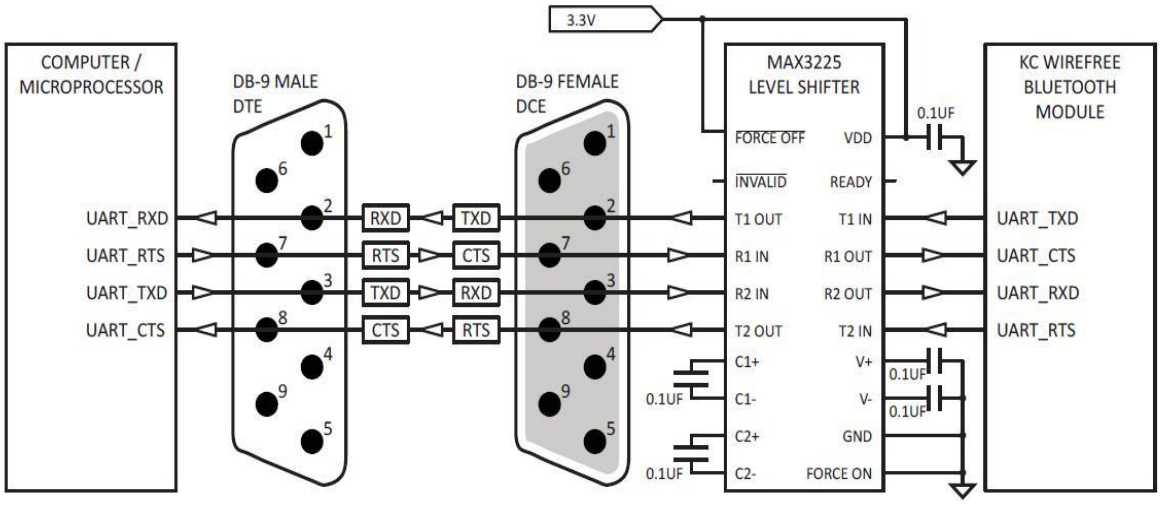
Figura 14. Circuitos de alimentación de 5V y 3.3V



5.2. CIRCUITO ADAPTADOR DE MÓDULO BLUETOOTH

Como se mencionó anteriormente, el modulo Bluetooth KC-5100 permite comunicación directa con el usuario a través del puerto serial de un computador, para lo cual requiere de la implementación de un circuito adaptador de niveles RS232 a 3.3V, como se observa en la Figura 15.

Figura 15. Conexión UART para módulo KC-5100 con adaptador de niveles



Fuente: http://www.kcwirefree.com/docs/KC5100_Datasheet.pdf

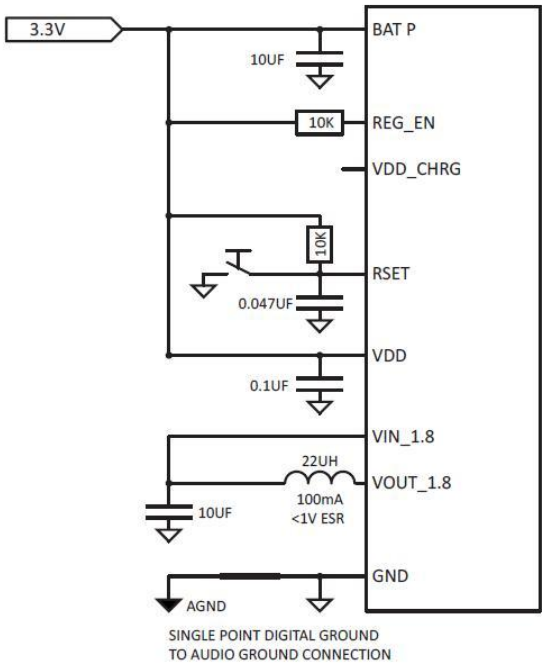
También es necesario un conector DB9 hembra para realizar la comunicación a través del puerto serial del computador. De éste conector, solo se usarán las conexiones básicas indicadas en la Tabla 6.

Tabla 6. Conexiones básicas del conector DB9 hembra

NOMBRE	TERMINAL	FUNCION
RX	2	Recepción de datos desde el MAX3225
TX	3	Transmisión de datos hacia el MAX3225
RTS	7	Requerimiento de envío
CTS	8	Borrar para envío
GND	5	Nivel de tierra

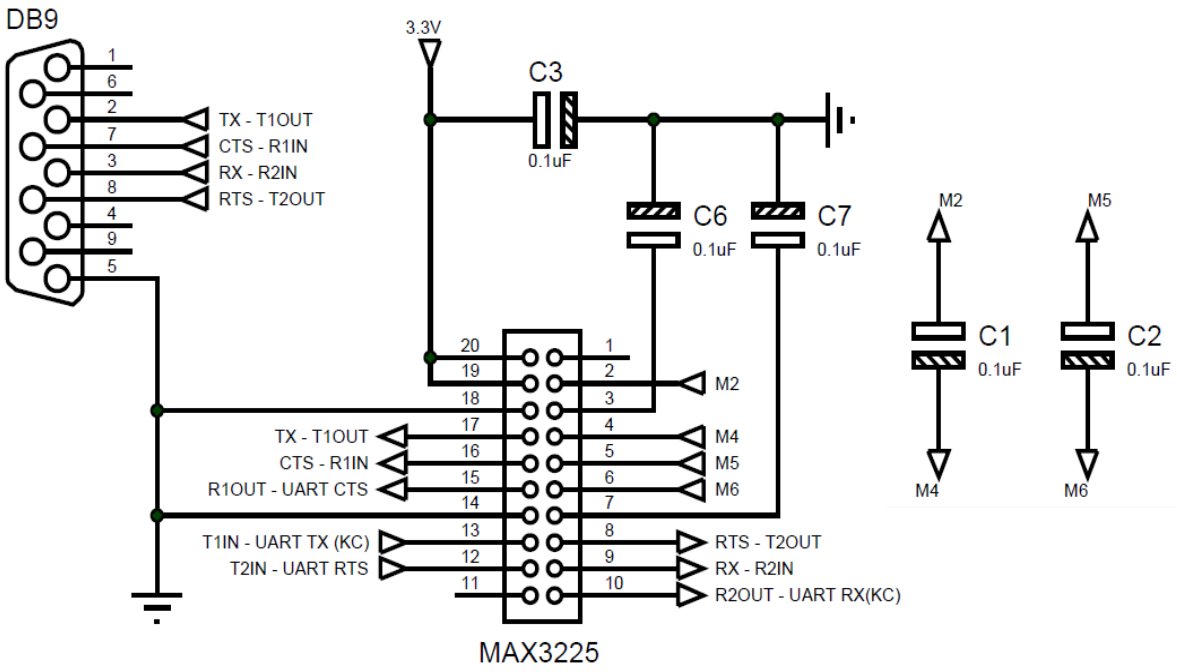
Además de la configuración antes mencionada, el modulo Bluetooth requiere conexiones adicionales para funciones adicionales (reset, modo análogo, etc.) mostradas en la Figura 16, completando así el diseño del circuito adaptador del módulo Bluetooth (véase Figura 17).

Figura 16. Conexión a alimentación regulada de 3.3V



Fuente: http://www.kcwirefree.com/docs/KC5100_Datasheet.pdf

Figura 17. Circuito adaptador de módulo Bluetooth



5.3. MÓDULO DE CONTROL

El módulo de control -véase Figura 18-, implementado a través de un PIC 18F4550, es el encargado de procesar y ejecutar las instrucciones recibidas por parte del módulo Bluetooth. La transmisión de instrucciones desde el módulo Bluetooth se realizó mediante el puerto UART del PIC, y se observa en la Figura 19.

Estas instrucciones -que pueden ser recibidas por conexión serial o interface Bluetooth- activan o desactivan determinados pines de salida del PIC. Además poseen un proceso de selección, y posterior control de información, éste último mediante encabezados, para reconocimiento de las instrucciones válidas por parte del microcontrolador.

El microcontrolador tiene indicadores tanto de activación, como de transmisión de información, con los cuales se verifica el correcto funcionamiento de esta sección del circuito.

Figura 18. Circuito módulo de control

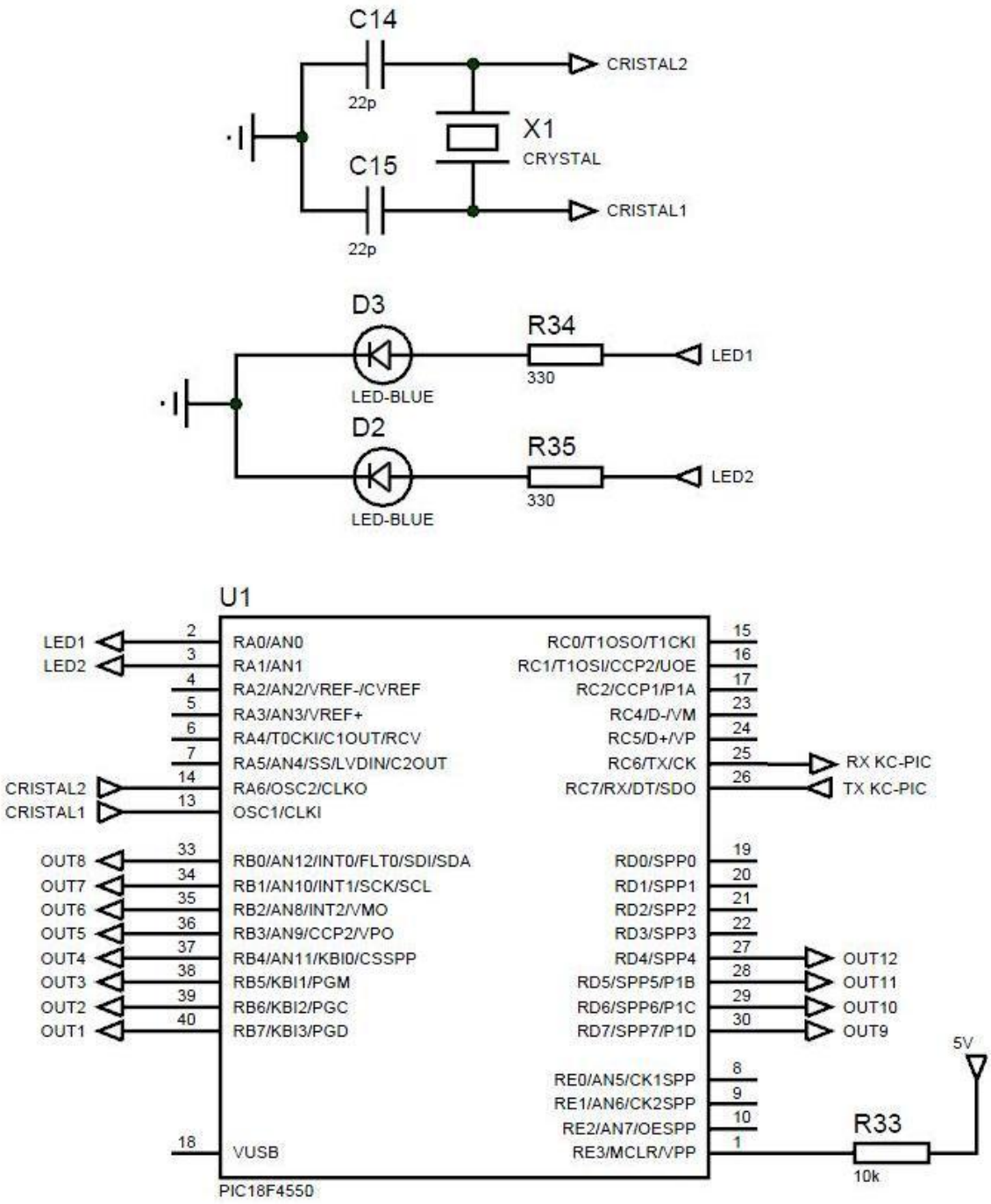
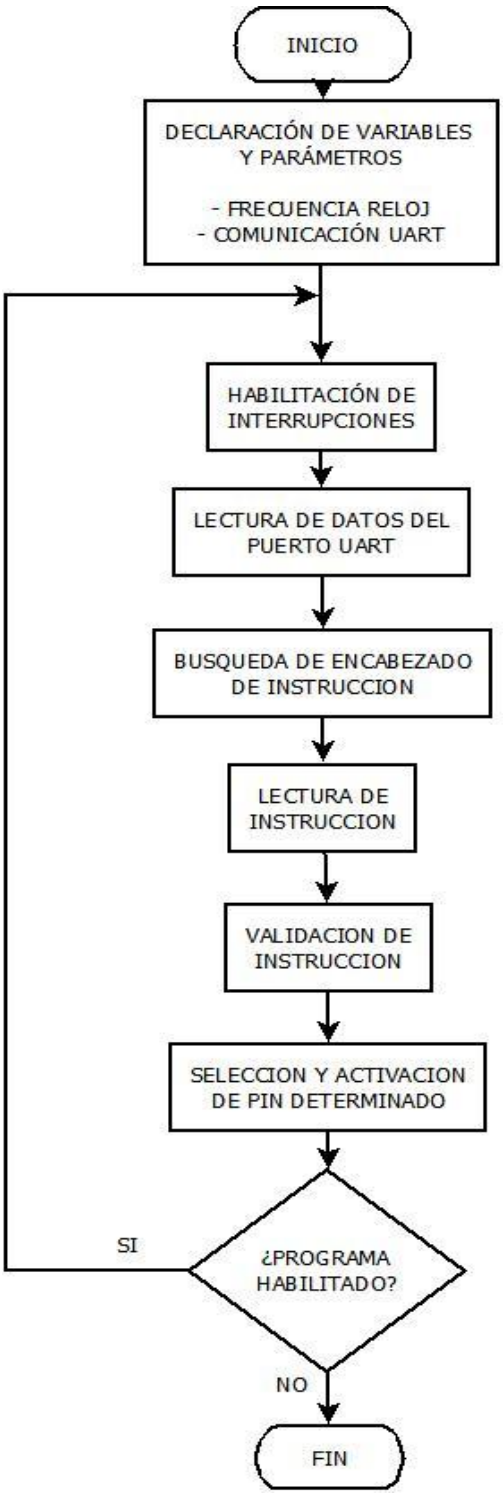


Figura 19. Diagrama de flujo del programa del microcontrolador



6. IMPLEMENTACION Y RESULTADOS

Esta sección incluye el proceso de implementación del hardware de control diseñado y la aplicación Java para realizar el control de las funciones de una vivienda, así como también los resultados presentados de las pruebas realizadas.

6.1 TARJETA DE CONTROL

Para el diseño de la tarjeta se utilizaron los siguientes parámetros:

- 10 salidas análogicas con un manejo de corriente máximo de 15 mA $^{c}/_{u}$.
- 5 salidas análogas con un manejo de corriente máximo de 50 mA $^{c}/_{u}$.
- Frecuencia de 20 MHz para trabajar desde 110 Bps hasta 115,2 KBps
- Comunicación inalámbrica a través del modulo KC-5100.
- Comunicación serial a través del puerto serial.

El diseño e implementación de la tarjeta -véase Figura 20 y Figura 21 respectivamente- se realizó utilizando el programa EAGLE (Easily Applicable Graphical Layout Editor), versión 6.1 para Windows 7.

6.2 CONFIGURACION DEL MÓDULO BLUETOOTH

Para el correcto funcionamiento del módulo KC-5100, es necesario establecer los parámetros de comunicación serial del Hyperterminal (o cualquier otro software para manejo de comunicación serial por computador), y de esta manera, realizar una configuración inicial de las propiedades y características que posee el módulo (nombre, tasa de baudios, etc.).

Al conectar el módulo mediante cable serial a un computador, se debe tener en cuenta que el Hyperterminal debe tener los parámetros de conexión[16], los cuales están indicados en la Tabla 7.

[16] KC WIREFREE, KC Wirefree Getting Started Guide. [En Línea]. [Citado el 15 de Marzo de 2012]. <Disponible en http://www.ylib.com.cn/products/Bluetooth/doc/KC_GettingStartedGuide.pdf>

Figura 20. PCB de la tarjeta de control

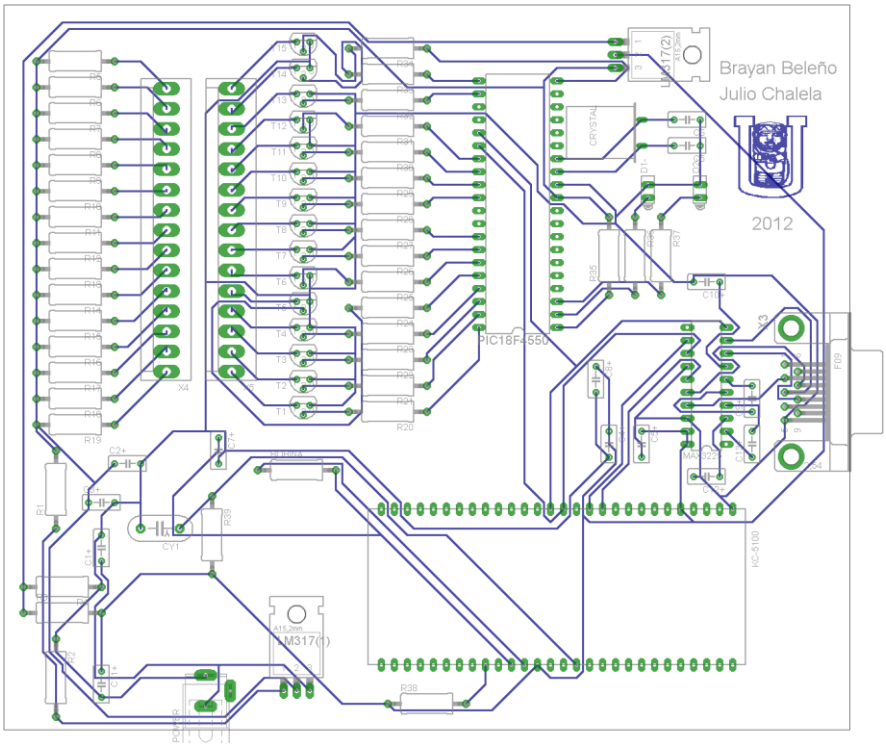


Figura 21. Módulo de control

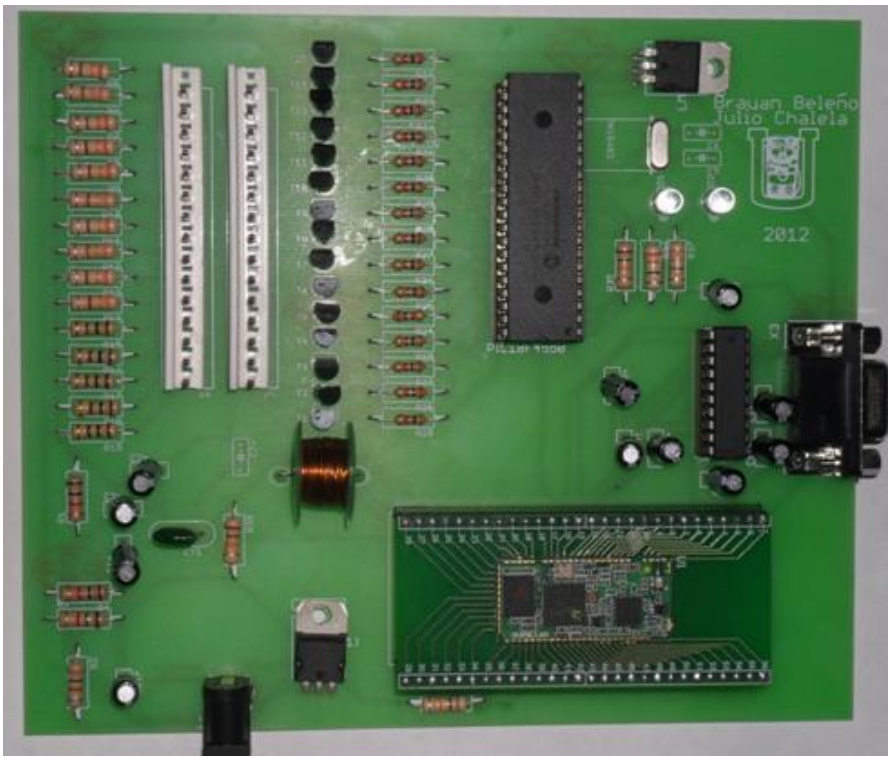


Tabla 7. Parámetros de comunicación serial del módulo KC-5100

PARÁMETRO	VALOR POR DEFECTO
Bits por segundo	1500
Bits de datos	8
Paridad	Ninguno
Bits de parada	1
Control de flujo	Hardware

Establecida la conexión, es posible configurar el módulo usando comandos AT provistos por el fabricante[17]. De esta manera, se modificó el nombre del módulo y la velocidad de transmisión, como se observa en la Tabla 8; características que no cambiaran aunque el módulo sea reiniciado, y por tanto, permiten su uso permanente, tanto para búsqueda del dispositivo, como para conexión serial respectivamente.

Tabla 8. Comandos AT para configuración inicial del módulo KC-5100

COMANDO AT	VALOR
AT+ZV DefaultLocalName	DOMCELL
AT+ZV ChangeDefaultBaud	9600

6.3 CONFIGURACIÓN DE LOS TERMINALES DEL PIC

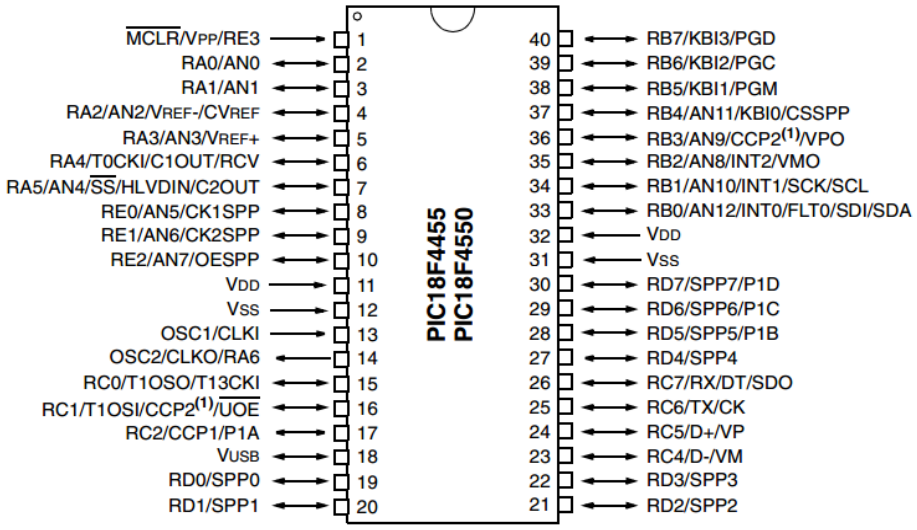
Para realizar el control de los dispositivos se hizo uso de los llamados *Terminales de Propósito General* (General Purpose Input-Output), con que cuenta el microcontrolador PIC 18F4550 -véase Figura 22-. Se están usando quince de los treinta y tres terminales disponibles: Puerto B y Puerto D.

Se puede utilizar cualquier puerto, pero se escogieron éstos porque la posición que tienen en el módulo facilitó el diseño del circuito impreso.

Los puertos del PIC pueden ser usados como entradas o como salidas. Como salidas, éstos pueden ser configurados con nivel alto o nivel bajo. Las condiciones iniciales predefinidas de los puertos utilizados en este proyecto evitarán conflictos en el caso de un eventual reinicio del circuito, ya que de esta manera se asegura un estado fijo y predefinido al iniciar.

[17] KC WIREFREE, KC Serial v2.4 User Guide. [En Línea]. [Citado el 15 de Marzo de 2012]. <Disponible en http://kcwirefree.com/docs/KCSerialUserGuide.pdf>

Figura 22. Diagrama PIC 18F4550



Fuente: http://www.microchip.com/downloads/en/devicedoc/39632c.pdf

Las condiciones iniciales de los terminales y las especificaciones técnicas se muestran en la Tabla 9.

Tabla 9. Configuración y especificaciones de los terminales del PIC

TERMINAL	CONFIGURACIÓN INICIAL	ESTADO INICIAL	CORRIENTE (mA)
RB0	OUT	ON	15
RB1	OUT	OFF	15
RB2	OUT	OFF	15
RB3	OUT	OFF	15
RB4	OUT	OFF	15
RB5	OUT	OFF	15
RB6	OUT	OFF	15
RB7	OUT	OFF	15
RD1	OUT	OFF	50
RD2	OUT	OFF	50
RD3	OUT	OFF	50
RD4	OUT	OFF	50
RD5	OUT	OFF	50
RD6	OUT	OFF	15
RD7	OUT	OFF	15

En este caso los terminales serán configurados como salidas, de este modo se pueden establecer en estado alto o bajo y así controlar los dispositivos eléctricos conectados.

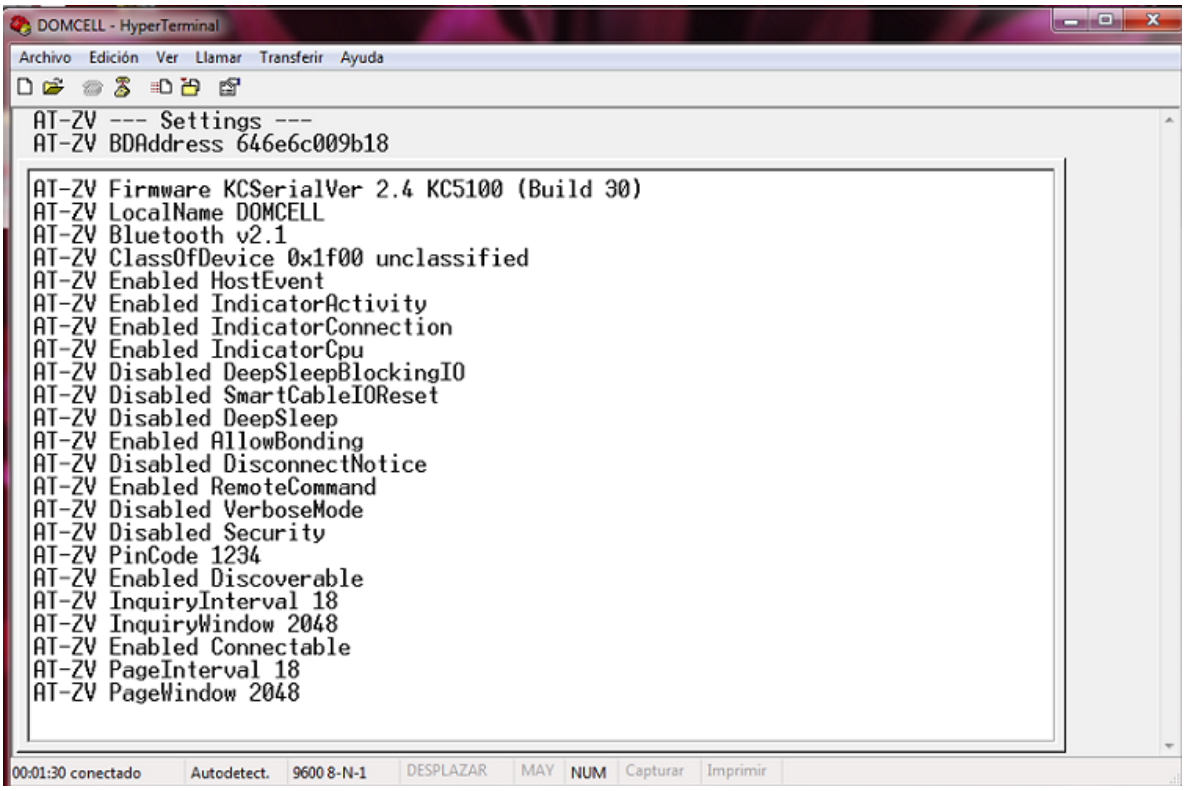
6.4 CONTROL DE DISPOSITIVOS VÍA CONEXIÓN SERIAL

Una vez realizado el programa para el microcontrolador PIC18F4550 en el compilador CCS, se realizó su respectiva conexión con el modulo de control configurando en Hyperterminal los parámetros comunicación mostrados en la Figura 23.

Figura 23. Parámetros de comunicación en Hyperterminal

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

Figura 24. Configuración de parámetros del módulo y comandos



En la Figura 24, se observa la pantalla del Hyperterminal con los comandos de configuración inicial del módulo, los comandos enviados vía conexión serial y sus respectivas respuestas. Los comandos mostrados corresponden a los usados para el control de los diversos elementos eléctricos conectados al módulo de control.

Aparte del control vía Bluetooth, se tiene la alternativa del control vía conexión serial. Se hicieron pruebas y los dispositivos eléctricos pueden ser controlados simultáneamente de ambas formas sin presentar inconvenientes.

6.5 CONTROL DE DISPOSITIVOS VÍA CONEXIÓN BLUETOOTH

Para realizar las pruebas de funcionamiento se utilizó el teléfono Nokia 2630 -véase Figura 25-, un celular de gama media-baja que cuenta con las características necesarias. La administración de archivos e instalación de aplicaciones para este tipo de celulares se realiza mediante Bluetooth. En este caso, se realizó la transferencia del archivo DOMCELL.JAR y automáticamente se instaló la aplicación.

Figura 25. Nokia 2630

Fuente: http://abc-economia.com/2011/03/09/nokia-recibira-de-microsoft-1-000-millones-de-dolares-por-pasarse-a-windows7/celular-nokia-2630/

Una vez en el teléfono, la aplicación apareció con el nombre DOMCELL. Se ejecutó y se empezó a usar. Cuando se abrió la aplicación, esta pidió confirmar la activación del Bluetooth. Una vez autorizado, se muestra la pantalla de inicio de la aplicación y, automáticamente el teléfono empezó una búsqueda a su alrededor y pudo encontrar al Módulo Bluetooth. Éste procedimiento es transparente para el usuario, tarda alrededor de 8 segundos hasta que se complete la búsqueda y se puedan mostrar los dispositivos encontrados.

6.5.1 Búsqueda

El software instalado en el celular realiza una lista con todos los dispositivos encontrados en la búsqueda realizada, dentro de la cual busca “DOMCELL” - el módulo Bluetooth- y accede a los servicios que éste nos proporciona. Este proceso de búsqueda es realizado de manera transparente para el usuario, de modo que no es necesario realizar ningún tipo de interface o formulario para éste.

6.5.2 Control de acceso

Al escoger la opción “Continuar” en la pantalla de inicio de la aplicación, apareció un formulario donde hay un cuadro de texto para ingresar una contraseña, como se muestra en la Figura 26. Se debe ingresar una contraseña y luego seleccionar la opción “OK”. Cuando la contraseña no se ingresó correctamente, apareció una alerta indicando que no se puede continuar, y luego regresa al mismo formulario para realizar un nuevo intento. No está definido un número máximo de intentos. La contraseña es una cadena de texto simple, no cifrado. Los valores ingresados son comparados por la aplicación con un texto único incluido en el código fuente.

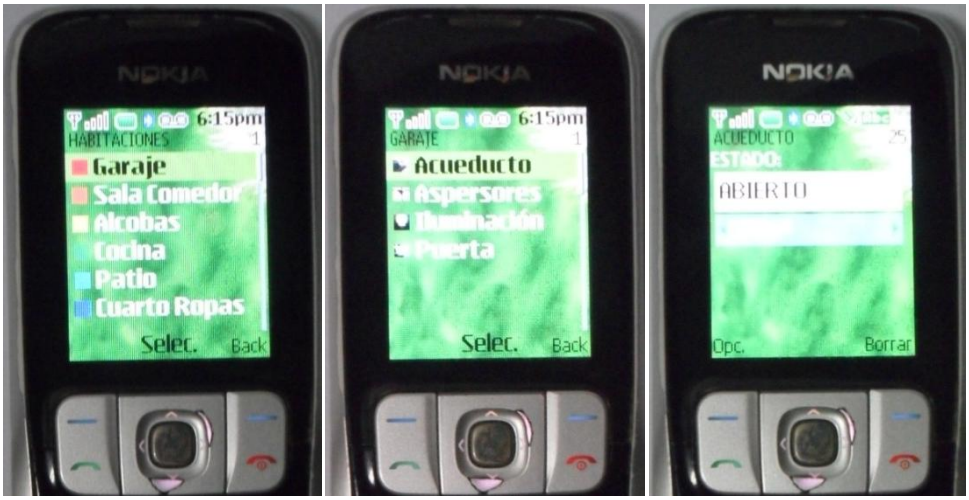
Cuando la contraseña se ingresó correctamente (“3456”) y se escogió la opción “OK”, se obtuvo el acceso a la aplicación y ésta continuó hacia el menú de control.

Figura 26. Control de acceso

6.5.3 Control de dispositivos

Para permitir el control de cada uno de los dispositivos, se debe ingresar a cada elemento en la sección de habitaciones en el menú de control (véase Figura 27). Una vez ubicado el dispositivo eléctrico, se muestran las posibles opciones con las cuales se cuenta: ENCENDER y APAGAR.

Figura 27. Control de dispositivos



Al seleccionar cada opción, ENCENDER o APAGAR, se envía el comando especificado para este dispositivo. Se comprobó también que es posible controlar los dispositivos vía serial y vía Bluetooth al mismo tiempo.

6.5.4 Ayuda

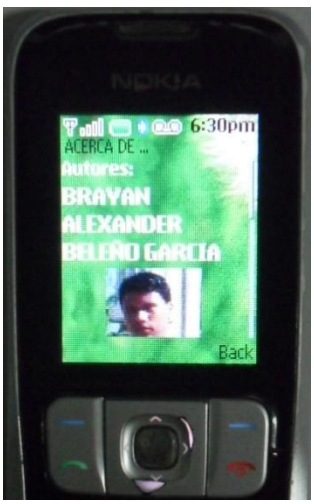
Este formulario -véase Figura 28- se encuentra ubicado en el menú de control. Allí se encuentra la información necesaria con la cual el usuario podrá obtener las instrucciones para realizar el control de los dispositivos.

6.5.5 Acerca de

Este formulario -véase Figura 29- se encuentra ubicado en el menú principal. Allí se incluye información del proyecto y algunos datos de los creadores de la aplicación.

Figura 28. Ayuda

Figura 29. Acerca de



6.6 RESULTADOS

La presentación y el funcionamiento de un MIDlet podrían variar de un modelo a otro, dependiendo del sistema operativo que use, de las características de la pantalla, teclado, etc. El menú principal de la aplicación ejecutándose en un teléfono Nokia 2630 se aprecia en la Figura 30.

Figura 30. Menú principal

Las primeras pruebas se realizaron con el celular junto al módulo de control. El tiempo de búsqueda de dispositivos es de aproximadamente 8 segundos. Los comandos se pueden enviar las veces que el usuario lo desee, siempre y cuando todos se realicen dentro de la misma conexión.

Se puede usar más de un teléfono celular a la vez. Se probó simultáneamente la conexión vía serial y vía Bluetooth, y los comandos se ejecutaron en el mismo orden en que se enviaron, respetando el tiempo de respuesta entre cada tipo de conexión.

Se fue aumentando progresivamente la distancia entre el dispositivo móvil y el módulo de control sin presentar alguna variación considerable en los tiempos de búsqueda y de envío. Se logró alcanzar hasta 30 metros sin obstáculos. Pasada esta distancia la búsqueda tomaba un poco más de tiempo y muchas veces el módulo de control, “DOMCELL”, no era detectado. Si en algún momento se lograba detectar, al seleccionar el comando se demoraba más de lo usual y muchas veces no se concretaba la operación.

Tomando en cuenta que la frecuencia de trabajo de Bluetooth es 2,4GHz, también usada por otros aparatos comunes para una casa como teléfonos inalámbricos, hornos microondas, equipos wireless, entre otros, se probó cerca de los mismos y los resultados fueron similares a los obtenidos en pruebas anteriores. Por consiguiente, no se observaron fallas por interferencia.

Generalmente, todas las pruebas se realizaron teniendo conectado un computador al módulo para ver en la pantalla del Hyperterminal las respuestas a los comandos, de esta manera se pudo saber si se ejecutaron correctamente o no; aunque los resultados fueron evidentes en los dispositivos eléctricos.

CONCLUSIONES

- Con la realización de este proyecto, se ha dado un uso adicional a la tecnología Bluetooth integrada en los teléfonos celulares demostrando así que la plataforma Java J2ME especializada en dispositivos de bajo recursos se puede utilizar para el control de un entorno domótico.

- El desarrollo de la aplicación para el dispositivo móvil fue posible gracias a la versatilidad y la gran ventaja de contar con software libre Java; por tal motivo este producto puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

- Se comprobó que la tecnología Bluetooth con frecuencia de trabajo de 2.4GHz utilizada en este proyecto no se ve afectada por la interferencia de dispositivos eléctricos como: hornos microondas, teléfonos inalámbricos y los diversos sistemas IEEE 802.15 que se pudieran presentar en el ambiente del hogar.

- La tecnología Bluetooth es muy difundida en los teléfonos celulares y cada día más modelos la incluyen; por consiguiente, este proyecto cuenta con gran campo de aplicación.

- La tecnología Bluetooth en comparación con otras tecnologías inalámbricas para aplicaciones domóticas, como infrarrojo, Wi-fi e inclusive ZigBee, posee muchas ventajas ya que al utilizar un dispositivo de control como el teléfono celular, esta viene incluida con el artefacto.

- Ya que la comunicación implementada se realiza en una sola vía, debido a que el teléfono no recibe respuestas por parte del módulo Bluetooth del estado de los dispositivos eléctricos o del cumplimiento de la orden enviada; al considerar el alcance de la aplicación, se concluye que para las distancias cortas dentro de la vivienda, es suficiente que el usuario observe directamente los resultados esperados.

- Se comprobó, a pesar de que las especificaciones técnicas del fabricante del módulo Bluetooth radio Clase 1 es de 100m, la distancia real está alrededor de 30m en condiciones normales y hasta 10m atravesando grandes objetos; confirmando así que la aplicación servirá para los propósitos que fue desarrollada.

- Las condiciones iniciales del módulo Bluetooth y del microprocesador PIC, evitan tener conflictos con los terminales de control. Por tal motivo, en caso de un eventual reinicio, los elementos utilizados siempre estarán en un estado único y fijo programado desde un comienzo de acuerdo a las necesidades de cada hogar, asegurando su funcionalidad.

- El módulo de control solo puede controlar los dispositivos conectados en dos estados: apagado y encendido. Por consiguiente, es muy importante limitar el tipo de artefactos eléctricos que pueden ser manejados.

- Se ha desarrollado un sistema sencillo y de gran utilidad, esto pensado en la comodidad y bienestar de las personas y aun más para aquellas con algún tipo de discapacidad o que por algún motivo no pueden usar los controles convencionales dentro de una vivienda.

- El funcionamiento del sistema es satisfactorio y se cumplen todos los objetivos propuestos en la realización del proyecto. Por lo tanto, se comprueba que la tecnología 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.

RECOMENDACIONES

- Es recomendable el uso de lenguaje de programación Java, para la creación de aplicaciones para dispositivos móviles, debido a la versatilidad y a la gran cantidad de equipos existentes en el mercado que lo soportan.

- La presentación de la aplicación puede ser diferente en cada dispositivo móvil, ya que esta depende de las características de cada uno. Por tal motivo, se recomienda modificar la aplicación según las dimensiones de la pantalla, el sistema operativo, la disposición de los botones y otros recursos propios con los que cuenta cada dispositivo en el cual se vaya a instalar el software.

- El desarrollo de este proyecto puede servir de referencia y guía para quien desee continuar ideando este tipo de aplicaciones. Por tal motivo, se recomienda mejorar este sistema para que haga monitoreo y control simultáneamente o hacer alguna modificación para manejar mas de los 2 estados de control que posee.

- En el diseño de sistemas Bluetooth se recomienda usar módulos que cuenten con una interface serial ya que de esta manera se podría apreciar los resultados de los datos enviados y recibidos a través del Hyperterminal de Windows.

- Se recomienda hacer la instalación del módulo de control en un lugar central, para así cubrir un área lo más útil posible, es decir, para ser detectado desde mas lugares dentro de una vivienda.

- Se recomienda usar una UPS para garantizar tanto el funcionamiento de los aparatos eléctricos como el módulo de control.

- Se recomienda especial cuidado en el momento de manipular la tarjeta electrónica hecha para el módulo de control debido a que esta puede sufrir daños por golpes o por mala conexión de los elementos que posee.

- Se recomienda aprovechar todas las entradas/salidas que tiene el módulo de control para aumentar el número de dispositivos controlados dentro de la vivienda evitando así el uso de módulos de control adicionales y de esta manera disminuir los costos.

BIBLIOGRAFIA

BORCHES, Pablo Daniel. Java 2 Micro Edition: Soporte Bluetooth. Madrid: Universidad Carlos III de Madrid. [En línea]. 2004. [Citado el 27 de Marzo]. <Disponible en http://www.it.uc3m.es/celeste/docencia/j2me/tutoriales/bluetooth/EstudioTecnologico1_0.pdf>

BLUEZONA, Especialista Bluetooth. Bluetooth. [En Línea]. 2008. [Citado el 13 de Marzo de 2012]. <Disponible en http://www.bluezona.com/que-es-bluetooth/>

CANO, Jesús, GARCIA, Teresa, SANCHEZ, Pedro. Sistema de acceso bluetooth. Madrid: Universidad Complutense de Madrid. [En línea]. Septiembre 2009. [Citado el 27 de Marzo]. <Disponible en http://eprints.ucm.es/9434/>

CASADOMO, El Portal Del Edificio y Hogar Digital. Domótica. [En Línea]. Marzo 2012. [Citado el 13 de Marzo de 2012]. <Disponible en http://www.casadomo.com/noticiasDetalle.aspx?c=14 >

CHICANGANA, Mary Luz, BASTO, Jairo, HURTADO, Javier. Desarrollo de aplicaciones bluetooth utilizando el API JAVA JSR-82. Popayán: Universidad del Cauca. [En línea]. [Citado el 27 de Marzo]. <Disponible en ftp://jano.unicauca.edu.co/cursos/Electiva_ApliMovil/documentos/java-bluetooth-jidtel.pdf>

DIEZ-ANDINO, Guillermo, GARCIA, Rosa María. Java2 Micro Edition: un primer vistazo. Madrid: Universidad Carlos III de Madrid. [En línea]. [Citado el 27 de Marzo]. <Disponible en http://es.scribd.com/doc/6927801/Java2-Micro-Edition-un-primer-vistazo>

DOMÍNGUEZ, Hugo Martin, SÁEZ VACAS, Fernando. Domótica: Un Enfoque Sociotécnico. Madrid: Universidad Politécnica de Madrid. [En línea]. Junio 2006. [Citado el 27 de Marzo]. <Disponible en http://www.gsi.dit.upm.es/~fsaez/intl/libro_domotica.pdf>

DOMINGUEZ-DORADO, Manuel. NetBeans IDE 4.1. La alternativa a Eclipse, Todo Programación. Nº 13. Iberprensa, Madrid, España, 2005.

FAIRCHILD SEMICONDUCTOR, Solutions For Your Success. LM317 Datasheet. [En Línea]. [Citado el 14 de Marzo de 2012]. <Disponible en http://www.fairchildsemi.com/ds/LM/LM317.pdf>

FROUFE QUINTAS, Agustín, JORGE CÁRDENAS, Patricia. J2ME Java 2 Micro Edition: Manual de Usuario y Tutorial, Primera Edición. Alfaomega, México, 2004.

GALVEZ ROJAS, Sergio, ORTEGA DÍAZ, Lucas. Java A Tope: Java 2 Micro Edition J2ME. Edición Electrónica, Universidad de Málaga. España, 2003.

GARCIA SERRANO, Alberto. Programación de juegos para móviles con J2ME.[En línea]. [Citado el 27 de Marzo]. <Disponible en http://www.agserrano.com/publi.html>

GIMENO BRIEBA, Alberto. JSR-82: Bluetooth desde Java™. Universidad Castilla-La Mancha, Modelling Ambient Intelligence. [En línea]. 2004. [Citado el 27 de Marzo]. <Disponible en http://mami.uclm.es/j2me/J2ME/java-bluetooth.pdf>

GÓMEZ MÁRMOL, Félix. Desarrollo de aplicaciones Java para dispositivos móviles J2ME. Murcia, España: Universidad de Murcia, Área de Tecnologías de la Información y Comunicaciones Aplicadas. [En línea]. 2008 - 2011. [Citado el 27 de Marzo]. <Disponible en http://ants.dif.um.es/~felixgm/docencia/j2me/>

IEEE XPLORE, Digital Library. IEEE Standard for Information Technology – Telecomunications and Information Exchange. [En Línea]. 2002. [Citado el 13 de Marzo de 2012]. <Disponible en http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7932>

INSTITUTO TECNOLOGICO DE CANARIAS. Telefonía Móvil. [En Línea]. Marzo 2012- [Citado el 13 de Marzo de 2012]. <Disponible en http://www.itccanarias.org/web/difusion/como_funciona/movil/index.jsp?lang=es>

KC WIREFREE, We Make Bluetooth Work for You. KC Serial v2.4 User Guide. [En Línea]. [Citado el 15 de Marzo de 2012]. <Disponible en http://kcwirefree.com/docs/KCSerialUserGuide.pdf>

--------, --------. KC Wirefree Getting Started Guide. [En línea]. [Citado el 15 de Marzo de 2012]. <Disponible en http://www.ylib.com.cn/products/Bluetooth/doc/KC_GettingStartedGuide.pdf>

--------, --------. KC5100 Datasheet. [En Línea]. [Citado el 14 de Marzo de 2012]. <Disponible en http://kcwirefree.com/docs/KC5100_Datasheet.pdf >

INVESTIGACIÓN PARA el desarrollo de software en Domótica orientado al sector doméstico y empresarial de Colombia. [Anónimo]. [En Línea]. [Citado el 12 de Marzo de 2012]. <Disponible en http://es.scribd.com/doc/30715431/INVESTIGACION-PARA-EL-DESARROLLO-DE-SOFTWARE-EN-DOMOTICA-ORIENTADO-AL-SECTOR-DOMESTICO-Y-EMPRESARIAL-DE-COLOMBIA>

MAXIM, Innovation Delivered. MAX3225 Datasheet. [En Línea]. [Citado el 14 de Marzo de 2012]. <Disponible en http://datasheets.maxim-ic.com/en/ds/MAX3224-MAX3245.pdf>

NOKIA DEVELOPER. PACKAGE javax.bluetooth. [En línea]. 2012. [Citado el 27 de Marzo]. <Disponible en http://www.developer.nokia.com/>

ORACLE DOCUMENTATION. Java for Mobile Devices Documentation. [En línea]. 2006. [Citado el 27 de Marzo]. <Disponible en http://docs.oracle.com/javame/mobile.html>

OVERVIEW (JSR 82 Bluetooth API and OBEX API). Package javax.bluetooth. [En línea]. 2006. [Citado el 14 de Marzo de 2012]. <Disponible en http://docs.oracle.com/javame/config/cldc/opt-pkgs/api/bluetooth/jsr082/index.html>

SÁNCHEZ ARIAS, David. Domótica: diseño de una casa inteligente basado en la tecnología Jini. Puebla, México: Universidad de las Américas Puebla. [En línea]. Mayo 2004. [Citado el 27 de Marzo]. <Disponible en http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/sanchez_a_d/indice.html>

SANCHEZ OJEDA, Luisa Fernanda. Conectividad inalámbrica entre un robot y un teléfono celular utilizando sistemas embebidos. Riobamba, Ecuador: Escuela Superior Politécnica de Chimborazo. [En línea]. Junio 2010. [Citado el 27 de Marzo]. <Disponible en http://dspace.espoch.edu.ec/handle/123456789/382>

SONY ERICCSON, Developing Applications with the Java APIs for Bluetooth™ (JSR-82). [En línea]. 2004. [Citado el 27 de Marzo]. <Disponible en: http://mokik.googlecode.com/files/Bluetooth-jsr-82-training.pdf>

SUN DEVELOPER NETWORK. BLUETOOTH API JSR82, Using the Java APIs for Bluetooth, Part 2 - Putting the Core APIs to Work. [En línea]. 2005. [Citado el 27 de Marzo]. <Disponible en http://developers.sun.com/mobility/apis/articles/bluetoothcore/>

TOLEDO TESTA, Juan Ignacio. Descobriment de serveis en xarxes AD-HOC: Jade Bluetooth discovery middleware. Bellaterra, España: Universitat Autónoma de Barcelona. [En línea]. Septiembre de 2007. [Citado el 27 de Marzo]. <Disponible en http://www.recercat.net/bitstream/handle/2072/5414/PFCToledoTesta.pdf>

UNIVERSIDAD DE VIGO. Escuela Superior de Enxeñería Informática. FORMELLA, Arno. Hilos en Java - La interfaz Runnable. [En línea]. 2006. [Citado el 27 de Marzo]. <Disponible en http://trevinca.ei.uvigo.es/~formella/doc/cd05/node50.html> <Disponible en http://trevinca.ei.uvigo.es/~formella/doc/cd05/node44.html>

ANEXOS

Anexo A. Datasheet módulo Bluetooth KC-5100

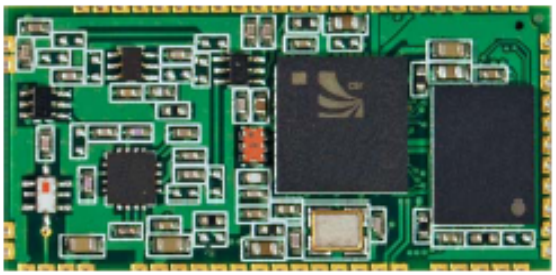
KC Wirefree

KC-5100

OEM Hi Power Bluetooth Data Module Datasheet

Firmware Features

- Wireless Data Communications Subsystem
- Embedded Bluetooth Serial Port Profile (SPP)
- Easy To Use AT Command Interface Using UART
- Remote Command And Control
- Multipoint / Piconet Capable
- Custom Firmware Available



33.2mm x 15.8 mm x 1.8 mm

Bluetooth csr

FC CE RoHS

Hardware Features

- CSR BlueCore™ 5 Chip Set
- Bluetooth v2.1+EDR Compliant
- Class 1 Radio, Range Typically Exceeds 150m
- High Speed Data Rate, Up To 3Mbps
- Programmable I/O pins - 20 Digital, 2 Analog
- 64 Mips DSP Co-Processor Onboard
- Two 16 Bit ADC Inputs up to 44 KHz Rate
- Two 16 Bit DAC Outputs up to 48 KHz Rate
- External Antenna Port
- USB, UART, SPI, I^2S, PCM, and SPDIF Interfaces

Applications

- Bluetooth Serial Cable Replacement
- Bluetooth Data Cable Replacement
- Bluetooth Advertising
- Bluetooth RFID Tag Readers
- Bluetooth Digital Picture Frames
- Bluetooth Hand-Held Bar Code Readers
- Bluetooth Medical Monitoring
- Bluetooth Credit Card Readers
- Many, many, more . . .

Description

The KC-5100 data module is pre-engineered, pre-qualified, and highly tuned surface mount PCB module that provides fully embedded, ready to use Bluetooth wireless technology. Multi-surface pads provide both bottom pads for high volume reflow soldering and edge pads for low volume hand soldering.

The KC-5100 offers reprogrammable, embedded firmware for serial cable replacement deploying the Bluetooth Serial Port Profile (SPP). OEM specific parameters and settings can be easily loaded into these modules.

Our kcSerial embedded firmware provides an easy to use AT style command interface over UART. kcSerial is capable of storing OEM default settings, and is upgradable over UART. kcSerial also provides remote control capability, where our AT commands can be issued remotely from any other Bluetooth device using SPP. Custom firmware is available.

(This module is also available for audio applications, refer to our KC-5190 -- Class 1, OEM Hi Power Bluetooth Audio Module.)

Electrical Characteristics (Preliminary)

(Conditions VDD= 3.3V and 25 °C)

Absolute Maximum Ratings	Min	Max	Unit
Storage temperature range	-40	+85	°C
Supply voltage VDD	-0.4	3.6	Volts
Supply voltage BAT_P	-0.4	4.4	Volts
Supply voltage VDD_CHG	-0.4	6.5	Volts
Input current VDD_CHG	--	6	mA
Supply voltage REG_EN	-0.4	4.9	Volts
Input voltage for PIO, USB, UART	-0.4	3.6	Volts

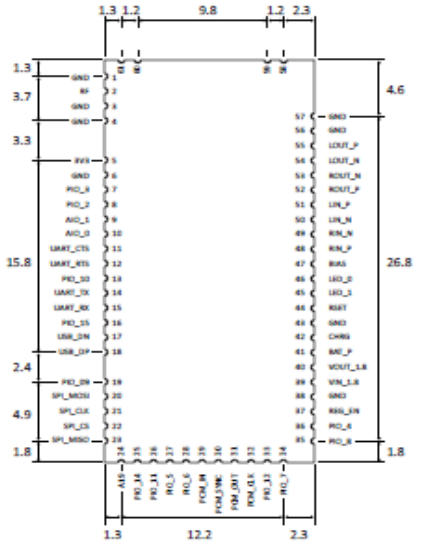
Recommend Operating Conditions	Min	Typical	Max	Unit
Operating temperature range	-40	20	+85	°C
Supply voltage VDD	1.7	3.3	3.6	Volts
Supply voltage BAT_P	2.5	--	4.4	Volts
Supply voltage VDD_CHG	4.5	--	6.5	Volts
Supply voltage VIN_1.8	1.70	1.80	1.95	Volts
Supply voltage REG_EN	2.5	--	4.4	Volts
Input voltage for PIO, USB, UART	1.7	3.3	3.6	Volts

Current Consumption	Avg	Unit
Data Transmission @ 380kbps	40	mA
No Connection	2	mA
Peak current	60	mA

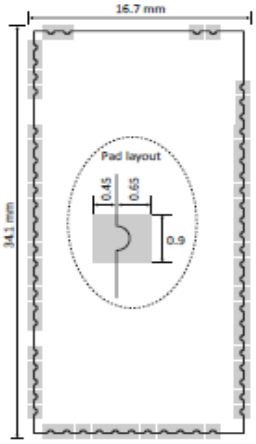
RF Characteristics	Min	Max	Unit
Carrier Frequency	2400	2483.5	MHz
Transmission Line	50	50	Ω
Transmission Power	0	20	dBm
Receive Sensitivity	-20	-90	dBm

Physical Dimensions

KC-5100
Top View
33.2 mm x 15.8 mm x 1.8 mm



KC-5100
Landing Pattern



Anexo B. Datasheet microcontrolador PIC18F4550

MICROCHIP **PIC18F2455/2550/4455/4550**

28/40/44-Pin, High-Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 bidirectional)
- 1-Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver with On-Chip Voltage Regulator
- Interface for Off-Chip USB Transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 µA typical
- Sleep mode currents down to 0.1 µA typical
- Timer1 Oscillator: 1.1 µA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 µA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes, including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal Oscillator Block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Dual Oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-Current Sink/Source: 25 mA/25 mA
- Three External Interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 5.2 ns (TCY/16)
  - Compare is 16-bit, max. resolution 83.3 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI (all 4 modes) and I²C™ Master and Slave modes
- 10-bit, up to 13-channel Analog-to-Digital Converter module (A/D) with Programmable Acquisition Time
- Dual Analog Comparators with Input Multiplexing

Special Microcontroller Features:

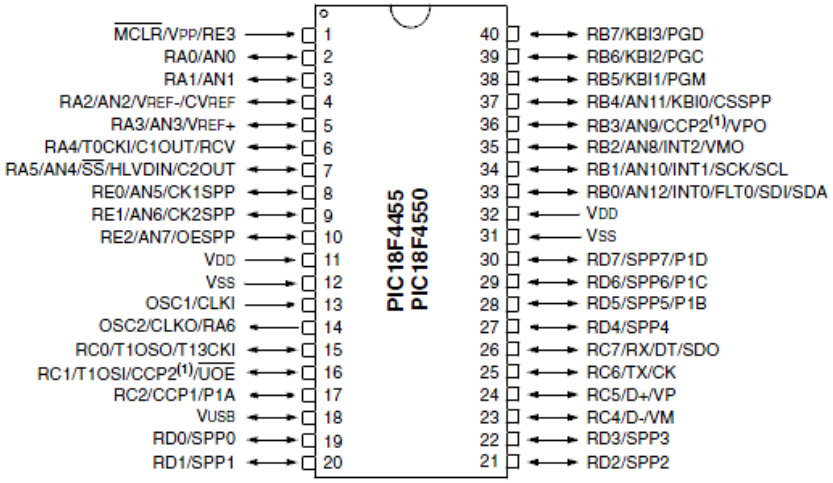
- C Compiler Optimized Architecture with optional Extended Instruction Set
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)
- Wide Operating Voltage Range (2.0V to 5.5V)

Device	Program Memory		Data Memory		I/O	10-Bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-Bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI	Master I²C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

TABLE 1-1: DEVICE FEATURES

Features	PIC18F2455	PIC18F2550	PIC18F4455	PIC18F4550
Operating Frequency	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz
Program Memory (Bytes)	24576	32768	24576	32768
Program Memory (Instructions)	12288	16384	12288	16384
Data Memory (Bytes)	2048	2048	2048	2048
Data EEPROM Memory (Bytes)	256	256	256	256
Interrupt Sources	19	19	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	1	1
Enhanced Capture/ Compare/PWM Modules	0	0	1	1
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Universal Serial Bus (USB) Module	1	1	1	1
Streaming Parallel Port (SPP)	No	No	Yes	Yes
10-Bit Analog-to-Digital Module	10 Input Channels	10 Input Channels	13 Input Channels	13 Input Channels
Comparators	2	2	2	2
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT
Programmable Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	28-pin PDIP 28-pin SOIC	28-pin PDIP 28-pin SOIC	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP

40-Pin PDIP



Anexo C. Datasheet adaptador de niveles MAX3225

19-1289; Rev 1; 3/98

MAXIM

1µA Supply Current, 1Mbps, 3.0V to 5.5V, RS-232 Transceivers with AutoShutdown Plus

General Description

The MAX3224–MAX3227/MAX3244/MAX3245 are 3V-powered EIA/TIA-232 and V.28/V.24 communications interfaces with automatic shutdown/wakeup features and high data-rate capabilities.

All devices achieve a 1µA supply current using Maxim's revolutionary AutoShutdown Plus™ feature. These devices automatically enter a low-power shutdown mode when the RS-232 cable is disconnected or the transmitters of the connected peripherals are inactive, and the UART driving the transmitter inputs is inactive for more than 30 seconds. They turn on again when they sense a valid transition at any transmitter or receiver input. AutoShutdown Plus saves power without changes to the existing BIOS or operating system.

The MAX3225/MAX3227/MAX3245 also feature MegaBaud™ operation, guaranteeing 1Mbps for high-speed applications such as communicating with ISDN modems. The MAX3224/MAX3226/MAX3244 guarantee 250kbps operation. The transceivers have a proprietary low-dropout transmitter output stage enabling true RS-232 performance from a +3.0V to +5.5V supply with a dual charge pump. The charge pump requires only four small 0.1µF capacitors for operation from a 3.3V supply. The MAX3224–MAX3227 feature a logic-level output (READY) that asserts when the charge pump is regulating and the device is ready to begin transmitting.

All devices are available in a space-saving SSOP package.

Applications

Notebook, Subnotebook, and Palmtop Computers

Cellular Phones

Battery-Powered Equipment

Hand-Held Equipment

Peripherals

Printers

Features

- **1µA Supply Current**
- **AutoShutdown Plus—EDN Innovation of the Year**
- **Guaranteed Data Rate:**
  250kbps (MAX3224/3226/3244)
  1Mbps (MAX3225/3227/3245)
- **Guaranteed Slew Rate:**
  6V/µs (MAX3224/3226/3244)
  24V/µs (MAX3225/3227/3245)
- **Meets EIA/TIA-232 Specifications Down to 3.0V**
- **Guaranteed Mouse Driveability (MAX3244/3245)**
- **Ready-to-Transmit Logic-Level Output**

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX3224CPP	0°C to +70°C	20 Plastic DIP
MAX3224CAP	0°C to +70°C	20 SSOP
MAX3224EPP	-40°C to +85°C	20 Plastic DIP
MAX3224EAP	-40°C to +85°C	20 SSOP
MAX3225CPP	0°C to +70°C	20 Plastic DIP
MAX3225CAP	0°C to +70°C	20 SSOP
MAX3225EPP	-40°C to +85°C	20 Plastic DIP
MAX3225EAP	-40°C to +85°C	20 SSOP

Ordering Information continued at end of data sheet.

Selector Guide

PART	NO. OF DRIVERS/ RECEIVERS	GUARANTEED DATA RATE (bps)	READY OUTPUT	AUTO-SHUTDOWN PLUS
MAX3224	2/2	250k	✓	✓
MAX3225	2/2	1M	✓	✓
MAX3226	1/1	250k	✓	✓
MAX3227	1/1	1M	✓	✓
MAX3244	3/5	250k	—	✓
MAX3245	3/5	1M	—	✓

ELECTRICAL CHARACTERISTICS

(V_{CC} = +3V to +5.5V, C1–C4 = 0.1µF, tested at 3.3V ±10%; C1 = 0.047µF, C2–C4 = 0.33µF, tested at 5.0V ±10%; T_A = T_{MIN} to T_{MAX}, unless otherwise noted. Typical values are at T_A = +25°C.)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
DC CHARACTERISTICS (V_{CC} = 3.3V or 5.0V, T_A = +25°C)							
Supply Current, AutoShutdown Plus		FORCEON = GND, $\overline{FORCEOFF}$ = V_{CC}, all R_IN idle, all T_IN idle			1	10	µA
Supply Current, Shutdown		$\overline{FORCEOFF}$ = GND			1	10	µA
Supply Current, AutoShutdown Plus Disabled		FORCEON = $\overline{FORCEOFF}$ = V_{CC}, no load			0.3	1	mA
LOGIC INPUTS AND RECEIVER OUTPUTS							
Input Logic Threshold Low		T_IN, FORCEON, $\overline{FORCEOFF}$				0.8	V
Input Logic Threshold High		T_IN, FORCEON, $\overline{FORCEOFF}$	V_{CC} = 3.3V	2			V
			V_{CC} = 5.0V	2.4			
Transmitter Input Hysteresis					0.5		V
Input Leakage Current		T_IN, FORCEON, $\overline{FORCEOFF}$			±0.01	±1	µA
Output Leakage Current		R_OUT (MAX3244/MAX3245), receivers disabled			±0.05	±10	µA
Output Voltage Low		I_{OUT} = 1.6mA				0.4	V
Output Voltage High		I_{OUT} = -1.0mA		V_{CC} - 0.6	V_{CC} - 0.1		V
RECEIVER INPUTS							
Input Voltage Range				-25		+25	V
Input Threshold Low		T_A = +25°C	V_{CC} = 3.3V	0.6	1.2		V
			V_{CC} = 5.0V	0.8	1.5		
Input Threshold High		T_A = +25°C	V_{CC} = 3.3V		1.5	2.4	V
			V_{CC} = 5.0V		1.8	2.4	
Input Hysteresis					0.5		V
Input Resistance		T_A = +25°C		3	5	7	kΩ
TRANSMITTER OUTPUTS							
Output Voltage Swing		All transmitter outputs loaded with 3kΩ to ground		±5	±5.4		V
Output Resistance		V_{CC} = V+ = V- = 0, transmitter outputs = ±2V		300	10M		Ω
Output Short-Circuit Current						±60	mA
Output Leakage Current		V_{CC} = 0 or 3V to 5.5V, V_{OUT} = ±12V, Transmitters disabled				±25	µA

Cellular Phones

Battery-Powered Equipment

Handheld Equipment

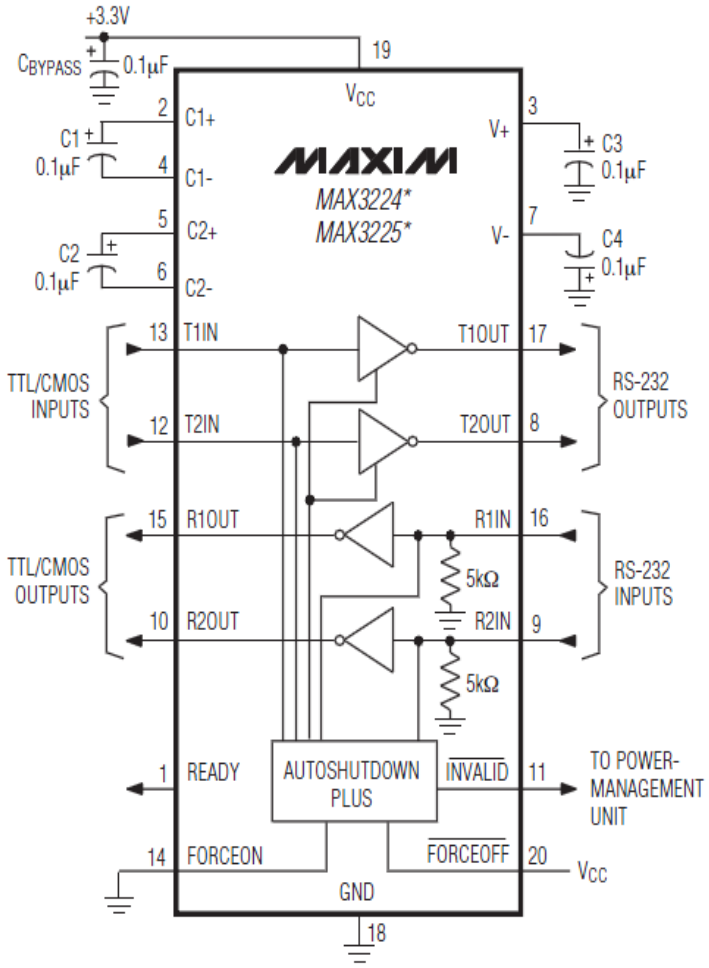
Peripherals

Printers

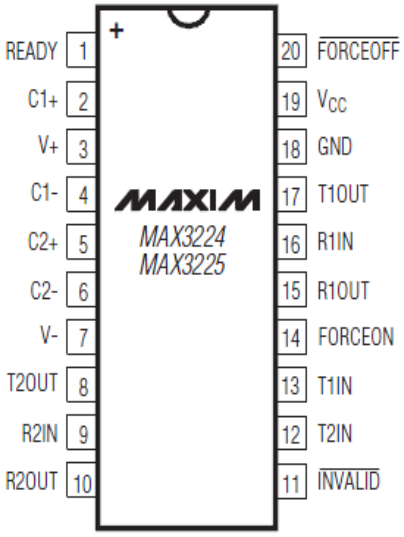
Selector Guide

PART	NO. OF DRIVERS/ RECEIVERS	GUARANTEED DATA RATE (bps)	READY OUTPUT	Auto-Shutdown Plus
MAX3224	2/2	250k	✔	✔
MAX3225	2/2	1M	✔	✔
MAX3226	1/1	250k	✔	✔
MAX3227	1/1	1M	✔	✔
MAX3244	3/5	250k	—	✔
MAX3245	3/5	1M	—	✔

Typical Operating Circuits



Pin Configurations



DIP/SSOP/TSSOP

Anexo D. Descripción de algunos JSR

JSR	DESCRIPCIÓN
30	**Java ME™ Connected, Limited Device Configuration.** Esta especificación definirá un estándar para la configuración de Java2 Platform Micro Edition (Java ME™) para dispositivos pequeños con recursos limitados con conexión (CLDC).
36	**Connected Device Configuration.** La configuración de dispositivos con conexión (CDC) provee las bases de Java 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 ME™ Platform.** Esta especificación definirá un perfil que extenderá y ampliará el Java ME 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 ME™ 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.
68	**Java ME™ Platform Specification.** Esta especificación definirá la siguiente mejor revisión del Java™ 2 Platform, Micro Edition.
75	**PDA Optional Packages for the Java ME™ 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	**Java™ APIs for Bluetooth.** Bluetooth es un estándar importante que está emergiendo para la integración wireless 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 ME™ Mobile Information Device Profile" (JSR-000037).
134	**Java 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 ME™, 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 ME™ Connected, Limited Device Configuration (CLDC).

JSR	DESCRIPCIÓN
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.
246	**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.
927	**Java TV API 1.1** El mantenimiento de la especificación Java TV.

Anexo E. Clases incluidas en javax.bluetooth

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 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.
BluetoothConnectionException	BluetoothConnectionException es devuelto cuando la conexión Bluetooth (L2CAP, RFCOMM, OBEX) no pudo ser establecida exitosamente.
Authenticator	Esta interfaz provee una forma de responder a un desafío y a una cabecera de respuesta de autenticación.

Anexo F. Interface de comunicación Bluetooth

```
public void bluetoothDiscovery() {
        LocalDevice localDevice = null;
        try {
            localDevice = LocalDevice.getLocalDevice();
            localDevice.setDiscoverable(DiscoveryAgent.GIAC);
            discoveryAgent = localDevice.getDiscoveryAgent();
        } catch(Exception e) {
            e.printStackTrace();
        }

        dispositivosEncontrados = new Vector();
        try {
            discoveryAgent.startInquiry(DiscoveryAgent.GIAC, this);
        } catch(BluetoothStateException e) {
            e.printStackTrace();
        }
    }

public void deviceDiscovered(RemoteDevice remoteDevice,DeviceClass
deviceClass) {

        String address = remoteDevice.getBluetoothAddress();
        try {
            String friendlyName = remoteDevice.getFriendlyName(true);
            if ("DOMCELL".equals(friendlyName)){
                friendlyName = "DOMCELL";
            }
        } catch(IOException e) {
        }

        try {
                int transID =
discoveryAgent.searchServices(ATRIBUTOS,SERVICIOS, remoteDevice, this);
        } catch(BluetoothStateException e) {
                e.printStackTrace();
        }
    }

public void inquiryCompleted(int discType) {
    }

public void servicesDiscovered(int transID,ServiceRecord[] servRecord) {
        ServiceRecord service = null;
        for(int i=0; i<servRecord.length; i++){
            service = servRecord[i];
            url =
service.getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYPT, false);
            enviar(url);
        }
    }
```

```
public void enviar(String url){
        t = new Thread(this);
        t.start();
    }

public void run(){
        try {
            StreamConnection connection = (StreamConnection)
Connector.open(url);
            DataInputStream in = connection.openDataInputStream();
            DataOutputStream out = connection.openDataOutputStream();

            //<editor-fold defaultstate="collapsed" desc=" Generated
Getter: SELECCION COMANDOS ">

            if (puerto == 0)
                x = "#0\n";
            if (puerto == 1)
                x = "#1\n";
            if (puerto == 2)
                x = "#2\n";
            if (puerto == 3)
                x = "#3\n";
            if (puerto == 4)
                x = "#4\n";
            if (puerto == 5)
                x = "#5\n";
            if (puerto == 6)
                x = "#6\n";
            if (puerto == 7)
                x = "#7\n";
            if (puerto == 8)
                x = "#8\n";
            if (puerto == 9)
                x = "#9\n";

            out.writeUTF(x); out.flush(); t.sleep(1000);
            in.close();
            out.close();
            connection.close();
            } catch(IOException e) {
                e.printStackTrace();
            } catch(IllegalMonitorStateException e){
                e.printStackTrace();
            } catch(InterruptedException e){
                e.printStackTrace();
            }
    }

public void serviceSearchCompleted(int transID, int respCode) {
    }
```

Anexo G. Programación del microcontrolador

```
#include <18F4550.h>
#device adc=8
#include <stdio.h>
#include <string.h>
#include <stdlib.h>


#FUSES NOWDT                    //No Watch Dog Timer
#FUSES HS                       //Crystal osc > 4mhz for PCM/PCH , 3mhz
to 10 mhz for PCD
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or
B5(PIC18) used for I/O
#FUSES NOCPD                    //No EE protection
#FUSES NOWRT                    //Program memory not write protected
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOPROTECT                //Code not protected from reading

#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)

char stri[50];
char datos;
char comando;
char *pch;

#INT_RDA
void RDA_isr(void)
{
   output_high(PIN_A0);
   delay_ms(500);
   output_low(PIN_A0);
   gets(datos);
   getc();
   delay_us(10);
}

void main()
{
   output_high(PIN_A1);
   enable_interrupts(INT_RDA);
   enable_interrupts(GLOBAL);

   pch=strchr(datos,'#');

   while(pch != NULL)
   {
      strcpy(comando,*pch[1]);
      pch=strchr(pch+1,'#');
```

```
      switch (comando)
      {
         case '0':
         output_high(PIN_B0);
         break;

         case '1':
         output_low(PIN_B0);
         break;

         case '2':
         output_high(PIN_B1);
         break;

         case '3':
         output_low(PIN_B1);
         break;

         case '4':
         output_high(PIN_B2);
         break;

         case '5':
         output_low(PIN_B2);
         break;
      }
   }
}
```

Anexo H. Lista de comandos

COMANDO	OPCIÓN SELECCIONADA	PUERTO
#1	ON	1
#2	OFF	
#3	ON	2
#4	OFF	
#5	ON	3
#6	OFF	
#7	ON	4
#8	OFF	
#9	ON	5
#0	OFF	
#a	ON	6
#b	OFF	
#c	ON	7
#d	OFF	
#e	ON	8
#f	OFF	
#g	ON	9
#h	OFF	
#i	ON	10
#j	OFF	
#k	ON	11
#l	OFF	
#m	ON	12
#n	OFF	
#o	ON	13
#p	OFF	
#q	ON	14
#r	OFF	
#s	ON	15
#t	OFF	

Anexo I. Manual de uso

Inicio

La aplicación aparece con el nombre de DOMCELL en el lugar de instalación específico para este tipo de aplicaciones dentro de cada teléfono celular. Al instalar y ejecutar la aplicación, ésta solicita confirmar la activación del Bluetooth. Una vez se autoriza, se muestra la pantalla de inicio de la aplicación y automáticamente el teléfono empieza una búsqueda a su alrededor, conectándose al Módulo Bluetooth. Éste procedimiento de búsqueda y conexión es transparente para el usuario, tardando aproximadamente 8 segundos hasta que se completa la conexión.

Control de acceso

Al escoger la opción “Continuar” en la pantalla de inicio de la aplicación, aparece un formulario donde hay un cuadro de texto para ingresar una contraseña, como se muestra en la Figura 1. Al ingresar la contraseña, se debe seleccionar la opción “OK”. Cuando la contraseña no se ingresa correctamente, se muestra una alerta indicando que no se puede continuar, y luego se debe seleccionar la opción “VOLVER” para regresar al formulario anterior y realizar un nuevo intento. No está definido un número máximo de intentos. La contraseña es una cadena de texto simple, no cifrado. Los valores ingresados son comparados por la aplicación con un texto único incluido en el código fuente.

Figura 1. Control de acceso

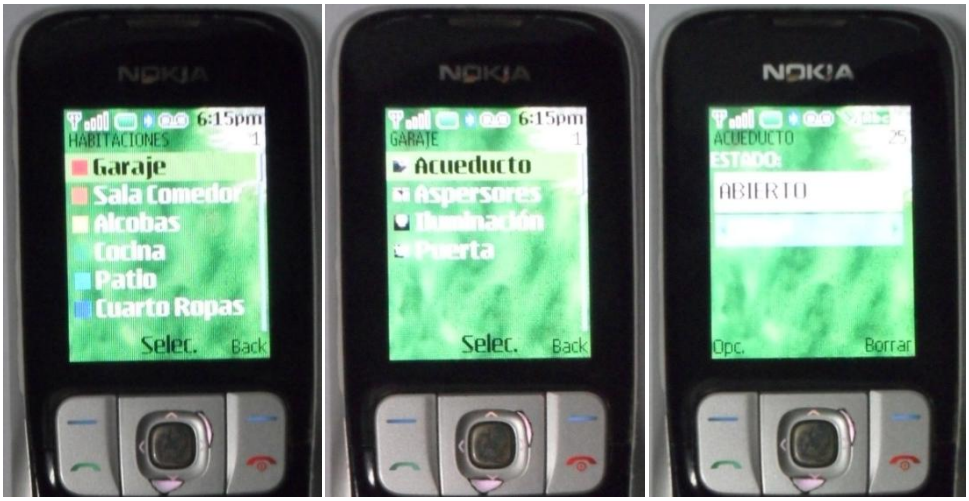
Cuando la contraseña se ingresa correctamente y se escoge la opción “OK”, se obtiene acceso a la aplicación y ésta continúa hacia el menú de control (ver Figura 2).

Figura 2. Menú de control

Control de dispositivos

Para permitir el control de cada uno de los dispositivos del hogar, se debe ingresar a cada elemento en la sección “HABITACIONES” en el menú de control (véase Figura 3). Una vez ubicado el dispositivo, se muestran las posibles opciones con las cuales se cuenta: “ENCENDER” y “APAGAR”.

Figura 3. Control de dispositivos



Ayuda

Este formulario -véase Figura 4- se encuentra ubicado en el menú de control. Allí se encuentra la información necesaria con la cual el usuario podrá obtener las instrucciones para realizar el control de los dispositivos.

Figura 4. Ayuda

Acerca de...

Este formulario -véase Figura 5- se encuentra ubicado en el menú de control. Allí se incluye información del proyecto y de los autores de la aplicación.

Figura 5. Acerca de