UNIVERSIDAD TECNOLÓGICA ISRAEL
FACULTAD DE ELECTRÓNICA

ESTUDIO, DISEÑO, E IMPLEMENTACIÓN DE UN ENTRENADOR LÓGICO Y GUÍAS DE PRÁCTICA BASADO EN MICROCONTROLADORES, RADIOFRECUENCIA Y BLUETOOTH PARA LA FACULTAD DE INGENIERÍA ELECTRÓNICA DE LA UNIVERSIDAD TECNOLÓGICA ISRAEL

TERÁN GUALOTO EDISON JAVIER

TUTOR
ING. MAURICIO ALMINATI

Quito D.M., AGOSTO 2012

DECLARACIÓN

Yo, Edison Javier Terán Gualoto, declaro bajo juramento que el trabajo aquí descrito, es de mí autoría; que no ha sido previamente presentado para ningún grado o calificación profesional y que he consultado e investigado en base a las referencias bibliográficas que se incluyen en este documento.

_____

Edison Terán

CERTIFICACIÓN 1

Una vez que se ha culminado la elaboración del trabajo de titulación de pregrado cuyo tema es: “ESTUDIO, DISEÑO, E IMPLEMENTACIÓN DE UN ENTRENADOR LÓGICO Y GUÍAS DE PRÁCTICA BASADO EN MICROCONTROLADORES, RADIOFRECUENCIA Y BLUETHOOTH PARA LA FACULTAD DE INGENIERÍA ELECTRÓNICA DE LA UNIVERSIDAD TECNOLÓGICA ISRAEL ”, certifico que el mismo se encuentra habilitado para su defensa pública.

_____

Ing. Wilmer Albarracín MBA

COORDINADOR DE LA FACULTAD DE

ELECTRÓNICA Y TELECOMUNICACIONES

UNIVERSIDAD ISRAEL

CERTIFICACIÓN 2

A través de la presente, Certifico que el señor Edison Javier Terán Gualoto ha realizado y concluido su trabajo de titulación de pregrado cuyo tema es: “ESTUDIO, DISEÑO, E IMPLEMENTACIÓN DE UN ENTRENADOR LÓGICO Y GUÍAS DE PRÁCTICA BASADO EN MICROCONTROLADORES, RADIOFRECUENCIA Y BLUETHOOTH PARA LA FACULTAD DE INGENIERÍA ELECTRÓNICA DE LA UNIVERSIDAD TECNOLÓGICA ISRAEL “, para obtener el título de Ingeniero en Electrónica y Telecomunicaciones, bajo mí tutoría.

_____

Ing. Mauricio Alminati

DIRECTOR DE TRABAJO DE TITULACIÓN DE PREGRADO

AGRADECIMIENTO

En primer lugar a mi Dios por darme sabiduría, fuerza y conocimiento con los cuales pude finalizar con mucho éxito el presente proyecto, a mis padres, Edito y Emma, hermanos , Katty, Mau, Danny y Emmy por su impulso moral y económico a lo largo de mi trayectoria estudiantil finalmente a: Ing. Mauricio Alminati, Ing. Fernando Vallejo e Ing. Jose Robles por el apoyo recibido y desinteresado.

Javier

DEDICATORIA

A mi Dios, a mis Padres para que sepan que han forjado un hombre en todo el sentido de la palabra, a mis Hermanos quienes con sus compañía y buen humor me inspiraban a seguir y en especial a mi Esposa Miriam, a quien Amo con todo mi corazón, por todo su apoyo recibido, por su compañía, por su tiempo y su amor .

Javier

PRÓLOGO

El presente proyecto de grado titulado “ESTUDIO, DISEÑO, E IMPLEMENTACIÓN DE UN ENTRENADOR LÓGICO Y GUÍAS DE PRÁCTICA BASADO EN MICROCONTROLADORES, RADIOFRECUENCIA Y BLUETOOTH PARA LA FACULTAD DE INGENIERÍA ELECTRÓNICA DE LA UNIVERSIDAD TECNOLÓGICA ISRAEL.” se ha desarrollado, implementado y probado en la Facultad de Electrónica de la Universidad Tecnológica Israel, permitiendo de esta manera contar con un equipo que facilite al estudiante realizar ejercicios prácticos de comunicaciones inalámbricas de acuerdo a la teoría recibida en los últimos años de su formación profesional.

El Entrenador Lógico está diseñado en base a un proceso de análisis y síntesis mediante la recopilación de antecedentes bibliográficos, para implementar un equipo que cuente con los componentes óptimos para la elaboración de proyectos basados en RF, BT, JAVA y ANDROID, tomando en cuenta parámetros eléctricos, electrónicos así como también parámetros de funcionalidad, costo y tamaño, siguiendo un formato establecido en la Guía de Práctica desarrollada dentro del presente proyecto de grado.

El análisis financiero del mismo está realizado en base un análisis costo beneficio para la Uisrael.

ABSTRACT

This graduation project entitled "STUDY, DESIGN, AND IMPLEMENTATION OF A LOGIC TRAINER AND GUIDES PRACTICE BASED MICROCONTROLLERS, RADIOFREQUENCY AND BLUETOOTH FOR THE FACULTY OF ELECTRONIC ENGINERING ISRAEL TECH UNIVERSITY. " has developed, implemented and tested in the Faculty of Electronics of the Technological University Israel, thus allowing to have a team to provide the student practical exercises wireless communications according to the received theory in the last years of his vocational training.

The logic trainer is designed based on a process of analysis and synthesis by compilation of background bibliographic, to implement a team with the best components for the elaboration of projects based on RF, BT, JAVA and ANDROID, considering electrical parameters, electronic and also parameters of functionality, cost and size, following a format established practice guideline developed within this project grade.

The financial analysis is performed on the same basis a cost benefit analysis for Uisrael.

TABLA DE CONTENIDOS

CAPITULO I

1.1. Introducción

En este capítulo se trata sobre el proceso de elaboración de un plan de proyecto de grado, el cual consta de los antecedentes del lugar donde se implementará el proyecto propuesto, se plantea un problema principal, y problemas secundarios, lo que conlleva a un objetivo principal con sus objetivos específicos.

Además trata sobre los métodos de investigación que se desarrollaran en las diferentes etapas que son: estudio, diseño e implementación del proyecto propuesto

1.2. Antecedentes

La Universidad Tecnológica Israel (UISRAEL) es una institución de nivel superior encargada de formar profesionales, nace en 1999 de la experiencia académica y profesional de dos institutos: el Instituto Tecnológico Israel y el Instituto Tecnológico Italia, alianza estratégica de fortalecimiento para brindar una educación superior de excelencia académica.

En la década de los noventa, estos institutos obtuvieron distintos reconocimientos a nivel nacional e internacional, legado del cual se nutre la UISRAEL y que permitió consolidar una oferta académica que tiene en la actualidad carreras en modalidades presencial, semi presencial y a distancia.

Las carreras de pregrado que ofrece la universidad son: Ingeniería en Administración de Empresas, Ingeniería en Administración Hotelera y Turística, Ingeniería Comercial, Ingeniería en Diseño Gráfico, Ingeniería en Electrónica Digital y Telecomunicaciones, Ingeniería en Producción de Televisión y Multimedia, Ingeniería en Sistemas Informáticos, Licenciatura en Contabilidad Pública y Auditoría y Licenciatura en Gastronomía.

La Facultad de Ingeniería Electrónica a través de los proyectos de investigación básica encamina, estimula y entrega herramientas teóricas y prácticas para que, desde una visión integral, sus estudiantes puedan impulsar sus propias iniciativas empresariales.

1.3. Problema Investigado

Actualmente en la Facultad de Ingeniería Electrónica hay un promedio de trescientos estudiantes que concurren usualmente a los laboratorios de: Sistemas, Instrumentación Electrónica, Redes de Datos, Microcontroladores que sirven para su formación profesional sin embargo en la Facultad de Electrónica no existe los equipos necesarios dedicados a la elaboración de sistemas o prototipos electrónicos controlados mediante comunicación inalámbrica.

La Facultad de Ingeniería Electrónica imparte las materias de Sistemas Digitales, Redes de Datos, Microprocesadores, Sistemas de Comunicación, Control Automático, Sistemas TX RX, dichas materias se imparten en forma teórica y no se realizan prácticas que evidencien el concepto científico-teórico.

Además, en el área de programación, no se diseña proyectos de interfaz gráfica, por lo que no se realiza transmisión de datos a través de de computadoras ni con teléfonos celulares.

La existencia de avances tecnológicos tales como el manejo de la tecnología Bluethooth, para comunicación mediante una interfaz gráfica y programación en J2ME, y Microcontroladores está siendo estudiada en los diferentes centros de educación superior, mientras en la Facultad de Electrónica de la Universidad Israel aún no se ha iniciado el estudio sobre estos temas.

A la fecha se realizan diferentes proyectos de comunicaciones en las que interviene; tecnología Bluethooth, Microcontroladores, Programación J2ME, Android, dichas tecnologías y materias son puestas en práctica en algunos centros de educación superior, mismos en los que se desarrollan proyectos que conllevan la utilización de la tecnología mencionada por ejemplo, Automatización de un departamento a través de una interfaz gráfica en un celular por Bluethooth (proyecto realizado en la Escuela Politécnica del Ejercito en mayo del 2011, previo a la obtención del título de Ingeniero Electrónico), consecuentemente para la elaboración de los mencionados proyectos estas universidades poseen laboratorios con los equipos necesarios para el diseño y construcción de esta clase de proyectos. Además que estas prácticas son realizadas como parte del pensum académico desde niveles intermedios y en

otras universidades también existen las opciones de ahondar en estos temas ya sea en seminarios, materias optativas y talleres de investigación.

1.4. Problema Principal

En los laboratorios de la Facultad de Electrónica de la Universidad Tecnológica Israel no se dispone de los equipos necesarios dedicados a la elaboración de sistemas o prototipos controlados mediante Comunicación Inalámbrica a través de J2ME y ANDROID.

1.5. Problemas Secundarios

- No existe suficientes antecedentes bibliográficos acerca del uso de de J2ME y ANDROID para el desarrollo de prácticas de Comunicación Inalámbrica a través de celulares.

- En la Facultad de Electrónica no se dispone de un Entrenador Lógico y equipos secundarios que lo complementen.

- No se dispone de guías de prácticas de Comunicación Inalámbrica y Programación.

1.6. Formulación del Problema

¿La implementación de un Entrenador Lógico y Guías de Práctica basado en Microcontroladores, Radiofrecuencia y Bluethooth permitirá que los estudiantes de la Universidad Tecnológica Israel adquieran el conocimiento teórico y práctico de acuerdo al nivel de formación adquirido durante su carrera en comunicación a través de RF y BT basado en J2ME y ANDROID?

1.7. Justificación

El presente proyecto está orientado al desarrollo de habilidades y capacidades de los estudiantes de la Facultad de Electrónica para la creación y correcto manejo de aplicaciones basados en J2ME y ANDROID además de comunicación a través de Microcontroladores y Módulos de Transmisión Inalámbrica como Radiofrecuencia y

Bluethooth.

Este entrenador será de mucho beneficio puesto que los proyectos que podrán ser ejecutados utilizarán tecnología de última generación lo que desarrollará capacidades teóricas y prácticas en los estudiantes que les permitan estar capacitados a la par del resto de profesionales.

El Entrenador Lógico permitirá tener acceso a conocimientos acerca de comunicación a través de Módulos Bluethooth y Módulos de Radiofrecuencia se podrá realizar interfaces gráficas mediante el sistema operativo basado en Linux diseñado exclusivamente para dispositivos celulares que es Android, este tipo de interfaz es para teléfonos celulares de última generación, otro método de creación de interfaz gráfica es mediante java que es J2ME (Java Segunda Edición Micro).

Entre las guías que complementarán el proyecto se trabajará aspectos de interés y utilidad para los estudiantes por ejemplo la Comunicación Inalámbrica mediante Bluethooth que podría ser utilizada en proyectos novedosos como el monitoreo de la seguridad de una casa a través de un dispositivo celular.

Dicho proyecto se lo implementará en la Facultad de Electrónica en el laboratorio de Microcontroladores y Diseño Electrónico, además se utilizará el Entrenador en el grupo denominado “club de robótica”, el cual es un grupo consolidado e independiente, y en los seminarios que se dicten para los niveles superiores de la Facultad.

1.8. Objetivos

1.8.1. Objetivo Principal

Estudiar, Diseñar e Implementar un Entrenador Lógico y guías de práctica basado en Microcontroladores, Radiofrecuencia, Bluethooth, J2ME y ANDROID para la Facultad de Ingeniería Electrónica de la Universidad Tecnológica Israel.

1.8.2. Objetivos Específicos

- Investigar y analizar el lenguaje de programación JAVA 2 MICRO EDITION y el sistema operativo ANDROID como herramientas para el proyecto a

implementarse.

- Diseñar e Implementar un Entrenador Lógico y equipo complementario.

- Elaborar una guía de prácticas de Comunicaciones Inalámbricas y Programación.

1.9. Metodología Científica

Esta investigación se realizó en varias etapas que son; Estudio, Diseño, Implementación y Validación. Para el Estudio, se utilizó el Método de Análisis y Síntesis por medio del cual se recopiló argumentos, textos, citas para tener un concepto general del proyecto.

El Diseño está basado en el Método Inductivo, con este método se realizó un análisis ordenado, coherente y lógico del problema de investigación, tomando como referencia premisas verdaderas de la problemática planteada mediante el cual se pudo tener acceso a un diseño único y funcional.

El Método Experimental fue utilizado para la Implementación del proyecto, ya que dicho método permitió recopilar información, analizar, sintetizar y elaborar el diseño experimental para realizar los ensayos necesarios lo que conlleva a comparar los resultados y así poder realizar las conclusiones y recomendaciones necesarias.

Finalmente se utilizó el Método Sistémico para realizar la validación del proyecto el cual permitió modelar el objeto mediante la determinación de los componentes que poseen el Entrenador Lógico, así como la relación entre ellos para determinar la estructura y su funcionamiento.

CAPÍTULO II

MARCO TEÓRICO

2.1. Introducción

En este capítulo se trata los antecedentes bibliográficos necesarios para la elaboración del proyecto, los cuales constan de dos partes que son hardware y software. Con respecto al hardware se describe módulos de transmisión y recepción inalámbrica como:

- ✓ Modulo de Radiofrecuencia.
- ✓ Módulo Bluetooth.

Además se describe los Microcontroladores, su arquitectura, los microcontroladores más utilizados y Atmel.

Con respecto al software se detalla el lenguaje de programación Java el sistema operativo Android, además de programas quemadores de PICs, software de desarrollado para J2ME.

Entre otros temas, se especifica acerca de bluethooth, radiofrecuencia y Netbeans.

2.2. JAVA

2.2.1. Reseña Histórica

En 1950 Patrick Naugton, ingeniero de Sun Microsystems reunió un grupo de ingenieros, entre ellos James Gosling y Bill Joy, para trabajar en el proyecto conocido como The Green Project “El Proyecto Verde”. Tenían como objetivo principal desarrollar una nueva tecnología para programar la siguiente generación de dispositivos inteligentes, tecnología que permita la creación de un lenguaje de programación fácil de aprender y de usar.

El equipo “Green Team” compuesto por trece personas y liderada por James Gosling, trabajó durante 18 meses en una pequeña oficina en Sand Hill Road en Menlo Park, California. En un principio se considero C++ como el lenguaje a utilizar, Gosling intentó primero extender y modificar C++ resultando el lenguaje C++ ++- (+ +- por que se añadían y eliminaban características a C++) pero tanto Gosling como Bill Joy lo encontraron inadecuado y lo abandonaron completamente para crear un

nuevo lenguaje desde cero al que Gosling lo llamo Oak (por un roble que había fuera de su oficina). El resultado fue un lenguje que tenia similitud con C, C++ y Objetive C y que no estaba ligado a un tipo de CPU concreta.

En 1992 el “Green Team” ya había diseñado un prototipo llamado Star 7, dispositivo parecido a un PDA, después de indicar los prototipos de bajo nivel del sistema a Bill Joy y Scott McNealy, continúan con el desarrollo, incluyendo el sistema operativo, Green OS; el lenguaje Oak, las librerías, una aplicación básica y el hardware, hasta que en septiembre de 1992 se termina el desarrollo y con ello el proyecto verde.

En 1994 se cambió el nombre de Oak por Java, por cuestiones de propiedad intelectual, al existir ya un lenguaje con el nombre de Oak.

El cese del Proyecto Verde coincidió con el nacimiento del fenómeno mundial WEB. Al examinar las dinámicas de Internet, lo realizado por el ex equipo verde se adecuaba a este nuevo ambiente.

Patrick Naughton procedió a la construcción del lenguaje de programación Java que se accionaba con un browser prototipo. El 29 de septiembre de 1994 se termina el desarrollo del prototipo de HotJava. Cuando se hace la demostración a los ejecutivos de Sun, se reconoce el potencial de Java y se acepta el proyecto.

En 1995, en la conferencia SunWorld, John Gage, de Sun Microsystems, y Marc Andreessen, cofundador y vicepresidente de Netscape, anunciaban la versión alpha de Java, que en ese momento sólo corría en Solaris, y el hecho de que Java iba a ser incorporado en Netscape Navigator, el navegador más utilizado de Internet.

Este mismo año Sun forma la empresa Java Soft para dedicarse al desarrollo de productos basados en la tecnología Java, y así trabajar con terceras partes para crear aplicaciones, herramientas, sistemas de plataforma y servicios para aumentar las capacidades del lenguaje. Ese mismo mes aparece la versión 1.0 del JDK.

Con la segunda alpha de Java en Julio de 1995, se añade el soporte para Windows NT y en la tercera, en Agosto, para Windows 95.

Netscape Communications decide apoyar a Java applets en Netscape Navigator 2.0. Ese fue el factor clave que lanzó a Java a ser conocido y famoso. Y como parte de su estrategia de crecimiento mundial y para favorecer la promoción de la nueva

tecnología, Java Soft otorgó permisos para otras compañías para que pudieran tener acceso al código fuente y al mismo tiempo mejorar sus navegadores.

También les permitía crear herramientas de desarrollo para programación Java y los facultaba para acondicionar máquinas virtuales Java (JVM), a varios sistemas operativos.

Muy pronto las licencias o permisos contemplaban prestigiosas firmas como: IBM, Microsoft, Symantec, Silicon Graphics, Oracle, Toshiba y Novell.

Los apples Java (basados en JDK 1.02) son apoyados por los dos más populares navegadores web (Nestcape Navigator 3.0 y Microsoft Internet Explorer 3.0. I.B.M./Lotus, Computer Asociates, Symantec, Informix, Oracle, Sybase y otras poderosas empresas de software están construyendo Software 100% puro JAVA, por ejemplo el Corel Office que actualmente está en versión Beta.

Los nuevos proyectos de Java son co-patrocinados por cientos de millones de dólares en capital disponible de recursos tales como la Fundación Java, un fondo común de capital formado por 11 compañías, incluyendo Cisco Systems, IBM, Netscape y Oracle.

Hoy en día, puede encontrar la tecnología Java en redes y dispositivos que comprenden desde Internet y superordenadores científicos hasta portátiles y teléfonos móviles; desde simuladores de mercado en Wall Street hasta juegos de uso doméstico y tarjetas de crédito.[1]

2.2.2. Principales Características

- ✓ **Lenguaje simple.**

Java posee una curva de aprendizaje muy rápida. Resulta relativamente sencillo escribir applets interesantes desde el principio.

- ✓ **Orientado a Objetos.**

Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos.

1 http://www.cad.com.mx/historia_del_lenguaje_java.htm

- ✓ **Distribuido.**

Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

- ✓ **Robusto.**

Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.

- ✓ **Seguridad.**

Dada la naturaleza distribuida de Java, donde las applets se bajan desde cualquier punto de la Red, la seguridad se impuso como una necesidad de vital importancia.

- ✓ **Multihebra.**

Java soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas.

- ✓ **Produce Applets.**

Java puede ser usado para crear dos tipos de programas: aplicaciones independientes y applets. Las aplicaciones independientes se comportan como cualquier otro programa escrito en cualquier lenguaje, como por ejemplo el navegador de Web HotJava, escrito íntegramente en Java. Por su parte, las applets son pequeños programas que aparecen embebidos en las páginas Web, como aparecen los gráficos o el texto, pero con la capacidad de ejecutar acciones muy complejas, como animar imágenes, establecer conexiones de red, presentar menús y cuadros de diálogo para luego emprender acciones.[2]

2.2.3. Clasificación

Sun Microsystems dividió el lenguaje en varias versiones para diferentes tipos de

2 http://www.iec.csic.es/criptonomicon/java/quesjava.html

aplicaciones, estas versiones son las siguientes:

- Plataforma de Java Edición Empresarial (java Platform EE).
- Plataforma de Java Edición Estandar (java Platform SE).
- Plataforma de Java Edición Micro (java Platform ME).

2.2.3.1. Java Micro Edition

Esta versión de java está enfocada a la aplicación de la tecnología java en dispositivos electrónicos con capacidades computacionales y gráficas muy reducidas, tales como teléfonos móviles, PDAs Esta edición tiene unos componentes básicos que la diferencian de otras versiones, como el uso de una máquina virtual denominada KVM (Kilo Virtual Machine, debido a que requiere sólo unos pocos Kilobytes de memoria para funcionar) en lugar de JVM clásica.

- **ARQUITECTURA J2ME**

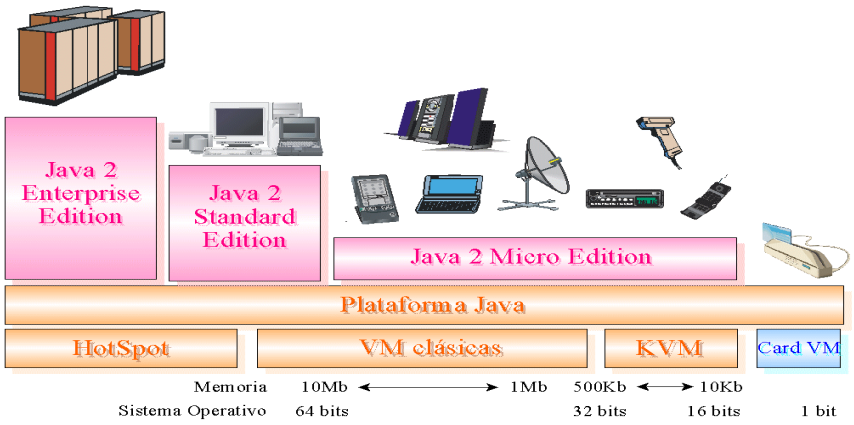


Figura. 2.1: Arquitectura de la plataforma Java 2 de Sun Microsystems.[3]

La figura 2.1. muestra la arquitectura Java™ 2 de Sun Microsystems dentro de la cual se describe la estructura de J2ME, J2SE y J2EE.

Java™ 2 Micro Edition que está orientada a pequeños dispositivos y sistemas embebidos como son teléfonos móviles, PDAs, Set-Top Boxes, máquinas expendedoras etc.

Al igual que sucede con J2EE™, que está orientado a entornos corporativos o J2SE™, orientado a sistemas de sobremesa, la arquitectura J2ME está formada por un conjunto de APIs estándares que permiten que las aplicaciones desarrolladas se

3 http://www.lcc.uma.es/~galvez/ftp/libros/J2ME.pdf

beneficien de las características multiplataforma de Java y que abren la puerta a la distribución de aplicaciones a millones de dispositivos.

La figura 2.2 muestra la arquitectura J2ME que se divide en dos grandes bloques de arquitecturas dependiendo del tipo de dispositivo y las características de los mismos.

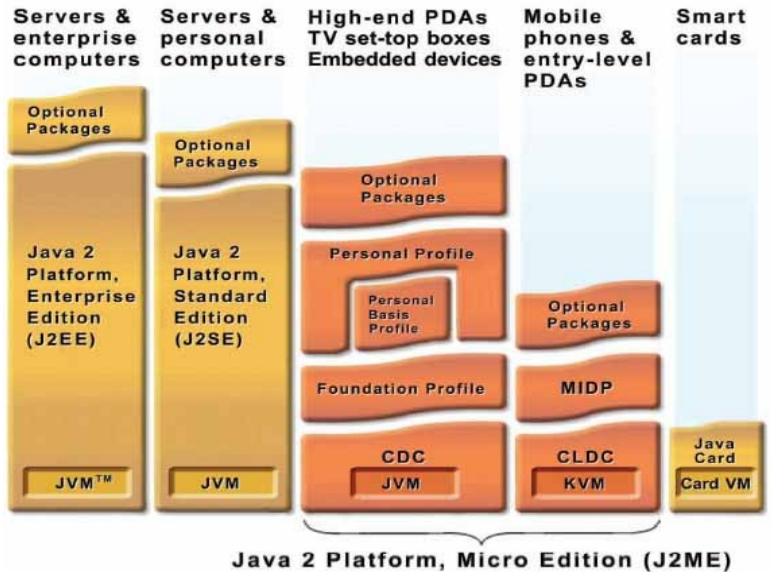


Figura 2.2: Arquitectura J2ME.[4]

Para poder tener un entorno de ejecución Java para J2ME que cumpla los requisitos de un rango amplio de dispositivos es necesario que se componga de:

- ✓ configuración
- ✓ perfiles
- ✓ paquetes opcionales

- **Configuraciones**

Las configuraciones se componen de una máquina virtual y un conjunto mínimo de bibliotecas de función. Proporcionan la funcionalidad básica para un conjunto de dispositivos que comparten características similares, tales como gestión de memoria o conectividad a la red.

En la actualidad existen dos configuraciones J2ME:

- ✓ Connected Limited Device Configuration (CLDC)

4 http://grasia.fdi.ucm.es/j2me/images/j2meLayers.jpg

- ✓ Connected Device Configuration (CDC)

- **Perfiles**

Para conformar un entorno de ejecución completo orientado a una categoría de dispositivos, las configuraciones se han de combinar con un conjunto de APIs de un nivel más alto, llamadas perfiles, que van un paso más allá en la definición del modelo de ciclo de vida de las aplicaciones, la interfaz de usuario y acceso a las propiedades específicas de los dispositivos.

En la actualidad existen los siguientes perfiles asociados a J2ME:

- Mobile Information Device Profile (MIDP)
- Foundation Profile
- Personal Profile
- Personal Basis Profile

- **Mobile Information Device Profile (MIDP)**

Está diseñado para teléfonos móviles y PDAs con capacidades básicas. Ofrece la funcionalidad básica para las aplicaciones móviles, incluyendo la interfaz de usuario, conectividad a redes, almacenamiento local de datos y gestión del ciclo de vida de las aplicaciones.

Al combinarlo con la configuración CLDC, MIDP proporciona un entorno de ejecución Java completo que incrementa la capacidad de los dispositivos móviles y que reduce el consumo de memoria y energía.[5]

2.2.4. Sofware de Desarrollo

- ✓ **Netbeans**

El IDE NetBeans es un entorno de desarrollo integrado - una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso.

5 http://grasia.fdi.ucm.es/j2me/_J2METech/index.html

El NetBeans IDE es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans. El NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). [6]

2.3. Android

2.3.1. Reseña Histórica

El lanzamiento inicial del Android Software Development Kit apareció en noviembre de 2007 y ya mucho tiempo después apareció el Android 0.9 SDK en beta. Al otro mes finalmente lanzaron Android 1.0 SDK (Release 1). Seis meses después -principios de marzo 2009-, Google presentó la versión 1.1 de Android para el "dev phone" y la actualización incluía algunos cambios estéticos menores además de soporte para "búsquedas por voz.

A mediados de mayo 2009, Google lanza la versión 1.5 de Android OS (llamada Cupcake) con su respectivo SDK que incluía nuevas caracteristicas como: grabación de video, soporte para stereo Bluetooth, sistema de teclado personalizable en pantalla, reconocimiento de voz y el AppWidget framework que permitió que los desarrolladores puedan crear sus propios widgets para la página principal. Android 1.5 fue la versión que más personas usaron para iniciarse en Android (con el T-Mobile G1 y HTC Dream en USA) y sigue siendo actualmente una versión que se encuentra disponible en muchos móviles Android como el HTC Hero o varios de los nuevos MOTOBLUR como el Motorola Backflip o Motorola Dext.

Luego apareció Android 1.6 "Donut" en septiembre de 2009 con mejoras en las búsquedas, indicador de uso de batería y hasta el VPN control applet. De hecho, esta versión fue tan buena que todos los Android que no tienen una interfaz personalizada como HTC Sense o Motoblur ahora corren 1.6, incluyendo el T-Mobile G1, y en la actualidad sigue siendo la versión más popular.

Poco después, el Google Nexus One (al cuál marcó un antes y un después ya que Google trató de venderlo por su cuenta y liberado, además de en algunas operadoras) llegó con Android 2.1 (el cual algunos llamaron "Flan" pero Google sigue considerándolo parte de "Eclair") con nuevas capacidades 3D, live wallpapers y lo que significó la gran mejora de la plataforma desde 1.6. De hecho, todos están

6 http://es.wikipedia.org/wiki/NetBeans

pidiendo que les actualicen a 2.1 sus propios dispositivos con Android.[7]

2.3.2. Características

- ✓ **Diseño de dispositivo**

La plataforma es adaptable a pantallas más grandes, VGA, biblioteca de gráficos 2D, biblioteca de gráficos 3D basada en las especificaciones de la OpenGL ES 2.0 y diseño de teléfonos tradicionales.

- ✓ **Almacenamiento**

SQLite, una base de datos liviana, que es usada para propósitos de almacenamiento de datos.

- ✓ **Conectividad**

Android soporta las siguientes tecnologías de conectividad: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE y WiMAX.

- ✓ **Mensajería**

SMS y MMS son formas de mensajería, incluyendo mensajería de texto y ahora la Android Cloud to Device Messaging Framework (C2DM) es parte del servicio de Push Messaging de Android.

- ✓ **Navegador Web**

El navegador web incluido en Android está basado en el motor de renderizado de código abierto WebKit, emparejado con el motor JavaScript V8 de Google Chrome. El navegador obtiene una puntuación de 93/100 en el test Acid3.

- ✓ **Soporte de Java**

Aunque las aplicaciones son escritas en Java, no hay una Máquina Virtual de Java en la plataforma. El código Java no es ejecutado. El código Java se compila en el ejecutable Dalvik y corre en la Máquina Virtual Dalvik. Dalvik es una máquina virtual especializada, diseñada específicamente para Android y optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados. El soporte para J2ME puede ser agregado mediante aplicaciones de terceros como el

7 http://www.celularis.com/software/historia-android.php

J2ME MIDP Runner.

- ✓ **Soporte Multimedia**

Android soporta los siguientes formatos multimedia: WebM, H.263, H.264 (en 3GP o MP4), MPEG-4 SP, AMR, AMR-WB (en un contenedor 3GP), AAC, HE-AAC (en contenedores MP4 o 3GP), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF y BMP.

- ✓ **Soporte Para Streaming**

Streaming RTP/RTSP (3GPP PSS, ISMA), descarga progresiva de HTML (HTML5 <video> tag). Adobe Flash Streaming (RTMP) es soportado mediante el Adobe Flash Player. Se planea el soporte de Microsoft Smooth Streaming con el port de Silverlight a Android. Adobe Flash HTTP Dynamic Streaming estará disponible mediante una actualización de Adobe Flash Player.

- ✓ **Soporte para hardware Adicional**

Android soporta cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, giroscopios, magnetómetros, sensores de proximidad y de presión, termómetro, aceleración 2D y 3D.

- ✓ **Entorno de desarrollo**

Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software. El entorno de desarrollo integrado es Eclipse (actualmente 3.4 o 3.5) usando el plugin de Herramientas de Desarrollo de Android.

- ✓ **Market**

El Android Market es un catálogo de aplicaciones gratuitas o de pago en el que pueden ser descargadas e instaladas en dispositivos Android sin la necesidad de un PC.

- ✓ **Multi-táctil**

Android tiene soporte nativo para pantallas multi-táctiles que inicialmente hicieron su aparición en dispositivos como el HTC Hero. La funcionalidad fue originalmente desactivada a nivel de kernel (posiblemente para evitar infringir patentes de otras compañías). Más tarde, Google publicó una actualización para el Nexus One y el

Motorola Droid que activa el soporte para pantallas multi-táctiles de forma nativa.

- ✓ **Bluetooth**

El soporte para A2DF y AVRCP fue agregado en la versión 1.5; el envío de archivos (OPP) y la exploración del directorio telefónico fueron agregados en la versión 2.0; y el marcado por voz junto con el envío de contactos entre teléfonos lo fueron en la versión 2.2.

- ✓ **Videollamada**

Android soporta videollamada através de Google Talk desde su versión HoneyComb.

- ✓ **Multitarea**

Multitarea real de aplicaciones está disponible, es decir, las aplicaciones que no estén ejecutándose en primer plano reciben ciclos de reloj, a diferencia de otros sistemas de la competencia en la que la multitarea es congelada

- ✓ **Características basadas en voz**

La búsqueda en Google a través de voz está disponible como "Entrada de Búsqueda" desde la versión inicial del sistema.

- ✓ **Tethering**

Android soporta tethering, que permite al teléfono ser usado como un punto de acceso alámbrico o inalámbrico (todos los teléfonos desde la versión 2.2, no oficial en teléfonos con versión 1.6 o superiores mediante aplicaciones disponibles en el Android Market, por ejemplo PdaNet). Para permitir a un PC usar la conexión 3G del móvil android se podría requerir la instalación de software adicional.[8]

2.3.3. Arquitectura

La figura 2.3 muestra los componentes principales del sistema operativo de Android en el que se muestra las aplicaciones en el bloque azul como calendarios, mapas, contactos y una gran variedad de aplicaciones que se las puede realizar, las librerías disponibles en el bloque verde y el núcleo del cual depende Android que es Linux en el bloque de color rojo.

8 http://es.wikipedia.org/wiki/Android#Caracter.C3.ADsticas

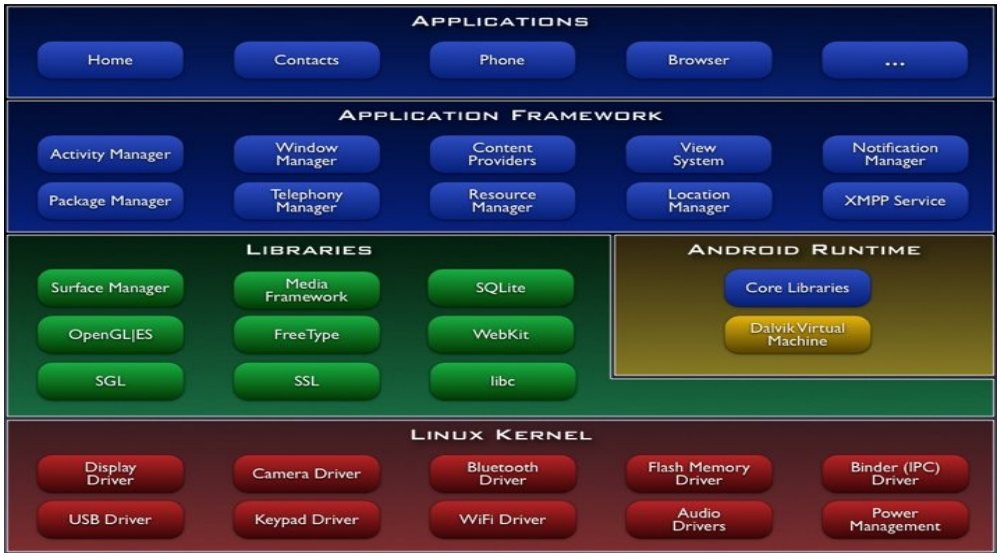


Figura 2.3: Arquitectura Android.[9]

2.4.Radiofrecuencia

Aunque se emplea la palabra radio, las transmisiones de televisión, radio, radar y telefonía móvil están incluidas en esta clase de emisiones de radiofrecuencia. Otros usos son audio, vídeo, radionavegación, servicios de emergencia y transmisión de datos por radio digital; tanto en el ámbito civil como militar. También son usadas por los radioaficionados.[10]

2.5. Bluetooth

Bluetooth se denomina al protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, con una cobertura baja y basada 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 por qué estar alineados, pueden incluso estar en habitaciones separadas si la potencia de transmisión lo permite.

9 http://es.wikipedia.org/wiki/Archivo:System-architecture.jpg
10 http://es.wikipedia.org/wiki/Radiofrecuencia

La clasificación de los dispositivos Bluetooth como "Clase 1", "Clase 2" o "Clase 3" es únicamente una referencia de la potencia de trasmisión del dispositivo, siendo totalmente compatibles los dispositivos de una clase con los de la otra.

Clase	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	~10 metros
Clase 3	1 mW	0 dBm	~1 metro

Tabla 2.1: Clasificación de los dispositivos Bluethooth.[11]

Cabe mencionar que en la mayoría de los casos, la cobertura efectiva de un dispositivo de clase 2 se extiende cuando se conecta a un transceptor de clase 1. Esto es así gracias a la mayor sensibilidad y potencia de transmisión del dispositivo de clase 1. Es decir, la mayor potencia de transmisión del dispositivo de clase 1 permite que la señal llegue con energía suficiente hasta el de clase 2. En cuanto al ancho de banda:

Versión	Ancho de banda
Versión 1.2	1 Mbit/s
Versión 2.0 + EDR	3 Mbit/s
UWB Bluetooth (propuesto)	53 - 480 Mbit/s

Tabla 2.2: Clasificación de dispositivos Bluethooth según su ancho de banda.[12]

11 http://es.wikipedia.org/wiki/Bluetooth
12 http://es.wikipedia.org/wiki/bluethooth

Bluetooth es la norma que define un estándar global de comunicación inalámbrica, que posibilita la transmisión de voz y datos entre diferentes equipos mediante un enlace por radiofrecuencia. Los principales objetivos que se pretende conseguir con esta norma son:

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

La tecnología Bluetooth comprende hardware, software y requerimientos de interoperatibilidad, por lo que para su desarrollo ha sido necesaria la participación de los principales fabricantes de los sectores de las telecomunicaciones y la informática, tales como: Ericsson, Nokia, Toshiba, IBM, Intel entre otros. Posteriormente se han ido incorporando muchas más compañías. Con estos avances se puede tener un panorama de total conectividad de los aparatos eléctricos y electrónicos en casa como en el trabajo.

2.5.1. La Interfase Aérea Bluetooth

El primer objetivo para los productos de primera generación son los entornos de la gente de negocios que viaja frecuentemente. Por lo que se debería pensar en integrar el chip de radio Bluetooth en equipos como: PCS portátiles, teléfonos móviles, PDAs y auriculares. Esto originaba una serie de cuestiones previas que deberían solucionarse tales como:

- El sistema deberá operar en todo el mundo.
- El emisor de radio deberá consumir poca energía, ya que debe integrarse en equipos alimentados por baterías.
- La conexión deberá soportar voz y datos, y por lo tanto aplicaciones multimedia.

2.5.2. Banda de Frecuencia Libre

Para poder operar en todo el mundo es necesaria una banda de frecuencia abierta a

cualquier sistema de radio independientemente del lugar del planeta donde se encuentre dicho sistema. Sólo la banda ISM (médico-científica internacional) de 2,45 Ghz cumple con éste requisito, con rangos que van de los 2.400 Mhz a los 2.500 Mhz, con algunas restricciones en países como Francia, España y Japón.

2.5.3. Salto de Frecuencia

Debido a que la banda ISM está abierta a cualquiera, el sistema de radio Bluetooth deberá estar preparado para evitar las múltiples interferencias que se puedan producir. Éstas pueden ser evitadas utilizando un sistema que busque una parte no utilizada del espectro o un sistema de salto de frecuencia. Los sistemas de radio Bluetooth utilizan el método de salto de frecuencia debido a que esta tecnología puede ser integrada en equipos de baja potencia y bajo costo. Este sistema divide la banda de frecuencia en varios canales de salto, donde, los transceptores, durante la conexión van cambiando de uno a otro de manera pseudo-aleatoria. Con esto se consigue que el ancho de banda instantáneo sea muy pequeño y también una propagación sobre el total de la banda. En conclusión, con el sistema FH (Salto de frecuencia), se pueden conseguir transceptores de banda estrecha con máxima inmunidad a las interferencias.

2.5.4. Definición de Canal

Bluetooth utiliza un sistema FH/TDD (salto de frecuencia/división de tiempo duplex), en el que el canal queda dividido en intervalos de 625 μs, llamados slots, donde cada salto de frecuencia es ocupado por un slot. Ésto da lugar a una frecuencia de salto de 1600 veces por segundo. Un paquete de datos puede ocupar un slot para la emisión y otro para la recepción y pueden ser usados alternativamente, dando lugar a un esquema de tipo TDD.

Dos o más unidades pueden compartir el mismo canal dentro de una red de de dispositivos que se conectan entre sí utilizando bluethooth (piconet), donde una unidad actúa como maestra, controlando el tráfico y las demás como esclavas. El salto de frecuencia del canal es determinado por la secuencia, es decir, el orden en que llegan los saltos y por la fase de esta secuencia. En Bluetooth, la secuencia está determinada por la identidad de la unidad maestra de la piconet (un código único para cada equipo), y por su frecuencia de reloj. Por lo que, para que una unidad esclava pueda sincronizarse con una unidad maestra, ésta primera debe añadir un

ajuste a su propio reloj nativo y así poder compartir la misma portadora de salto. En países donde la banda está abierta a 80 canales o más, espaciados todos ellos a 1 Mhz., se han definido 79 saltos de portadora, y en aquellos donde la banda es más estrecha se han definido 23 saltos.

2.5.5. Definición de Paquete

En cada slot, un paquete de datos puede ser intercambiado entre la unidad maestra y una de las esclavas. Cada paquete comienza con un código de acceso de 72 bits, que se deriva de la identidad maestra, seguido de un paquete de datos de cabecera de 54 bits. Este contiene importante información de control, como tres bits de acceso de dirección, tipo de paquete, bits de control de flujo, bits para la retransmisión automática de la pregunta, y chequeo de errores de campos de cabeza. Finalmente, el paquete que contiene la información, que puede seguir o no al de cabeza, puede tener una longitud de 0 a 2745 bits. Cada paquete que se intercambia en el canal es precedido por el código de acceso. Los receptores de la piconet comparan las señales que reciben con el código de acceso, si éstas no coinciden, el paquete recibido no es considerado como válido en el canal y el resto de su contenido es ignorado.

2.5.6. Definición de Enlace Físico

Se han definido dos tipos de enlace para poder soportar aplicaciones multimedia:

- Enlace de sincronización de conexión orientada (SCO)
- Enlace asíncrono de baja conexión (ACL)

Los enlaces SCO soportan conexiones asimétricas, punto a punto, usadas normalmente en conexiones de voz. Estos enlaces están definidos en el canal, reservándose dos slots consecutivos (envío y retorno) en intervalos fijos. Los enlaces ACL soportan conmutaciones punto a punto, simétricas o asimétricas típicamente usadas en la transmisión de datos.

2.5.7. Inmunidad a las Interferencias

Como se mencionó anteriormente Bluetooth opera en una banda de frecuencia que está sujeta a considerables interferencias, por lo que el sistema ha sido optimizado para evitar estas interferencias. Las técnicas de salto de frecuencia son aplicadas a

una alta velocidad y una corta longitud de los paquetes (1600 saltos/segundo, para slots-simples), si un paquete se perdiese, sólo una pequeña parte de la información se perdería. Los paquetes de datos están protegidos por un esquema ARQ (repetición automática de consulta), en el cual los paquetes perdidos son automáticamente retransmitidos. La voz no se retransmite nunca, sin embargo, se utiliza un esquema de codificación muy robusto. Este esquema, que está basado en una modulación variable de declive delta (CSVD), sigue la forma de la onda de audio, y es muy resistente a los errores de bits. Éstos son percibidos como ruido de fondo, que se intensifica si los errores aumentan.

2.5.8. Modulación por Desplazamiento de Frecuencia Gausiana

La modulación por desplazamiento de frecuencia gausiana, (en inglés Gaussian Frequency Shift Keying o GFSK), es un tipo de modulación donde un 1 lógico es representado mediante una desviación positiva (incremento) de la frecuencia de la onda portadora, y un 0 mediante una desviación negativa (decremento) de la misma.

GFSK es una versión mejorada de la modulación por desplazamiento de frecuencia (FSK). En GFSK la información es pasada por un filtro gausiano antes de modular la señal. Esto se traduce en un espectro de energía más estrecho de la señal modulada, lo cual permite mayores velocidades de transferencia sobre un mismo canal.

2.5.9. Pila de Protocolos de Bluetooth

En la figura 2.4 se muestra la pila completa de protocolos de Bluetooth, en la parte superior de la pila están las aplicaciones que soportan los modelos de uso de Bluetooth y en la parte inferior se encuentran las capas comunes, tal como la Banda Base. No todas las aplicaciones hacen uso de todos los protocolos que se muestran en la figura 2.4 sino que funcionan dependiendo de las necesidades de la aplicación.

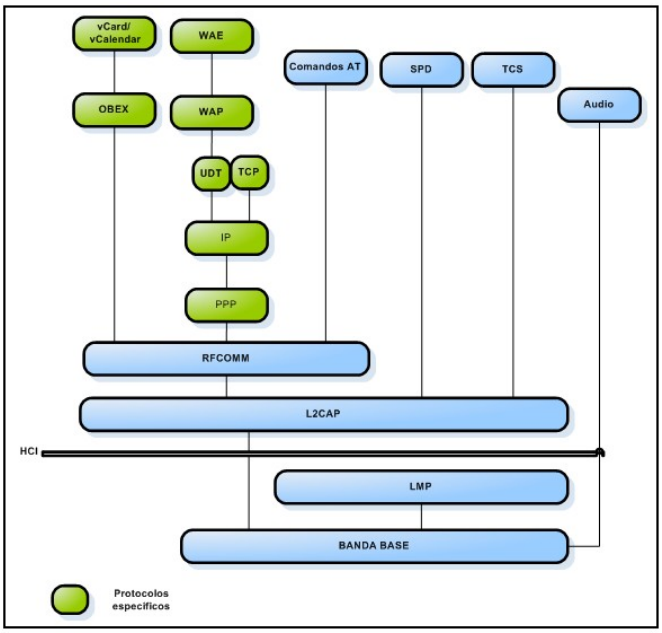


Figura 2.4: Stack de Protocolo Bluetooth.[13]

- **Banda Base** es la capa de comunicación más baja que implementa el canal físico real y emplea una secuencia aleatoria de saltos a través de 79 frecuencias de radio diferentes. Los paquetes son enviados sobre el canal físico en una frecuencia de salto diferente. Controla la sincronización de las unidades Bluetooth y la secuencia de saltos en frecuencia, además es la responsable de la información para el control de enlace a bajo nivel como el reconocimiento, control de flujo y caracterización de carga útil y soporta dos tipos de enlace: síncrono orientado a la conexión (SCO), para datos y asíncrono no orientado a la conexión (ACL), principalmente para audio. Los dos pueden ser multiplexados para usar el mismo enlace RF, usando un ancho de banda reservado, los enlaces SCO soportan tráfico de voz en tiempo real.

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

13 http://www.electronicafacil.net/tutoriales/Protocolos-Bluetooth.php

- **Host Controller Interface (HCI)** o Interfaz del Controlador de Enlace brinda un método de interfaz uniforme para acceder a los recursos de hardware de Bluetooth. Éste contiene una interfaz de comando para el controlador Banda Base y la gestión de enlace e incluso para acceder al hardware.

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

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

- **Protocolo RFCOMM** ofrece emulación de puertos seriales sobre el protocolo L2CAP. RFCOMM emula señales de control y datos RS-232 sobre la Banda Base Bluetooth. Éste ofrece capacidades de transporte a servicios de capas superiores (por ejemplo OBEX) que usan una línea serial como mecanismo de transporte.

  - **RFCOMM** soporta dos tipos de comunicación, directa entre dispositivos actuando como endpoints y dispositivo-modem-dispositivo, además tiene un esquema para emulación de null modem.

- **Control de telefonía binario (TCS binario)** es un protocolo que define la señalización de control de llamadas, para el establecimiento y liberación de una conversación o una llamada de datos entre unidades Bluetooth. Además,

éste ofrece funcionalidad para intercambiar información de señalización no relacionada con el progreso de llamadas.

- **Audio** es una capa especial, usada sólo para enviar audio sobre Bluetooth. Las transmisiones de audio pueden ser ejecutadas entre una o más unidades usando muchos modelos diferentes. Los datos de audio no pasan a través de la capa L2CAP, pero sí directamente después de abrir un enlace y un establecimiento directo entre dos unidades Bluetooth.

La pila completa de Bluetooth comprende protocolos específicos como:

- **Control de telefonía** Bluetooth soporta un número de comandos AT para el control de telefonía a través de la emulación de puerto serial (RFCOMM).

- **Protocolo Punto a Punto (PPP)** el PPP es un protocolo orientado a paquetes y por lo tanto debe usar su mecanismo serial para convertir una corriente de paquetes de datos en una corriente de datos seriales. Este protocolo corre sobre RFCOMM para lograr las conexiones punto a punto.

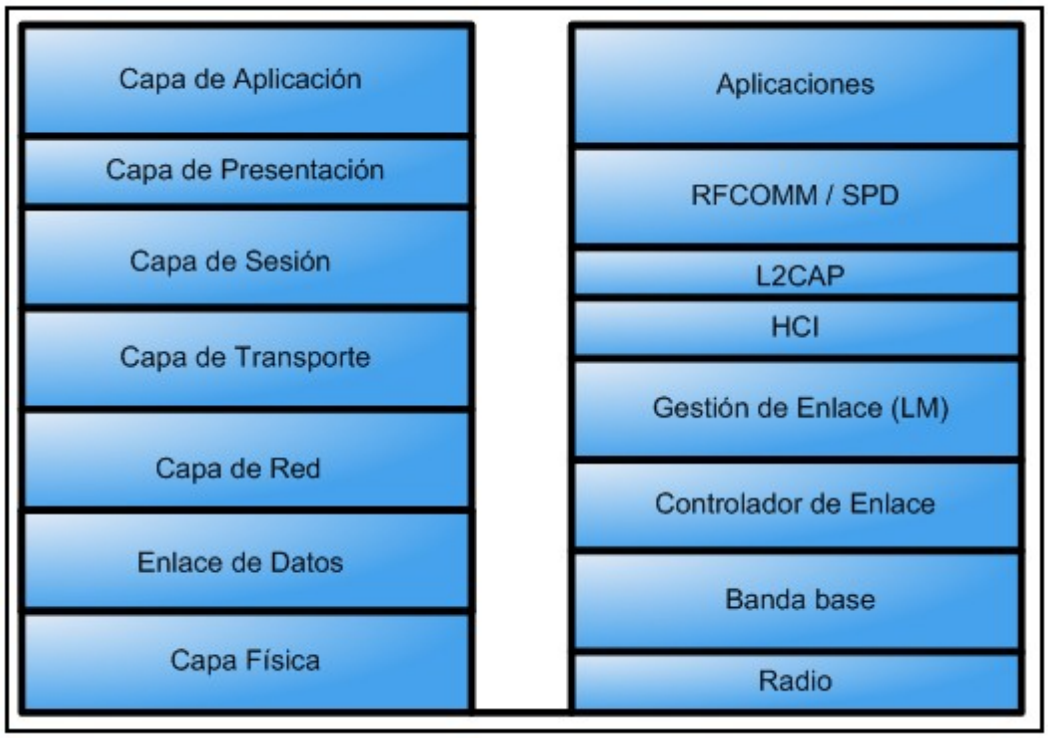
- **Protocolos UDP/TCP – IP** los estándares UDP/TCP e IP permiten a las unidades Bluetooth conectarse, por ejemplo a Internet, a través de otras unidades conectadas. Por lo tanto, la unidad puede actuar como un puente para Internet. La configuración TCP/IP/PPP está disponible como un transporte para WAP.

- **Wireless Application Protocol (WAP)** o Protocolo de Aplicación Inalámbrica, WAP es una especificación de un protocolo inalámbrico que trabaja con una amplia variedad de tecnologías de red inalámbricas conectando dispositivos móviles a Internet. Bluetooth puede ser usado como portador para ofrecer el transporte de datos entre el cliente WAP y su servidor de WAP adyacente. Además, las capacidades de red de Bluetooth dan a un cliente WAP posibilidades únicas en cuanto a lo que movilidad se refiere en comparación con otros portadores WAP.

- **Protocolo OBEX** el OBEX es un protocolo opcional de nivel de aplicación diseñado para permitir a las unidades Bluetooth soportar comunicación infrarroja para intercambiar una gran variedad de datos y comandos. Éste usa

un modelo cliente-servidor y es independiente del mecanismo de transporte y del API (Application Program Interface) de transporte. OBEX usa RFCOMM como principal capa de transporte.

La Figura 2.5 muestra una comparación del stack Bluetooth con el modelo de referencia estándar Open Systems Interconect, OSI, para stacks de protocolo de comunicaciones. A pesar de que Bluetooth no concuerda exactamente con el modelo, esta comparación es muy útil para relacionar las diferentes partes del stack Bluetooth con las capas del modelo OSI. Dado que el modelo de referencia es un stack ideal y bien particionado, la comparación sirve para resaltar la división de funciones en el stack Bluetooth.



a) Modelo de Referencia OSI **b)** Bluetooth

Figura 2.5: Modelo de referencia OSI y Bluetooth.[14]

2.6. Módulos de Transmisión y Recepción Inalámbrica

2.6.1. Módulo de Radiofrecuencia

Los módulos usados con mayor frecuencia son conocidos como TLP434(A) y RLP434(A), la “A” que está entre los paréntesis son modelos mejorados, pero la distribución de pines en los dos casos es la misma.

14 http://www.univalle.edu.co/~telecomunicaciones/trabajos_de_grado/informes/tg_OscarRodriguez_RicardoMaya.pdf

Se pueden trabajar con cualquiera de los dos tipos, no es necesario que los dos sean “A”, es decir, se puede enviar datos con el TLP434A y recibirlos con el RLP434 sin ningún problema, lo único que se debe tener en cuenta, es la velocidad y frecuencia a la que se están transmitiendo los datos.[15]

Transmisor TLP434A

Nuevo Transmisor inalámbrico ultra pequeño, ideal para proyectos de control remoto o la transferencia de datos a un objeto remoto. Las aplicaciones incluyen robots móviles, alarmas, control remoto y transferencia de datos inalámbricos. Esta unidad compacta opera desde 2V hasta 12V. Tiene un alcance de 150 metros al aire libre. La unidad se puede conectar directamente al codificador HT12E o similar.

Caracteristicas:

• Frecuencia - 434 MHz.
• Voltaje de operación - 1.5Vdc - 12 Vdc.
• Data Rate - 8000 bps.
• Compatible con el codificador HT12E o similar.[16]

Figura 2.6: Modulo de Transmisión TLP 434 A.[17]

Receptor RLP434

El nuevo receptor RLP434 es compacto y ligero, trabaja directamente con el

15 http://www.buenastareas.com/ensayos/Modulos-De-Transmision-y-Recepcion-Tipos-y/1714974.html
16 http://www.bilbaoelectronics.com/radio-modulos-rf.html
17 http://www.bilbaoelectronics.com/radio-transmisor-tlp434.jpg

transmisor TLP434A a una frecuencia de operaciones de 434MHz. Ideal para muchas aplicaciones caseras e insustriales, sin necesidad de un dificultosa conexión por cable.

Caracteristicas:

• Frecuencia - 434MHz
• Voltaje de operación - 3.5Vdc - 5.5 Vdc
• Data Rate - 6000 bps
• Compatible con decodificador HT12D o similar.[18]

Figura 2.7: Módulo de Recepción RLP 434 A.[19]

2.6.2. Módulo Bluetooth

Módulo Bluetooth - RN-42

Descripción

Este módulo de la compañía Roving Networks es pequeño y fácil de usar, está diseñado para reemplazar cables seriales. El stack Bluetooth está completamente encapsulado y el usuario final simplemente ve caracteres seriales siendo transmitidos y recibidos.

El RN-42 es un dispositivo Clase 2 con un rango de 15 metros a 18 metros con un reducido consumo de potencia. Es perfecto para aplicaciones de corto alcance alimentadas mediante batería. Usa solamente 26uA en modo "sleep" mientras permanece aún conectado y reconocible. Existen múltiples modos de configuración de potencia los cuales permiten al usuario elegir el diseño de menor consumo de potencia para una determinada aplicación.

Soporta múltiples perfiles Bluetooth como SPP y HID, así como la interface UART, lo cual facilita su integración simple con sistemas embebidos, (sistemas de comunicación con canal dedicado), o para su conexión con dispositivos existentes.

18 http://www.bilbaoelectronics.com/radio-modulos-rf.html
19 http://www.bilbaoelectronics.com/radio-receptor-rlp434.jpg

Características

- Módulo Bluetooth totalmente certificado
- Certificación FCC
- UART totalmente configurable
- Velocidades de transmisión UART hasta 3Mbps
- Velocidades de trsnamisión al aire de 721kbps a 2.0Mbps
- Modo sleep de baja potencia (low power sleep)
- Compatible con todos los productos Bluetooth que soportan SPP
- Incluye soporte para protocolos BCSP, DUN, LAN, GAP SDP, RFCOMM, y L2CAP
- Operación con 3.3V
- Pin de estado
- Compatible con la Tecnología Bluetooth v2.0
- Salida de potencia Clase 2.[20]

20http://www.tecbolivia.com/index2.php?page=shop.product_details&product_id=117&flypage=flypage.tpl&pop=1&option=com_virtuemart&Itemid=4

Figura 2.8: Módulo Bluethooth RN-42.[21]

2.7. Microcontroladores

Los PIC son una familia de microcontroladores tipo RISC fabricados por Microchip Technology Inc. y derivados del PIC1650, originalmente desarrollado por la división de microelectrónica de General Instrument.

El nombre actual no es un acrónimo. En realidad, el nombre completo es PICmicro, aunque generalmente se utiliza como Peripheral Interface Controller (controlador de interfaz periférico).

El PIC original se diseñó para ser usado con la nueva CPU de 16 bits CP16000. Siendo en general una buena CPU, ésta tenía malas prestaciones de E/S, y el PIC de 8 bits se desarrolló en 1975 para mejorar el rendimiento del sistema quitando peso de E/S a la CPU. El PIC utilizaba microcódigo simple almacenado en ROM para realizar estas tareas; y aunque el término no se usaba por aquel entonces, se trata de un diseño RISC que ejecuta una instrucción cada 4 ciclos del oscilador.

El PIC, sin embargo, se mejoró con EPROM para conseguir un controlador de canal programable. Hoy en día multitud de PICs vienen con varios periféricos incluidos (módulos de comunicación serie, UARTs, núcleos de control de motores, etc.) y con memoria de programa desde 512 a 32.000 palabras (una palabra corresponde a una instrucción en lenguaje ensamblador, y puede ser 12, 14 o 16 bits, dependiendo de la familia específica de PICmicro).[22]

Arquitectura central

La arquitectura del PIC es sumamente minimalista. Esta caracterizada por las siguientes prestaciones:

- ✓ Área de código y de datos separadas (Arquitectura Harvard).
- ✓ Un reducido número de instrucciones de largo fijo.
- ✓ La mayoría de las instrucciones se ejecutan en un sólo ciclo de ejecución (4 ciclos de clock), con ciclos de único retraso en las bifurcaciones y saltos.
- ✓ Un sólo acumulador (W), cuyo uso (como operador de origen) es implícito (no

21 http://www.tecbolivia.com/components/com_virtuemart/shop_image/product/M__dulo_Bluetoot_4dd290d121f28.jpg

22 http://es.wikipedia.org/wiki/Microcontrolador_PIC#Arquitectura_central

está especificado en la instrucción).

- ✓ Todas las posiciones de la RAM funcionan como registros de origen y/o de destino de operaciones matemáticas y otras funciones.[1]
- ✓ Una pila de hardware para almacenar instrucciones de regreso de funciones.
- ✓ Una relativamente pequeña cantidad de espacio de datos direccionable (típicamente, 256 bytes), extensible a través de manipulación de bancos de memoria.
- ✓ El espacio de datos está relacionado con el CPU, puertos, y los registros de los periféricos.
- ✓ El contador de programa está también relacionado dentro del espacio de datos, y es posible escribir en él (permitiendo saltos indirectos).

A diferencia de la mayoría de otros CPU, no hay distinción entre los espacios de memoria y los espacios de registros, ya que la RAM cumple ambas funciones, y esta es normalmente referida como "archivo de registros" o simplemente, registros.

Programadores

- ✓ PICStart Plus (puerto serie y USB)
- ✓ Promate II (puerto serie)
- ✓ MPLAB PM3 (puerto serie y USB)
- ✓ ICD2 (puerto serie y USB)
- ✓ ICD3 (USB)
- ✓ PICKit 1 (USB)
- ✓ IC-Prog 1.06B
- ✓ PICAT 1.25 (puerto USB2.0 para PICs y Atmel)
- ✓ WinPic 800 (puerto paralelo, serie y USB)

- ✓ PICKit 2 (USB)
- ✓ PICKit 3 (USB)
- ✓ Terusb1.0
- ✓ Eclipse (PICs y AVRs. USB.)
- ✓ MasterProg (USB)

PICs más comúnmente usados

- ✓ PIC12C508/509 (encapsulamiento reducido de 8 pines, oscilador interno, popular en pequeños diseños como el iPod remote).
- ✓ PIC12F629/675
- ✓ PIC16F84 (Considerado obsoleto, pero imposible de descartar y muy popular)
- ✓ PIC16F84A (Buena actualización del anterior, algunas versiones funcionan a 20 MHz, compatible 1:1)
- ✓ PIC16F628A (Es la opción típica para iniciar una migración o actualización de diseños antiguos hechos con el PIC16F84A. Posee puerto serial, módulos de comparacion análoga, PWM, módulo CCP, rango de operación de voltaje aumentado, entre otras)
- ✓ PIC16F88 (Nuevo sustituto del PIC16F84A con más memoria, oscilador interno, PWM, etc que podría convertirse en popular como su hermana).
- ✓ La subfamilia PIC16F87X y PIC16F87XA (los hermanos mayores del PIC16F84 y PIC16F84A, con cantidad de mejoras incluidas en hardware. Bastante común en proyectos de aficionados).
- ✓ PIC16F886/887 (Nuevo sustituto del 16F876A y 16F877A con la diferencia que el nuevo ya se incluye oscilador interno).
- ✓ PIC16F193x (Nueva gama media de PIC optimizado y con mucha RAM, ahora con 49 instrucciones por primera vez frente a las 35 de toda la vida).
- ✓ PIC18F2455 y similares con puerto USB 2.0

- ✓ PIC18F2550 manejo de puertos USB 2.0 y muy versatil.
- ✓ PIC18F452
- ✓ PIC18F4550
- ✓ dsPIC30F2010
- ✓ dsPIC30F3014
- ✓ dsPIC30F3011 (Ideales para control electrónico de motores eléctricos de inducción, control sobre audio).
- ✓ PIC32 (Nueva gama de PIC de 32 bits, los más modernos ya compatible con USB 2.0).[23]

2.8. ATMEL AVR

Los AVR son una familia de microcontroladores RISC de Atmel. La arquitectura de los AVR fue concebida por dos estudiantes en el Norwegian Institute of Technology, y posteriormente refinada y desarrollada en Atmel Norway, la empresa subsidiaria de Atmel, fundada por los dos arquitectos del chip.

El AVR es una CPU de arquitectura Harvard. Tiene 32 registros de 8 bits. Algunas instrucciones sólo operan en un subconjunto de estos registros. La concatenación de los 32 registros, los registros de entrada/salida y la memoria de datos conforman un espacio de direcciones unificado, al cual se accede a través de operaciones de carga/almacenamiento. A diferencia de los microcontroladores PIC, el stack se ubica en este espacio de memoria unificado, y no está limitado a un tamaño fijo.

El AVR fue diseñado desde un comienzo para la ejecución eficiente de código C compilado. Como este lenguaje utiliza profusamente punteros para el manejo de variables en memoria, los tres últimos pares de registros internos del procesador son usados como punteros de 16 bit al espacio de memoria externa, bajo los nombres X, Y y Z. Esto es un compromiso que se hace en arquitecturas de ocho bit desde los tiempos de Intel 8008, ya que su tamaño de palabra nativo de 8 bit (256 localidades accedidas) es pobre para direccionar. Por otro lado, hacer que todo el banco superior de 16 registros de 8 bit tenga un comportamiento alterno como un banco de

23 http://es.wikipedia.org/wiki/Microcontrolador_PIC#Arquitectura_central

8 registros de 16 bit, complicaría mucho el diseño, violando la premisa original de su simplicidad. Además, algunas instrucciones tales como 'suma inmediata' ('add immediate' en inglés) faltan, ya que la instrucción 'resta inmediata' ('substract immediate' en inglés) con el complemento dos puede ser usada como alternativa.

El set de instrucciones AVR está implementado físicamente y disponible en el mercado en diferentes dispositivos, que comparten el mismo núcleo AVR pero tienen distintos periféricos y cantidades de RAM y ROM: desde el microcontrolador de la familia Tiny AVR ATtiny11 con 1KB de memoria flash y sin RAM (sólo los 32 registros), y 8 pines, hasta el microcontrolador de la famila Mega AVRATmega2560 con 256KB de memoria flash, 8KB de memoria RAM, 4KB de memoria EEPROM, conversor análogo digital de 10 bits y 16 canales, temporizadores, comparador analógico, JTAG, etc. La compatibilidad entre los distintos modelos es preservada en un grado razonable.

El set de instrucciones de los AVR es más regular que la de la mayoría de los microcontroladores de 8-bit (por ejemplo, los PIC). Sin embargo, no es completamente ortogonal:

- ✓ Los registros punteros X, Y y Z tienen capacidades de direccionamiento diferentes entre sí (ver más arriba por qué).
- ✓ Los registros 0 al 15 tienen diferentes capacidades de direccionamiento que los registros 16 al 31.
- ✓ Las registros de I/O 0 al 31 tienen distintas características que las posiciones 32 al 63.
- ✓ La instrucción CLR afecta los 'flag', mientras que la instrucción SER no lo hace, a pesar de que parecen ser instrucciones complementarias (dejar todos los bits en 1, y dejar todos los bits en 0, respectivamente).
- ✓ Los códigos de operación 0x95C8 y 0x9004 hacen exactamente lo mismo (LPM).[24]

24 http://es.wikipedia.org/wiki/AV

CAPÍTULO III

DISEÑO IMPLEMENTACIÓN Y MOTAJE DEL ENTRENADOR LÓGICO

3.1. INTRODUCCIÓN

En el presente capítulo se detalla el proceso de Diseño e Implementación del Entrenador Lógico, para el cual se ha tomado en cuenta varias etapas que son:

- Diagrama de bloques del Entrenador Lógico.
- Diseño para los dispositivos de entrada.
- Diseño para los dispositivos de salida.
- Quemador de PIC.
- Diseño de la etapa de comunicación inalámbrica.
- Diseño de la fuente voltaje.
- Diseño del chasis del Entrenador Lógico.
- Diseño de las 10 prácticas con sus respectivas guías.

Además en el presente capítulo se describe la programación en J2ME y ANDROID como introducción a las prácticas a desarrollarse en las guías.

3.2. DIAGRAMA DE BLOQUES DEL ENTRENADOR LÓGICO

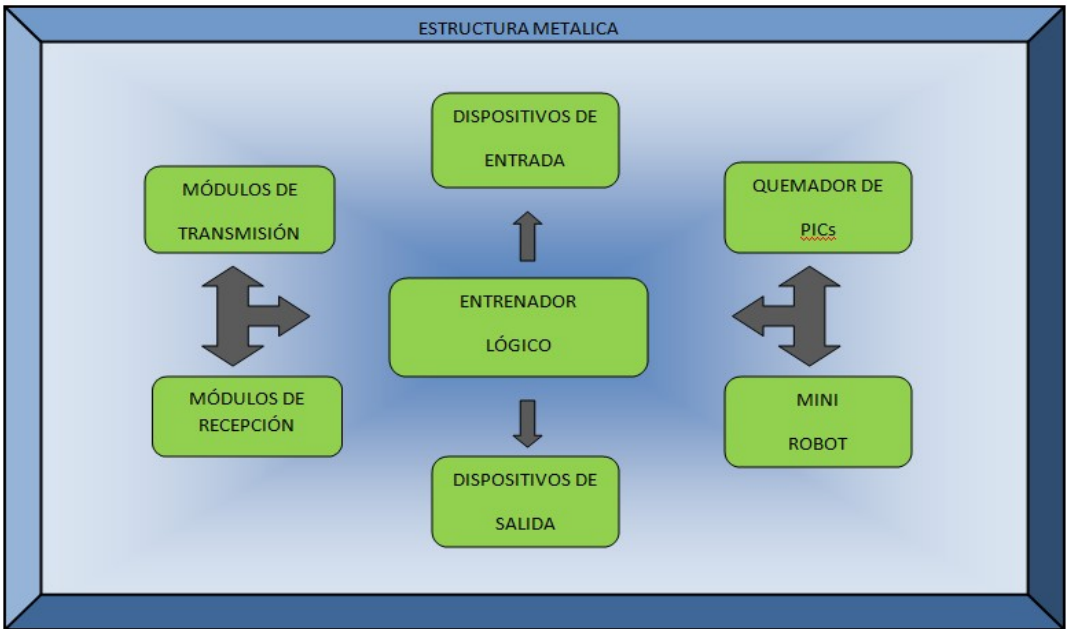


Figura 3.1: Diagrama de bloques del Entrenador Lógico.

La figura 3.1 muestra el diagrama de bloques del Entrenador Lógico que está compuesto por siete bloques y uno adicional que muestra la estructura metálica que cubre la circuitería del Entrenador Lógico, esta estructura metálica sirve como un punto común (negativo ó GND) de todos los circuitos de una misma placa o baquelita.

La primera parte del diseño hace referencia a los dispositivos de entrada que componen los elementos que emiten señales, logrando que los circuitos trabajen de acuerdo al diseño establecido, para este caso se utiliza los siguientes dispositivos: pulsadores, Dip-switch, potenciómetro digital y teclado matricial.

La segunda parte del diseño es el de dispositivos de salida, que son elementos que dan una respuesta o una señal de acuerdo a al dispositivo de entrada.
El Entrenador Lógico dispone de elementos de salida como: diodos led, buzer, parlante, display, relés, LCD y dentro de este diseño está también el diseño del mini

robot.

Para el quemador de PIC's, se establece un circuito totalmente independiente al Entrenador Lógico .

También consta del diseño de las etapas de transmisión y recepción mediante los módulos de bluethooth y radiofrecuencia que son las encargadas de la comunicación inalámbrica.

A continuación se realiza el diseño de una fuente de alimentación que permita el correcto funcionamiento de todos los bloques al mismo tiempo.

Finalmente se diseña la estructura que cubre la circuitería interna del Entrenador Lógico.

3.3. Diseño

3.3.1. Diseño Para Los Dispositivos de Entrada

Los dispositivos que son parte de este diseño son:

- Pulsadores
- Dip-switch
- Teclado Matricial
- Potenciómetro Digital

Para cada uno de estos dispositivos se requiere un circuito independiente para poder acoplar estos circuitos a las prácticas establecidas por el Entrenador Lógico.

PULSADORES

En las prácticas establecidas por el Entrenador Lógico se utiliza un microcontrolador el mismo que posee las siguientes características.

Voltaje de alimentación: 5vdc

Corriente máxima de entrada por cada pin: 25mA

Entonces para trabajar con los niveles apropiados de corriente en cada pin se utiliza una resistencia mínima que la calculamos de la siguiente manera:

$$V = R \times I$$

Ecuación 3.1

$$R = \frac{5V}{0{,}025\,A}$$

$$R = 200\,\Omega$$

$$R \approx 220\ \Omega$$

Esto quiere decir que se coloca una resistencia de 220Ω para estar al límite de la capacidad que soporta el PIC, pero no es ideal trabajar al límite,[25] sino que con una corriente de 1mA será suficiente para el correcto funcionamiento de cada pin. Entonces aplicando la ecuación 3.1 se tiene que:

$$R = \frac{5V}{0{,}001\,A}$$

$$R = 5000\,\Omega$$

$$R = 5\ K\Omega$$

$$R \approx 4{,}7\ K\Omega$$

Con todos los cálculos realizados se tiene el siguiente circuito para conectar uno o varios pulsadores, sin que el PIC sufra ningún daño, el Entrenador Lógico estará compuesto por 5 pulsadores en la siguiente disposición:

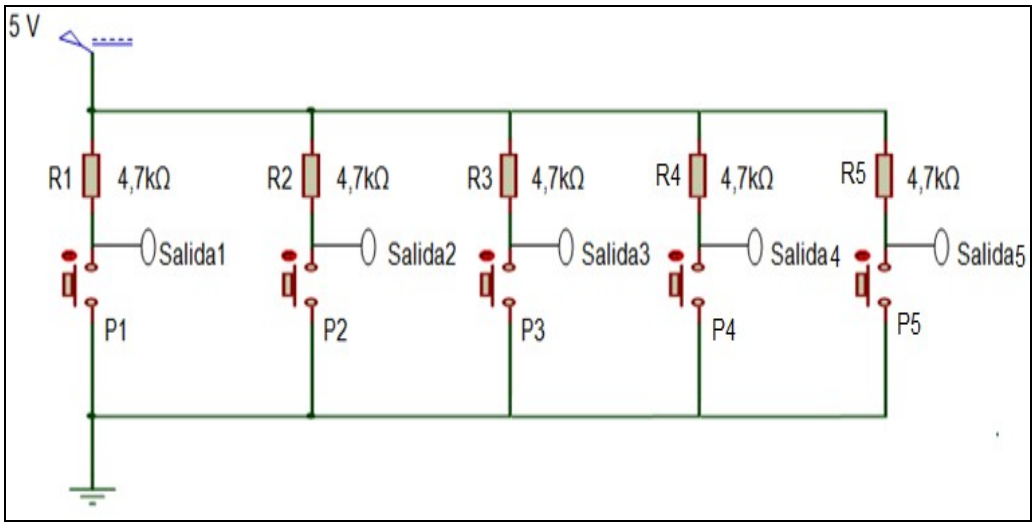


Figura 3.1: Circuito de pulsadores en el Entrenador Lógico.

25 Microcontroladores PIC Programación en Basic.

DIPSWITCH

El Entrenador Lógico también dispone de un Dip-switch de 8 posiciones el cual necesita de resistencias para controlar el paso de corriente a través de los pines del microcontrolador, los cálculos se lo realizan de a cuerdo a la ecuación 3.1 con los parámetros tomados en cuenta en el diseño del circuito para los pulsadores, ya que el Dip-switch es un grupo de interruptores que se asemeja a un pulsador normalmente cerrado. Entonces se tiene el diseño y los cálculos de la siguiente manera:

Voltaje de alimentación: 5VDC

Corriente de entrada por cada pin: 1mA

$$R = \frac{V}{I}$$

$$R = \frac{5\text{V}}{0{,}001\,A}$$

$$R = 5000\,\Omega$$

$$R = 5\ K\Omega$$

$$R \approx 4{,}7\ K\Omega$$

El valor de la resistencia establecida para cada posición del Dip-switch es de 4,7kΩ y el circuito es el que se muestra en la figura 3.2.

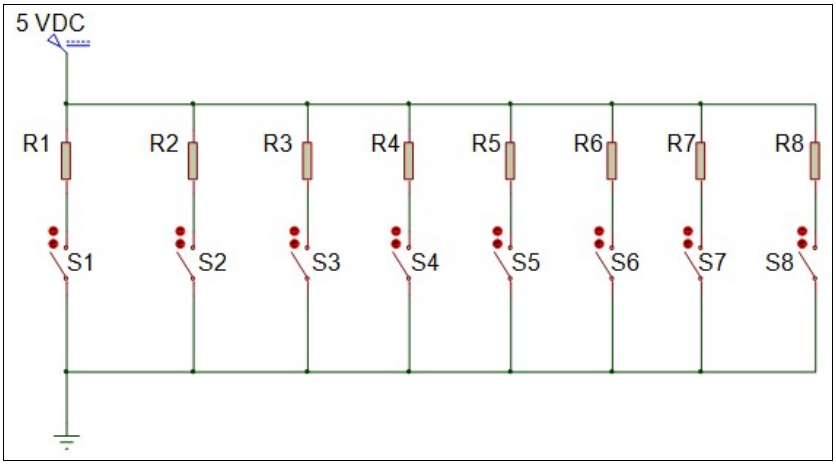


Figura 3.2: Circuito de Dip-switch de 8 posiciones.

TECLADO MATRICIAL

El teclado es matricial de 4x4 que quiere decir que es un arreglo de 16 pulsadores configurados en 4 filas y 4 columnas de acuerdo a la figura 3.3.

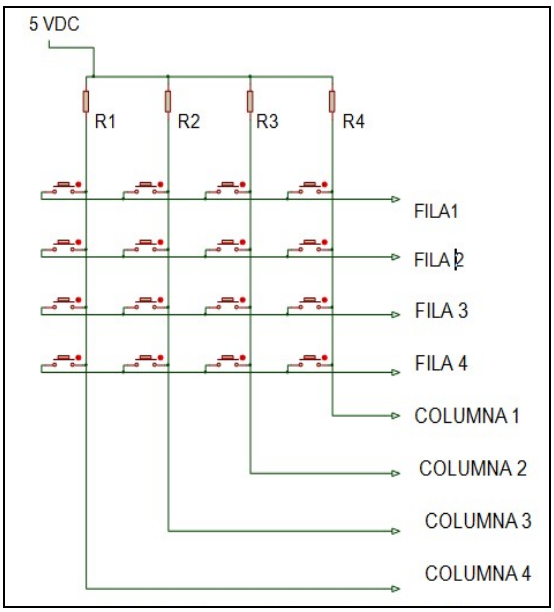


Figura 3.3: Estructura del teclado matricial de 4x4.

Durante la fase de lectura del teclado la mitad del puerto A ó B es configurada como entrada y la otra mitad como lectura y durante la escritura en el LCD u otro sistema, el puerto es configurado como salidas. Entonces se pueden cortocircuitar accidentalmente las salidas de los puertos provocando su destrucción, por esta razón se colocan las resistencias que actúan como un sistema de protección para el microcontrolador.

Si se pulsa alguna tecla se evita este hecho y así si se produjera el cortocircuito tan sólo circularía una pequeña corriente y el puerto del microcontrolador no correrá ningún riesgo.

El valor de las resistencias que se coloca como protección se lo determina de acuerdo a los cálculos realizados anteriormente para el diseño del circuito de los pulsadores.

Tomando en cuenta que se utiliza 5VDC y la corriente máxima que soporta el pin de un microcontrolador es de 25 mA, se determinó una resistencia de 200 Ω, pero lo

que se quiere obtener es un paso limitado de corriente para lo cual se utiliza la ecuación 3.1, para establecer el valor de la resistencia con 1mA, entonces se tiene el siguiente cálculo:

$$R = \frac{5\text{V}}{0{,}001\,A}$$

$$R = 5000\,\Omega$$

$$R = 5\ K\Omega$$

$$R \approx 4{,}7\ K\Omega$$

Entonces el valor de las resistencias R1, R2, R3 y R4 de la figura 3.3 es de 4,7 KΩ.

Para mostrar las teclas que han sido pulsadas ya sea en un display o en un LCD se necesita de un decodificador.

El decodificador lee la tecla que ha sido presionada y retorna un valor en código binario BCD. El diseño del mencionado circuito se muestra en la figura 3.4.

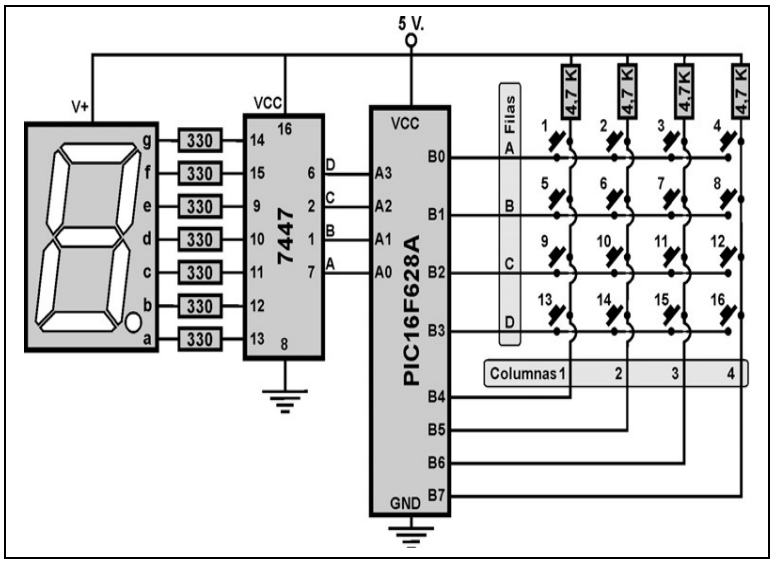


Figura 3.4: Diagrama de conexión de un teclado hexadecimal y un display de 7 segmentos.[26]

Para este caso se ha implementado el decodificador con un PIC 16F628A, además se ha implementado un decodificador de BCD a 7 segmentos para poder visualizar el código BCD en decimal a través del display de ánodo común.

26 Microcontroladores programación en basic

El circuito integrado que realiza esta conversión es el CI. 74LS47 tal como se muestra en la figura 3.4.

El teclado matricial implementado es muy útil a la hora de ingresar datos, un ejemplo es el teclado de una alarma que es parte de las 10 prácticas que se muestran más adelante. El funcionamiento del teclado matricial es el siguiente: siempre está censando las filas, cuando una de las filas detecta un nivel bajo, éste identifica el número de la columna que ha sido activada y almacena el valor dentro de una variable, para que esto ocurra, el PIC (16F628A) es programado de la siguiente manera:

Programación realizada en Micro Code Studio par el PIC 16F628A:

```
'*****************************************************************
'*  Name    : DECODIFICADOR.BAS                                  *
'*  Author  : JAVIER TERAN                                       *
'*  Notice  : Copyright (c) 2011 [select VIEW...EDITOR OPTIONS]  *
'*          : All Rights Reserved                                *
'*  Date    : 10/11/2011                                         *
'*  Version : 1.0                                                *
'*  Notes   :                                                    *
'*          :                                                    *
'*****************************************************************
cmcon=7             ;cambiar a modo digital todo el puerto A
TRISA=0              ;todo el puerto A como salidas

A     VAR PORTB.0     ;nombres para los pines de las filas
B     VAR PORTB.1
C     VAR PORTB.2
D     VAR PORTB.3

UNO      VAR PORTB.4     ;nombres para los pines de las columnas
DOS      VAR PORTB.5
TRES     VAR PORTB.6
CUATRO   VAR PORTB.7

GOTO TECLA
```

```
BARRIDO:
        LOW A                                       ;sensar la fila A
    IF UNO      = 0 THEN NUMERO =1 :RETURN   ;tecla pulsada retorne con
variable cargada con 1
    IF DOS      = 0 THEN NUMERO =2 :RETURN   ;tecla pulsada retorne con
variable cargada con 2
    IF TRES     = 0 THEN NUMERO =3 :RETURN   ;tecla pulsada retorne con
variable cargada con 3
    IF CUATRO   = 0 THEN NUMERO =4 :RETURN   ;tecla pulsada retorne con
variable cargada con 4
        HIGH A

        LOW B                                       ;sensar la fila B
    IF UNO      = 0 THEN NUMERO =5 :RETURN
    IF DOS      = 0 THEN NUMERO =6 :RETURN
    IF TRES     = 0 THEN NUMERO =7 :RETURN
    IF CUATRO   = 0 THEN NUMERO =8 :RETURN
        HIGH B

        LOW C                                       ;sensar la fila C
    IF UNO      = 0 THEN NUMERO =9 :RETURN
    IF DOS      = 0 THEN NUMERO =10:RETURN
    IF TRES     = 0 THEN NUMERO =11:RETURN
    IF CUATRO   = 0 THEN NUMERO =12 :RETURN
        HIGH C

         LOW D                                       ;sensar la fila D
     IF UNO      = 0 THEN NUMERO =13:RETURN
     IF DOS      = 0 THEN NUMERO =14:RETURN


    IF TRES     = 0 THEN NUMERO =15:RETURN
    IF CUATRO   = 0 THEN NUMERO =16:RETURN
        HIGH D
        PAUSE 10
GOTO BARRIDO

; ***************** programa de antirrebote de teclas *************
PTECLA:

    IF UNO      = 0 THEN PTECLA  ;si la tecla sigue pulsada ir espacio
    IF DOS      = 0 THEN PTECLA  ;si la tecla sigue pulsada ir espacio
    IF TRES     = 0 THEN PTECLA  ;si la tecla sigue pulsada ir espacio
    IF CUATRO   = 0 THEN PTECLA  ;si la tecla sigue pulsada ir espacio
    PAUSE 25
    RETURN                      ;retorna si se suelta las teclas

; *****************                          ************************

TECLA:
    GOSUB BARRIDO               ;ir a barrido y retornar con un valor
    GOSUB PTECLA        ;envía a un programa antirrebote para soltar tecla
GOTO TECLA
  END
```

POTENCIÓMETRO DIGITAL

El integrado MCP41100 es un potenciómetro digital, que puede controlar el volumen de audio, contraste de un LCD, fijar frecuencias en un VCO etc., usando dos pulsadores en lugar del común potenciómetro mecánico.
Este potenciómetro digital es más confiable que los de eje rotatorio porque no son afectados por el desgaste mecánico.

El diseño se lo realiza en base al circuito recomendado por la hoja de datos técnicos del circuito integrado de acuerdo al circuito de la figura 3.5.

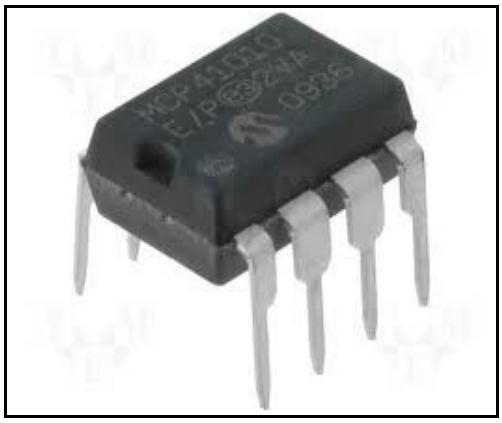


Figura 3.5: Potenciómetro digital.[27]

Si se conecta un capacitor a la fuente de corriente continua, uniendo uno de sus terminales al positivo y el otro al negativo, no hay circulación de electrones a través de él, debido a la presencia del dieléctrico, que como ya se sabe es un material aislante e impide que los electrones se desplacen a través de él, Para el circuito de la figura 3.6 el capacitor es de desacople para filtrar los posibles ruidos de la fuente de alimentación, por lo tanto debe estar lo más cerca posible del integrado.

El circuito se puede alimentar con una tensión de entre 5 VDC. y la corriente máxima que admite es de 1mA.

27 http://pdfserv.maxim-ic.com/en/ds/DS1669.pdf

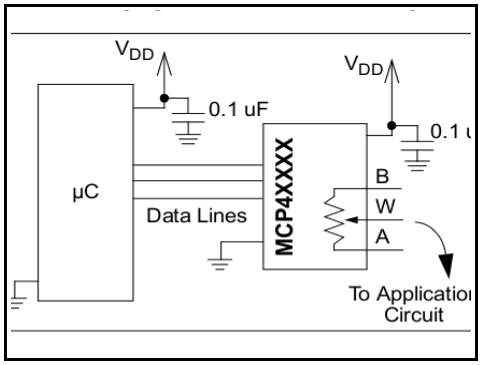


Figura 3.6: Potenciómetro digital con el MCP41100.[28]

3.3.2. Diseño Para Los Dispositivos de Salida

DIODOS LED

El circuito está compuesto por un grupo de 7 diodos led, se utiliza tres colores diferentes, cada color tiene una caída de voltaje diferente, para efecto de cálculos se toma en cuenta la corriente recomendada por los fabricantes de diodos led:

Tipo de diodo	Diferencia de potencial típica (voltios)
Rojo de bajo brillo	1.7 voltios
Rojo de alto brillo, alta eficiencia y baja corriente	1.9 voltios
Naranja y amarillo	2 voltios
Verde	2.1 voltios
Blanco brillante, verde brillante y azul	3.4 voltios
Azul brillante y LED especializados	4.6 voltios

Tabla 3.0: Caída de voltaje de los diodos LED.[29]

Se procede hacer los cálculos tomando en cuenta que el rango de corriente para el correcto funcionamiento de un diodo led es de 10 mA a 20mA, por lo que se toma una corriente promedio que es 15 mA, con la ecuación 3.1 se determina el valor de

28 http://jorgefloresvergaray.blogspot.com/2009/04/ds1669-potenciometro-digital.html
29 http://es.wikipedia.org/wiki/Circuito_de_LED

la resistencia a utilizar.

$$R = \frac{V}{I}$$

$$R = \frac{5\text{V}}{0{,}015\,A}$$

$$R = 333.33\,\Omega$$

$$R \approx 330\,\Omega$$

El valor de la resistencia, para que exista un consumo máximo de 15 mA es de 333,33Ω, pero se utiliza una resistencia aproximada existente en el mercado electrónico que es de 330Ω. Para el Entrenador Lógico se utiliza 7 diodos led y el circuito para esta etapa de dispositivos de salida resulta de la siguiente manera:

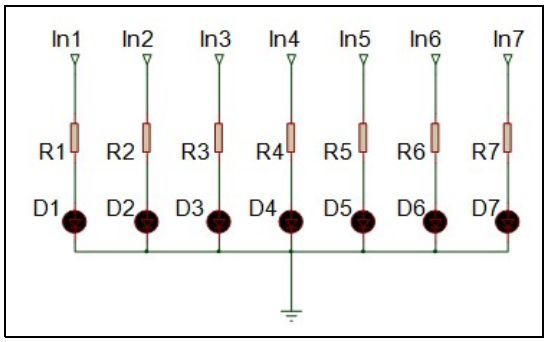


Figura 3.7: Diseño de circuito de 7 diodos led.

DISPLAY DE ANODO COMÚN

Para este circuito se incluyen las resistencias por cada segmento, ya que un display es un arreglo de diodos led, entonces el circuito para esta etapa es tal como muestra la figura 3.8.

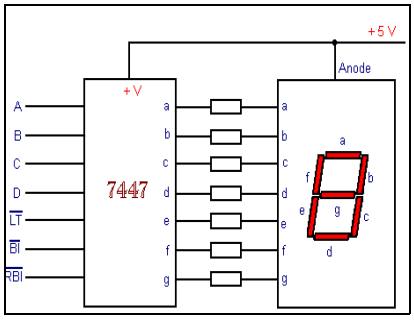


Figura 3.8: Diseño de circuito para controlar display de ánodo común.[30]

30 http://circuitosee.blogspot.com/2011/05/decodificador.html

El circuito integrado 7447 es un decodificador de BCD a siete segmentos (7 segmentos del display), que se encuentra implementado en el Entrenador Lógico como un sólo circuito.

Se implementan 4 display con transistores NPN 2N3906 en corte y saturación que actúan como interruptores automáticos permitiendo la activación o desactivación del display deseado, el circuito que se utiliza es el que se muestra en la figura 3.9.

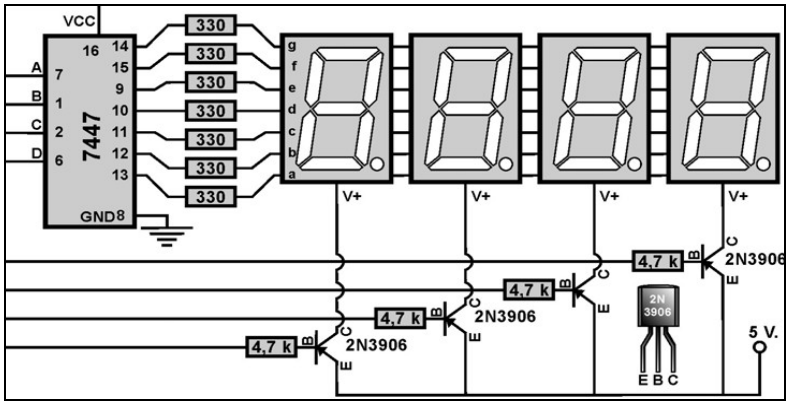


Figura 3.9: Circuito para manejar 4 display de siete segmentos con el C.I. 7447.[31]

El valor de las resistencias que se colocan es de 330Ω, cálculos que se lo realizó en el diseño del circuito para los diodos led, además se utiliza una fuente de 5 VDC.

MATRIZ DE LEDS

La matriz de diodos led con el que está diseñado el Entrenador Lógico es 8x8 (64 diodos led), de acuerdo a la figura 3.10.

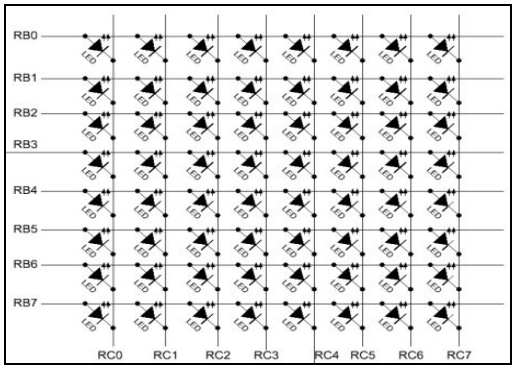


Figura 3.10: Diagrama de conexión de una matriz de diodos led de 8x8.[32]

Se utiliza 16 resistencias de 330Ω, 8 resistencias para las filas y 8 para las columnas,

31 Microcontroladores PIC programación en basic
32 http://www.neoteo.com/matriz-de-led-8x8

de acuerdo a la figura 3.9 las filas deben estar en un nivel bajo o cero lógico mientras que las columnas deben estar al contrario, es decir en nivel alto o uno lógico.

BUZER Y PARLANTE

Para poder realizar el diseño de este circuito se toma en cuenta el voltaje de funcionamiento del dispositivo y el voltaje que genera el microcontrolador para nivel alto. Este análisis es necesario ya que el buzer trabaja con un voltaje de 12VDC mientras que el nivel del voltaje que se obtiene de la salida de un pin del microcontrolador en nivel alto es tan solo de 5VDC, entonces el buzer no funcionará, por lo que se debe implementar un circuito de control adicional para el correcto funcionamiento.

El circuito es el que se muestra en la figura 3.11.

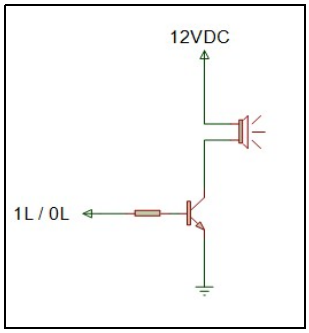


Figura 3.11: Circuito para buzer.

Para limitar el paso de corriente hacia el pin del microcontrolador se coloca una resistencia de 4,7 KΩ, ya que el pin soporta un máximo de 25 m A.

Para el circuito del parlante se implementa un control de volumen con un potenciómetro digital tal como muestra la figura 3.5, entonces el diseño será el que se muestra en la figura 3.12.

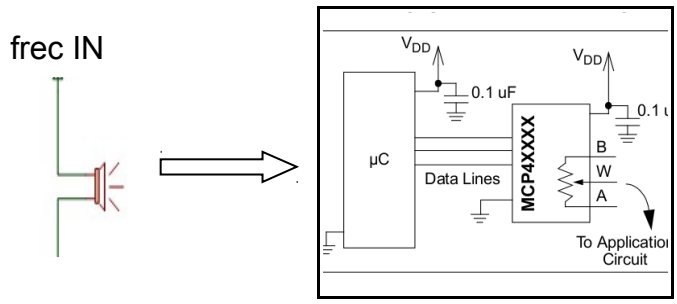


Figura 3.12: Diseño del circuito para el parlante con control de volumen.

MÓDULO LCD

Se implementa un módulo LCD de 2x16 con controlador Hitachi 44780 y back light, este dispositivo de salida se alimenta con 5VDC, para este circuito también se utiliza un potenciómetro para el ajuste del contraste del cristal líquido (0 VDC a 5 VDC).
El diseño para este circuito esta dado por la figura 3.13.

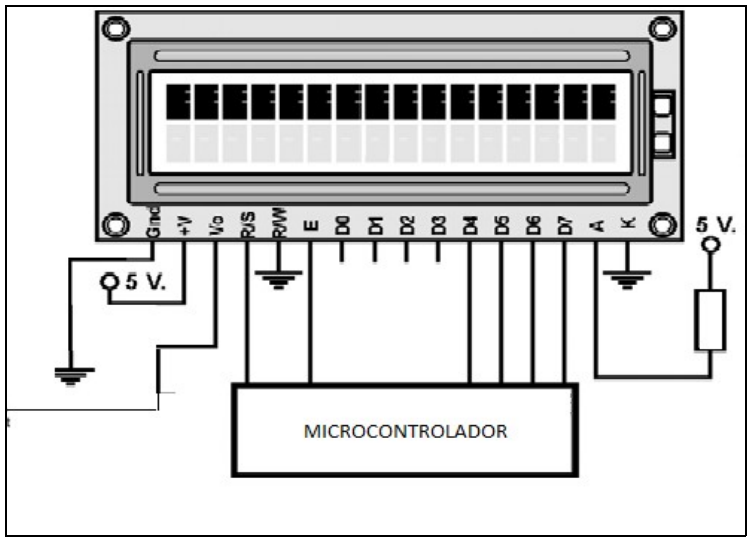


Figura 3.13: Diagrama de conexión de un módulo LCD con ajuste de contraste.

RELÉ

El Entrenador Lógico está compuesto por tres relés, estos están conectados con un circuito de control adicional ya que el voltaje de cada pin de un microcontrolador en nivel alto ó uno lógico es de 5VDC mientras que el relé se alimenta con 12VDC, el circuito se muestra en la figura 3.14.

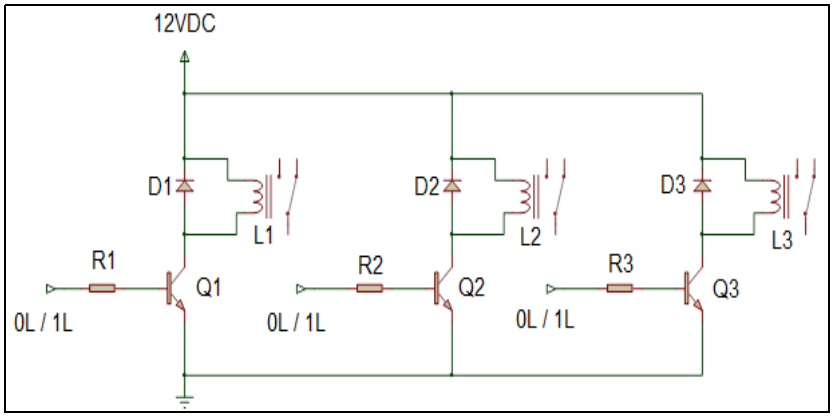


Figura 3.14: Circuito para utilizar tres relés.

En la figura 3.14 se utiliza tres diodos rectificadores, estos para impedir el paso de corriente en dirección contraria en caso de ocurrir un cortocircuito, las resistencias son de 4,7KΩ, los transistores Q1, Q2 y Q3 son 2N3904 y al circuito se alimenta con una fuente de 12VDC.

3.3.3. Quemador de Pic

Se utilizará el quemador de PICs de la serie 16Fxxx y para AVRs.

3.3.4. Diseño De La Etapa De Comunicación Inalámbrica

En el Entrenador Lógico se constituye una etapa de transmisión y recepción inalámbrica, se la implementa con módulos de RF de 433Mhz y módulos de 2,4Ghz.

Para la comunicación por RF de 434Mhz se utiliza los siguientes elementos:

TRANSMISOR TLP434 Y RECEPTOR RLP434 INALAMBRICO

El transmisor TLP434, y receptor RLP434 permiten comunicación inalámbrica a una velocidad de 9600 bps (baudios por segundo) de un punto a otro evitando la tediosa comunicación por medio de un cableado y más aun cuando estos puntos están lejanos.

Mencionados módulos en un trabajo conjunto entablan comunicación hasta de 160m en línea de vista y 30m en interiores. Entre las aplicaciones que se pueden realizar con el Entrenador Lógico y las más usadas por los módulos de radiofrecuencia, receptor y transmisor son:

Sistemas de seguridad inalámbrica, alarmas de seguridad e incendios, teledetección, comunicación de datos, sistema de buscapersonas, sistemas de puertas (libre de llaves), apertura de garajes y compuertas, control de luminosidad, sistemas de monitoreo médico, sistemas de llamado, transferencia periódica de datos, automatización de uso industrial y residencial.

TLP434A.- Módulo transmisor que emite señales de radiofrecuencia a una frecuencia fija de 433 MHz, la forma de transmitir es modulación por desplazamiento de amplitud (ASK), de esta forma los datos digitales se transmiten por amplitud, este módulo de un tamaño de 13.3x10.3mm, el circuito es el que se muestra en la figura 3.15.

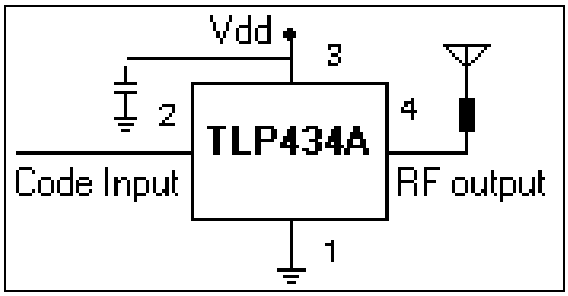


Figura 3.15: Circuito transmisor utilizando el TLP434A.

Las características principales del módulo de transmisión TLP434A se presentan en la tabla 3.1.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Vcc	Operating supply voltage		2.0	-	12.0	V
Icc 1	Peak Current (2V)		-	-	1.64	mA
Icc 2	Peak Current (12V)		-	-	19.4	mA
Vh	Input High Voltage	Idata= 100uA (High) Vcc-0.5		Vcc	Vcc+0.5	V
Vl	Input Low Voltage	Idata= 0 uA (Low)	-	-	0.3	V
FO	Absolute Frequency	315Mhz module	314.8	315	315.2	MHz
PO	RF Output Power- 50ohm	Vcc = 9V-12V	-	16	-	dBm
		Vcc = 5V-6V	-	14	-	dBm
DR	Data Rate	External Encoding	512	4.8K	200K	bps

Notes : (Case Temperature = 25°C +- 2°C , Test Load Impedance = 50 ohm)

Tabla 3.1: Características principales del módulo de transmisión TLP434A.

RLP434A.- Módulo receptor, que permite recibir información a una frecuencia que puede ser ajustada, la forma de receptar es demodulación por desplazamiento de amplitud (ASK), de esta forma los datos digitales se receptan por amplitud. Tiene 8 pines, y el tamaño de 43.42x11.5mm. , el circuito es el que se muestra en la figura 3.16.

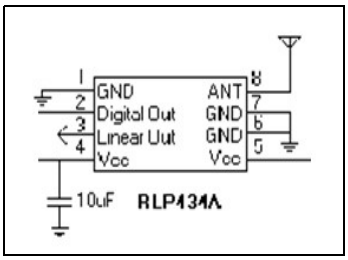


Figura 3.16: Circuito receptor utilizando el RLP434A.

Las características principales del módulo de transmisión RLP434A se presentan en la tabla 3.2.

Symbol	Parameter	Conditions	Min	Typ	Max	
Vcc	Operating supply voltage		3.3	5.0V	6.0	V
Itot	Operating Current		-	4.5		mA
Vdata	Data Out	Idata = +200 uA (High)	Vcc-0.5	-	Vcc	V
		Idata = -10 uA (Low)	-	-	0.3	V

Electrical Characteristics					
Characteristics	SYM	Min	Typ	Max	Unit
Operation Radio Frequency	FC	315, 418 and 433.92			MHz
Sensitivity	Pref		-110		dBm
Channel Width			+-500		Khz
Noise Equivalent BW			4		Khz
Receiver Turn On Time			5		ms
Operation Temperature	Top	-20	-	80	C
Baseboard Data Rate			4.8		KHz

Tabla 3.2: Características generales y eléctricas del módulo de recepción RLP434A.

MÓDULO BLUETOOTH

El módulo de desarrollo bluetooth de Microingeniería S.L. que se utiliza en el Entrenador Lógico es el RN41 de Roving Networks, es un perfecto dispositivo para eliminar los cables en los proyectos que se realizan con las guías de prácticas mostradas más adelante, para conectarse a ordenadores o teléfonos móviles. Con un alcance de hasta 100 metros (Clase I), posee antena integrada, es compatible con el estándar bluetooth 2.1, permite velocidades de transferencia de hasta 921Kbps (kilo bytes por segundo), y se lo puede conectar de forma sencilla mediante la UART (RX/TX) de cualquier microcontrolador, desde donde se puede controlarlo haciendo uso de sencillos comandos AT. El módulo, además, puede ser alimentado tanto a 5VDC como a 3,3VDC, y para ello dispone de un puente (jumper) de selección de la tensión de alimentación. El RN41 funciona a 3,3VDC y sus líneas no son 5VDC compatibles, por lo que las líneas de uso normal (Tx/Rx) están provistas de adaptación de niveles para su utilización con las placas de desarrollo

que operan a 5VDC. De todos modos, posee entradas auxiliares que permiten trabajar en forma directa con el RN41 y un bus Tx/Rx de 3,3VDC. El diagrama del circuito se lo presenta en la figura 3.17.

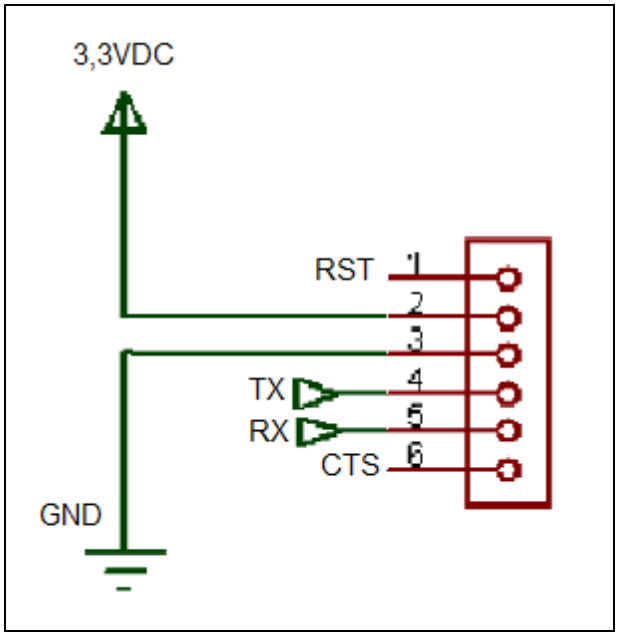


Figura 3.17: Diagrama de conexión del módulo Bluetooth.

DISEÑO DEL MINI ROBOT

El mini robot está compuesto por dos ruedas traseras y una delantera las mismas que permiten que se dirija en cuatro direcciones principales que son: derecha, izquierda, adelante y atrás. Las llantas traseras están provistas de un motor con el cual se puede variar la velocidad.

Además consta de un detector de obstáculos a través de ultrasonido.

El mini robot dispone de un módulo de radiofrecuencia RLP434A y un módulo bluetooth para la comunicación y control inalámbrico, se alimenta con una batería de 9VDC pero como el circuito y los PIC se deben alimentar con una fuente de 5VDC se utiliza el regulador de voltaje LM2575-5, se utiliza este regulador ya que es más eficiente que el 7805 porque disipa menos energía en forma de calor, gracias a ésto la batería dura más tiempo.

3.3.5. Diseño De La Fuente De Voltaje

La fuente de voltaje se la realiza de acuerdo a los requerimientos de voltaje mínimo y máximo de cada circuito que se ha diseñado para el Entrenador Lógico.

La fuente de voltaje tiene una alimentación de 120 VAC, pasa por la etapa de

transformación que reduce el voltaje, luego se rectifica y filtra la señal para obtener voltaje continua, luego de estas etapas se coloca los reguladores de voltaje que son: LM7812, LM7805 y LD1117 estos reguladores de voltaje son de 12VDC, 5VDC y 3,3VDC respectivamente.

La figura 3.18 muestra el diagrama de bloques de la fuente de voltaje que se usa en el Entrenador Lógico.

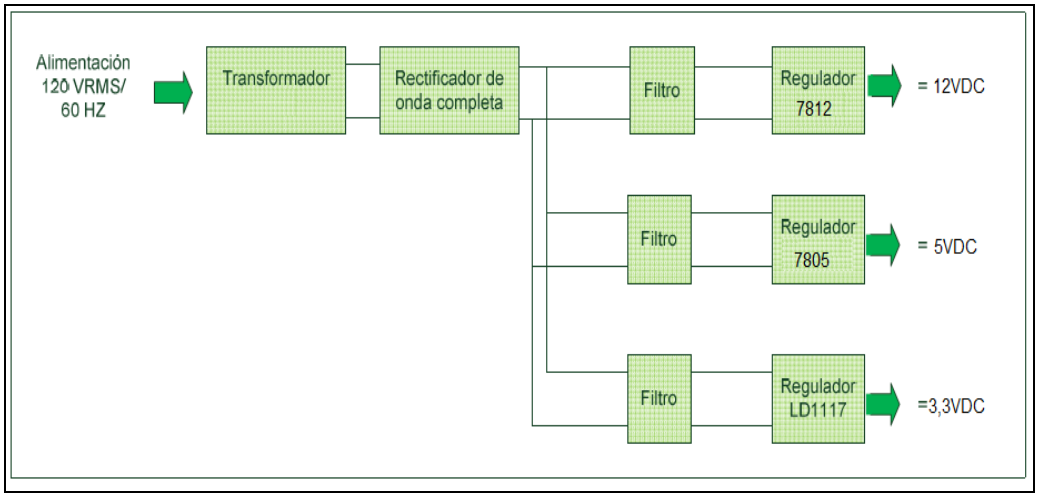


Figura 3.18: Diagrama de bloques de la fuente de voltaje.

En esta fuente de voltaje tiene tres salidas, la primera es de 12 VDC que proporciona el circuito integrado 7812, que se utiliza para alimentar los circuitos que utilizan los relés, el módulo de transmisión TLP434A, el circuito para el parlante y el circuito para el buzer.

Los 5VDC que proporciona el circuito integrado 7805 sirven para alimentar los circuitos integrados, diodos led, matriz de diodos led, pulsadores, Dip-switch, display y teclado matricial.

El circuito integrado LD1117 proporciona un voltaje de salida de 3,3VDC que sirven para alimentar el módulo bluethooth, voltaje que ha sido establecido por el fabricante y detallado en la hoja de datos técnicos del mismo.

Las salidas de voltaje son totalmente independientes entre sí, ya que para cada salida se tiene un regulador con el voltaje deseado para cada etapa del Entrenador Lógico

El circuito de la fuente de voltaje para el Entrenador Lógico es el que se muestra en la figura 3.19.

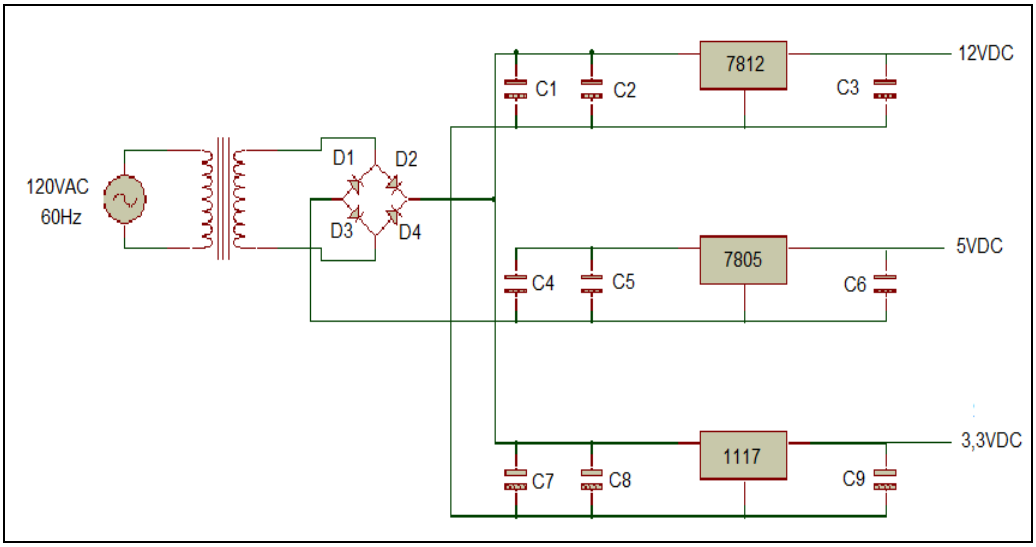


Figura 3.19: Diseño del circuito de la fuente voltaje.

3.4. Implementación

La implementación se la realiza en el mismo orden que se realizó el diseño de cada etapa y tomando en cuenta la disposición de los elementos para luego realizar el montaje.

Los Principales materiales utilizados en la implementación del Entrenador Lógico son: Multímetro y cables de cobre para los circuitos pequeños, mientras que para los circuitos que incluyen PIC´s se utiliza recursos informáticos con el hardware y software necesarios para el correcto uso del mismo, el sistema operativo usado es Windows XP para el quemador de PIC´s mientras que, para realizar la programación y la simulación se utiliza Linux (Ubuntu 10.04), además se maneja una fuente de voltaje dependiendo del diseño de cada circuito.

PULSADORES

De acuerdo al diseño para la implementación de los pulsadores y tomando en cuenta el circuito de la figura 3.1. (Circuito de pulsadores en el Entrenador Lógico) se tiene que implementar el circuito como muestra la figura 3.20.

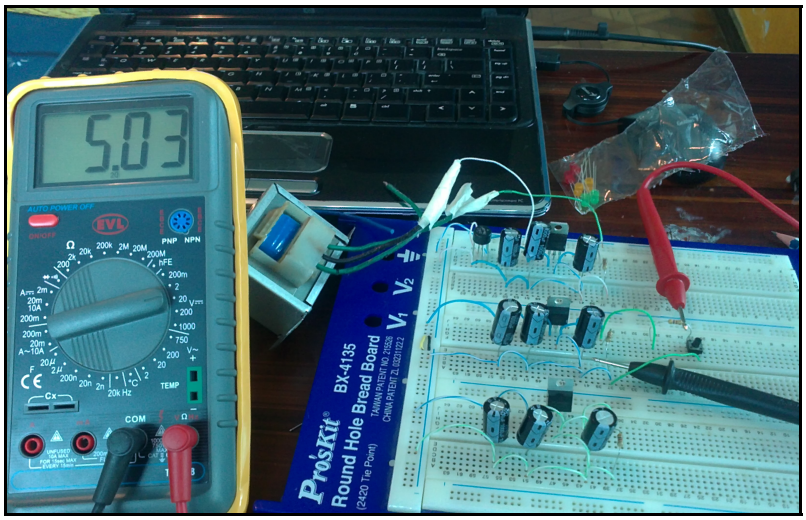


Figura 3.20. Implementación del circuito de pulsadores.

A continuación se presenta la tabla 3.3 que muestra los niveles de voltaje y corriente del circuito de pulsadores que se obtiene al presionarlos.

PULSADOR		VOLTAJE	CORRIENTE
SI	NO		
	X	5,03VDC	1,07mA
X		0,00VDC	0,00mA

Tabla 3.3: Niveles de voltaje y corriente del circuito de pulsadores.

DIP-SWITCH

Para la implementación del Dip-switch se utiliza uno de 8 posiciones y para cada pin se trabaja con una resistencia de 4.7KΩ tal como muestra la figura 3.21.

Figura 3.21: Implementación del circuito del Dip-switch.

Para este caso los niveles de voltaje son los mismos pero la diferencia es que en el Dip-switch el voltaje permanece constante y no como en el caso anterior que el nivel de voltaje dura sólo el tiempo de pulsación.

La tabla 3.4 muestra los niveles de voltaje y corriente que se obtiene cuando se abre o se cierra un circuito con el Dip-switch.

DIP-SWITCH		VOLTAJE	CORRIENTE
abierto	cerrado		
	X	5,03VDC	1,07mA
X		0,00VDC	0,00mA

Tabla 3.4: Niveles de voltaje y corriente del circuito del Dip-switch.

TECLADO MATRICIAL

Para el circuito del teclado matricial se implementa el decodificador de las teclas con un PIC 16F628A, además se coloca un decodificador de BCD a 7 segmentos el cual permite visualizar el código BCD en decimal a través del display de ánodo común, el circuito integrado que realiza esta conversión es el 74LS47.

El circuito implementado es el de la figura 3.4, se lo puede utilizar como un teclado independiente del PIC y del 74LS47.

El circuito del teclado matricial es el que se muestra en la figura 3.22.

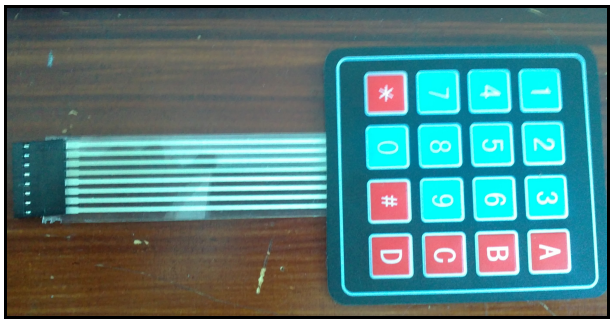


Figura 3.22: Teclado matricial para el Entrenador Lógico.

POTENCIOMETRO DIGITAL

El Entrenador Lógico dispone de un potenciómetro digital el PCM41100 que se lo puede controlar a través de un microcontrolador este circuito integrado es de 8 pines y se alimenta con un voltaje de 5VDC. La implementación del C.I. MCP41100 es la

que se muestra en la figura 3.23.

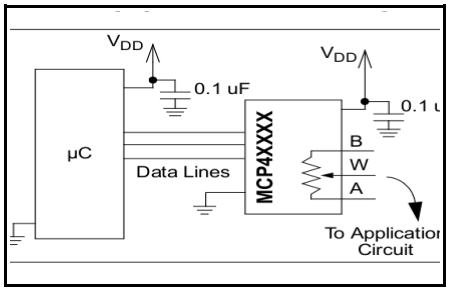


Figura 3.23: Potenciómetro digital para el Entrenador Lógico.

DIODOS LED

El circuito de los diodos led se desarrolla de acuerdo al circuito de la figura 3.7 y su implementación es tal como muestra la figura 3.24.

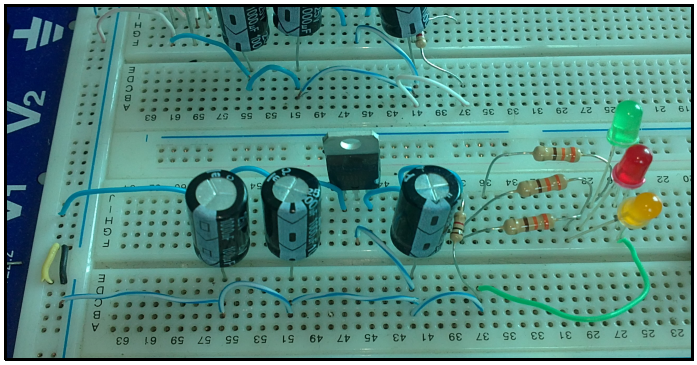


Figura 3.24: Implementación del circuito de diodos led para el Entrenador Lógico.

El circuito esta implementado con resistencias de 330Ω como protección para los diodos led, el voltaje recomendado para el led en serie con la resistencia es de 5VDC, voltaje máximo que genera un PIC.

La tabla 3.5 muestra los niveles de voltaje y corriente utilizando el diseño de la fuente del Entrenador Lógico y con una carga de 330Ω.

LED	VOLTAE	CORRIENTE
AMARILLO	1,92 VDC	8,95mA
VERDE	2,03 VDC	8,61mA
ROJO	2,05 VDC	8,60mA

Tabla 3.5: Niveles de voltaje y corriente.

DISPLAY DE ÁNODO COMÚN

En esta parte del circuito se tienen conectados 4 display de ánodo común en cascada los cuales son controlados por los transistores 2N3906 y un decodificador de BCD a 7 segmentos, las aplicaciones para este circuito dependen de los requerimientos de las guías de prácticas.

La figura 3.25 muestra la implementación del circuito de cuatro display en cascada.

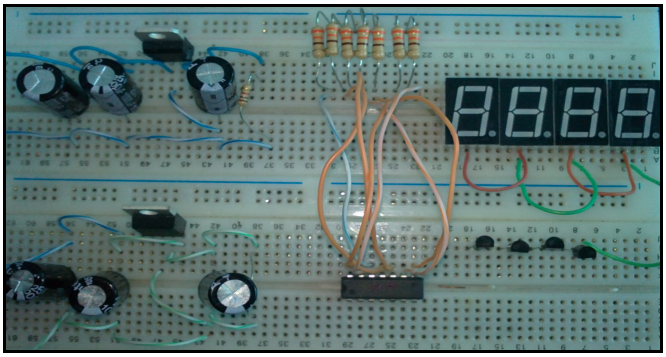


Figura 3.25: Implementación del circuito de 4 display en cascada.

El circuito de la figura 3.25 se lo alimenta con una fuente de 5VDC, las resistencias son de 330Ω que se utiliza como limitador de corriente para los segmentos de los display. Se puede utilizar los display por separado si así se lo requiere.

MATRIZ DE LEDS

El circuito está compuesto de una matriz de diodos led de 8 filas por 8 columnas utiliza 8 transistores 2N3904 para realizar el control de la misma.

La figura 3.26 muestra el circuito implementado de la matriz de diodos led de 8x8.

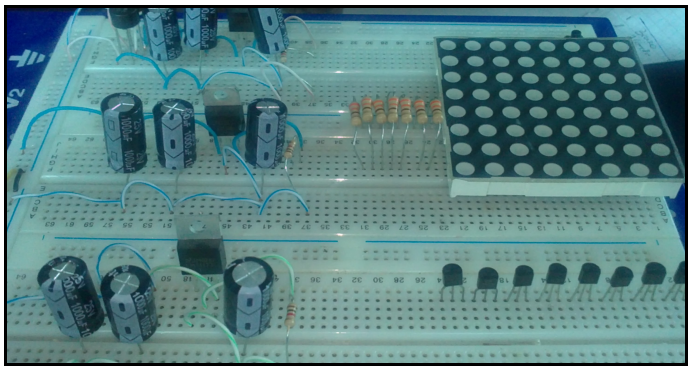


Figura 3.26: Implementación de la matriz de diodos led para el Entrenador Lógico.

El circuito funciona con una alimentación de 5VDC, las resistencias de 330Ω son de protección para las filas mientras que los transistores tienen una resistencia en su base de 4,7KΩ; estos elementos, trabajan como circuitos de control para activar o desactivar las columnas de la matriz de diodos led.

PARLANTE

El Entrenador Lógico posee de un parlante pequeño de 8Ω de 0,25W que se utiliza como un elemento indicador, el mismo sirve para simular una sirena de una alarma como parte de las prácticas expuestas más adelante.

Figura 3.27: Parlante para el Entrenador Lógico.

MÓDULO LCD

La configuración del módulo LCD está dada por la figura 3.13 y la implementación se la realiza con 2 potenciómetros, uno para el control del contraste y otro para la alimentación de backlight.

La implementación del módulo LCD es la que se muestra en la figura 3.28.

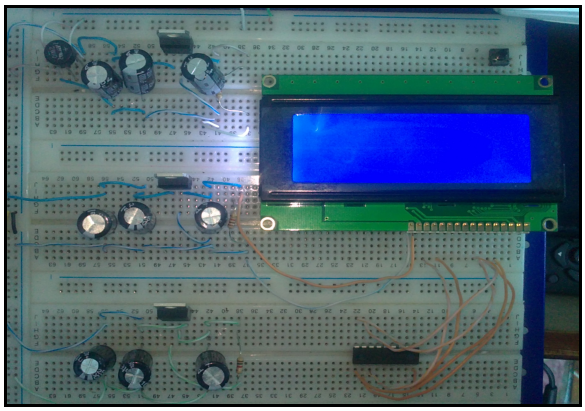


Figura 3.28: Implementación del módulo LCD para el Entrenador Lógico.

El circuito de la figura 3.28 se alimenta con un voltaje de 5 VDC y la programación está proporcionada por el PIC que se utilice y de los datos que se quieran mostrar en pantalla.

RELÉS

La implementación de este circuito se la efectúa de acuerdo al diseño de la figura 3.14, como es un circuito de control se utiliza transistores 2N3904 que actúan como interruptores automáticos, energizando la bobina la cual genera un campo magnético haciendo que se cierre el circuito a controlar.

Para esta etapa se implementa 3 relés con los cuales se pueden realizar las prácticas en el Entrenador Lógico.

La implementación es la que se muestra en la figura 3.29.

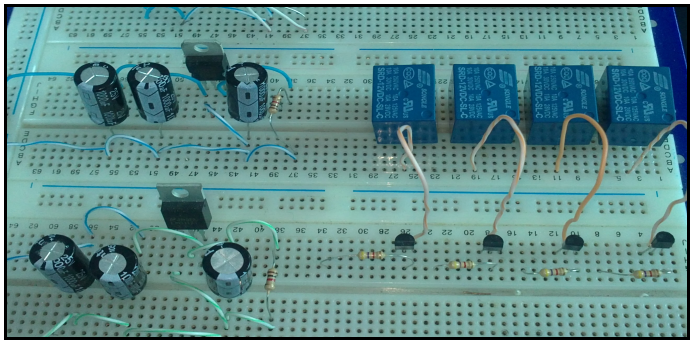


Figura 3.29: Implementación de relés para el Entrenador Lógico.

El circuito de la figura 3.29 se alimenta con un voltaje de 12VDC, las resistencias de 4,7KΩ, se utiliza como un limitador de corriente para el transistor, el cual funciona como un interruptor automático permitiendo que se energice la bobina cuando se presente un uno lógico a la entrada del circuito de la figura 3.29.

MÓDULO DE TRANSMISIÓN TLP434 Y RECEPCIÓN RLP434

La implementación del módulo de transmisión y de recepción se la ejecuta de acuerdo al diseño establecido en las figuras 3.15 y 3.16 respectivamente, los módulos se alimentan con el voltaje establecido en su hoja de datos técnicos.

La implementación del módulo de transmisión se la realiza de acuerdo a la figura 3.30.

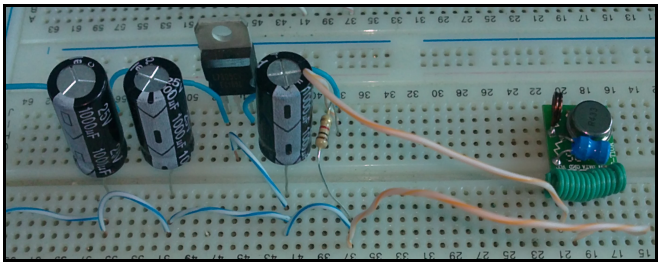


Figura 3.30: Implementación del módulo de transmisión TLP434 para el Entrenador Lógico.

El módulo de transmisión TLP434 está alimentado con una fuente de voltaje de 12VDC, voltaje que esta dado por la hoja de datos técnicos del módulo.

El módulo de recepción se alimenta con un voltaje de 5VDC. La implementación del módulo de recepción es tal como muestra la figura 3.31.

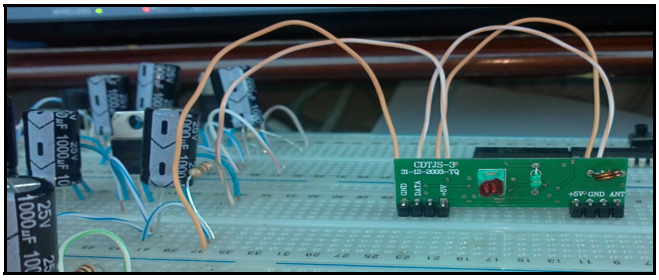


Figura 3.31: Implementación del módulo de recepción RLP434 para el Entrenador Lógico.

MÓDULO BLUETOOTH

El módulo bluetooth está alimentado con un voltaje de 3.3 VDC se lo implementa de acuerdo a la figura 3.32.

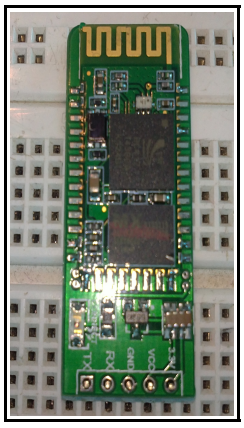


Figura 3.32: Implementación del módulo bluetooth para el Entrenador Lógico.

MINI ROBOT

El mini robot está diseñado de acuerdo a la figura 3.33, y posee los módulos de transmisión, recepción (TLP434 y RLP434), el módulo bluetooth y un zócalo de 18 pines para el PIC 16F628A para programar de acuerdo a las guías de prácticas, el mismo que dispone de un regulador de voltaje de bajo consumo que hace que la batería externa dure mayor tiempo.

La implementación del mini robot es la que se muestra en la figura 3.33.

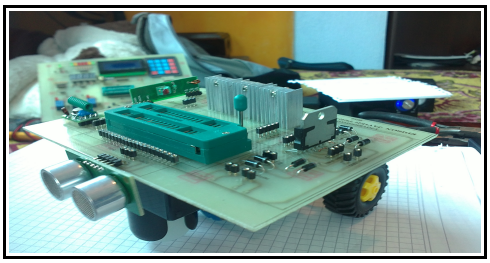


Figura 3.33: Implementación del mini robot para el entrenador Lógico.

FUENTE DE VOLTAJE

La implementación de la fuente de voltaje se realiza de acuerdo a los voltajes que necesita el Entrenador Lógico, estos voltajes son de 3.3VDC, 5VDC Y 12VDC.

Se diseñan tres salidas con los voltajes mencionados, estos voltajes son independientes el uno del otro ya que para cada salida de voltaje se utiliza un regulador diferente.

La implementación de la fuente de voltaje es la que se muestra en la figura 3.34 ((a) 3.3VDC (b) 5VDC (c) 12VDC) y se la realiza de acuerdo al diseño de la fuente de voltaje de la figura 3.19.

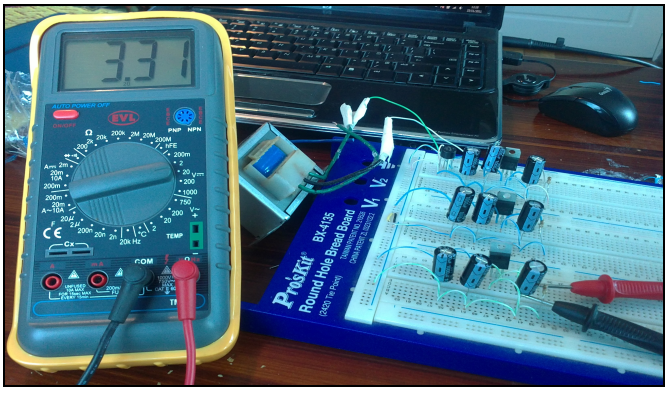


Figura 3.34: (a) Fuente de voltaje de 3.3VDC para el Entrenador Lógico .

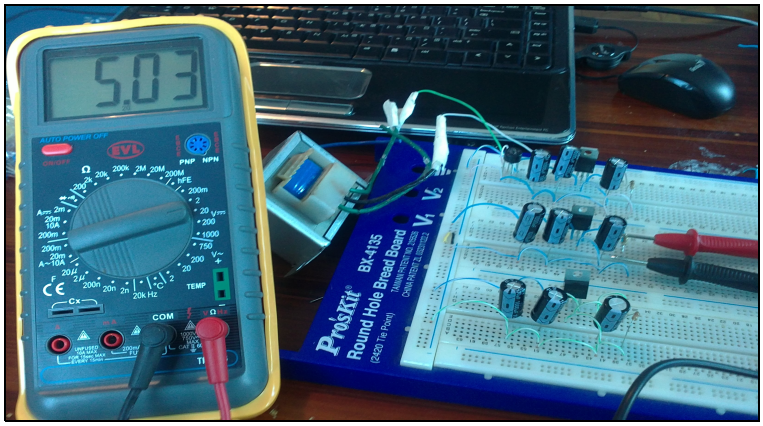


Figura 3.34: (b) Fuente de voltaje de 5 VDC para el Entrenador Lógico.

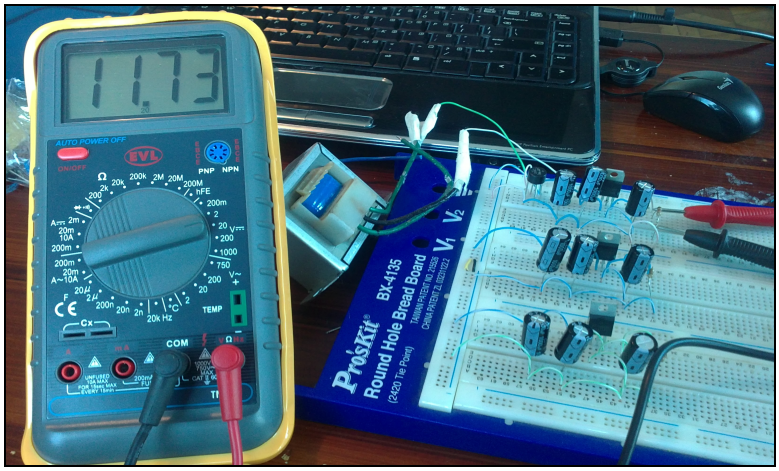


Figura 3.34: (c) Fuente de voltaje de 12 VDC para el Entrenador Lógico.

3.5. Montaje Del Entrenador Lógico

Una vez realizado el diseño y la implementación se realizó el montaje de del Entrenador Lógico.

A continuación se describe el proceso de montaje del Entrenador Lógico:

- Una vez terminado el diseño se elije el tipo de tarjeta o placa que se utilizará, en este caso se ha realizado el montaje del Entrenador Lógico en una placa de fibra de vidrio.
- Las pistas se realizaron en el software denominado Proteus 7, dentro del mismo software tenemos dos aplicaciones ISIS y ARES, para este caso se utilizó ARES. Se realizó el diseño de 4 placas de fibra de vidrio; dos son del Entrenador Lógico, una para el mini robot y una para la fuente de alimentación del Entrenador Lógico. A continuación se muestra en la figura 3.35 el diseño de las pistas realizadas en Proteus 7.

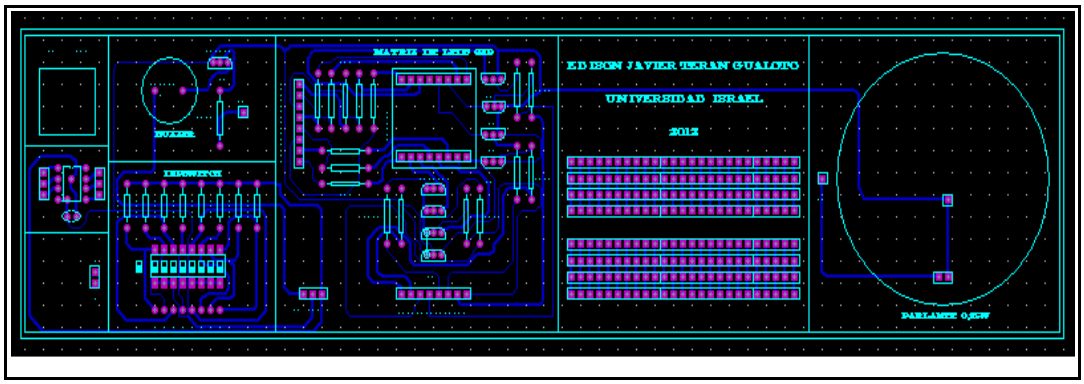


Figura 3.35: (a) Entrenador Lógico 1.

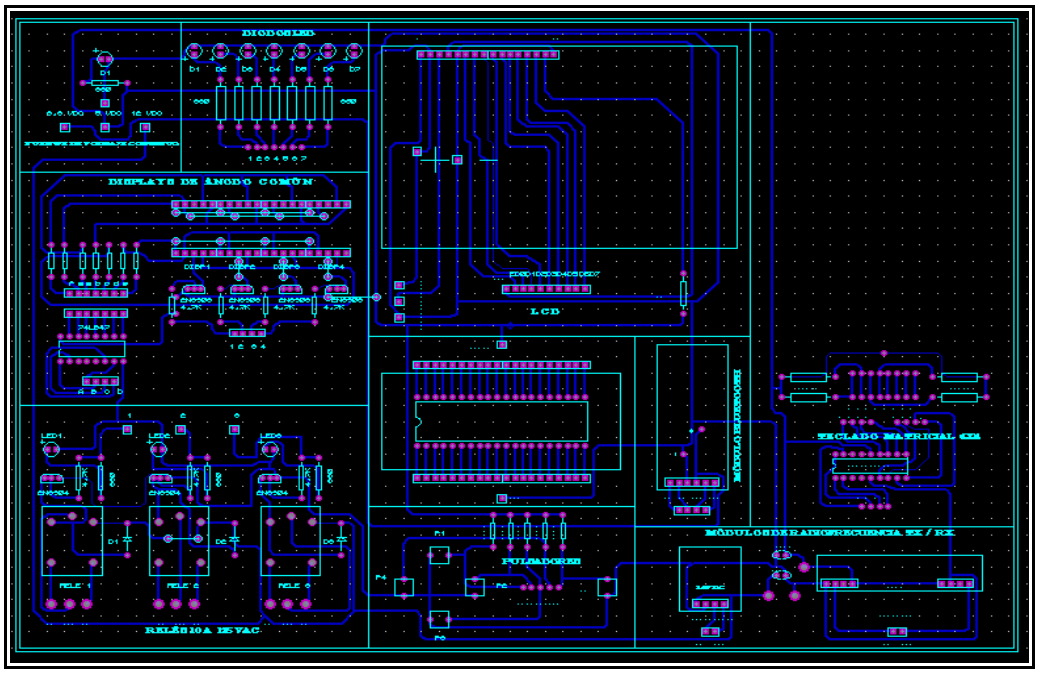


Figura 3.35:(b) Entrenador Lógico 2.

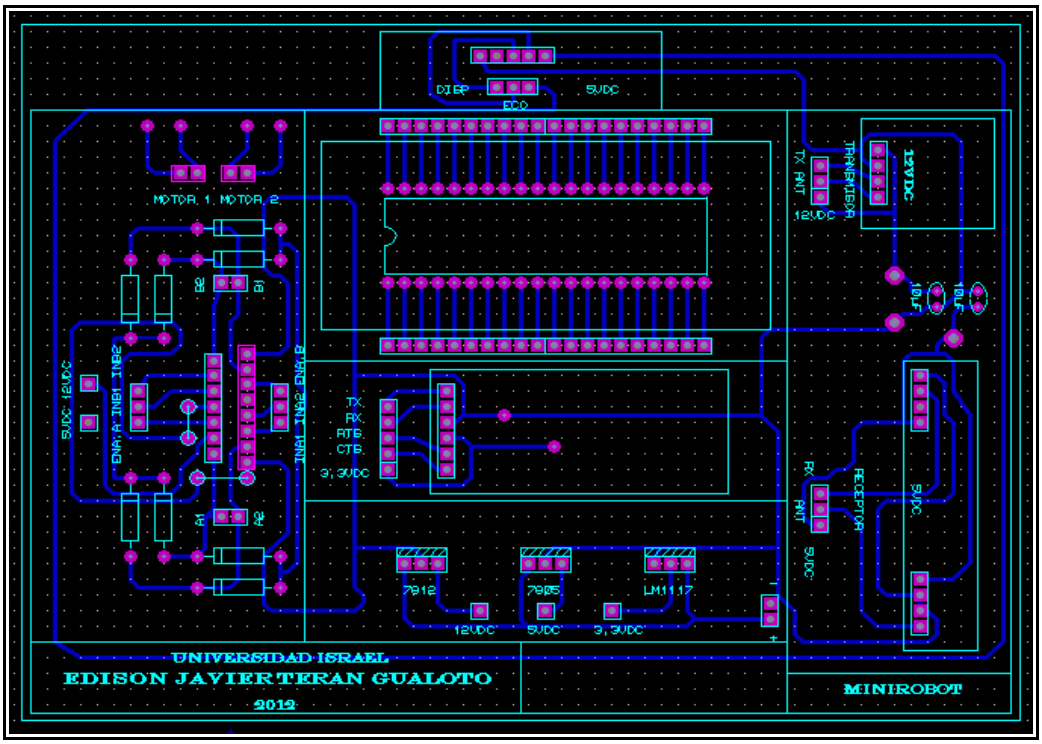


Figura 3.35: (c) Mini Robot.

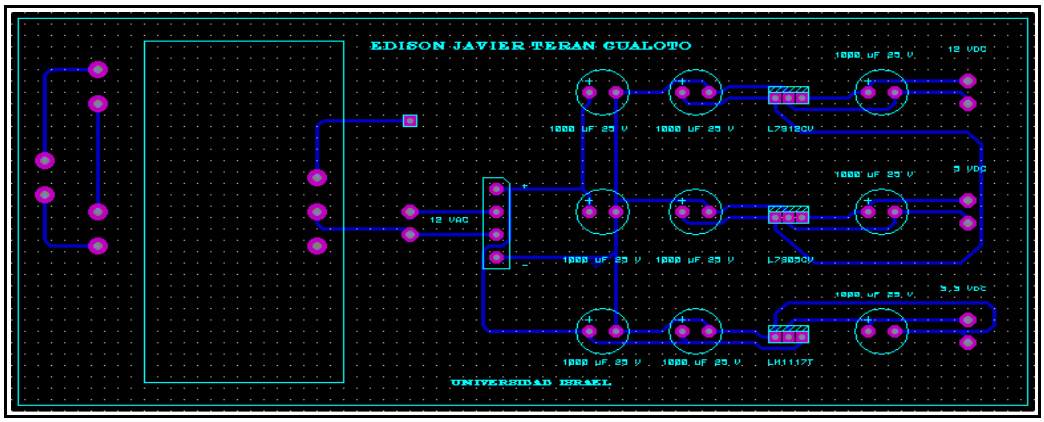


Figura 3.35: (d) Fuente de Voltaje para el Entrenador Lógico.

- Luego de haber realizado el diseño de las pistas se procede a quemar el cobre innecesario de la misma.
- Se realiza los huecos con un taladro pequeño para poder montar los elementos en las placas.
- Para soldar los elementos en la placa se utiliza alambre de estaño, pasta de

soldadura y cautín. La figura 3.36 muestra los elementos para el montaje en la placa de fibra de vidrio utilizada para la fuente de voltaje para el Entrenador Lógico.

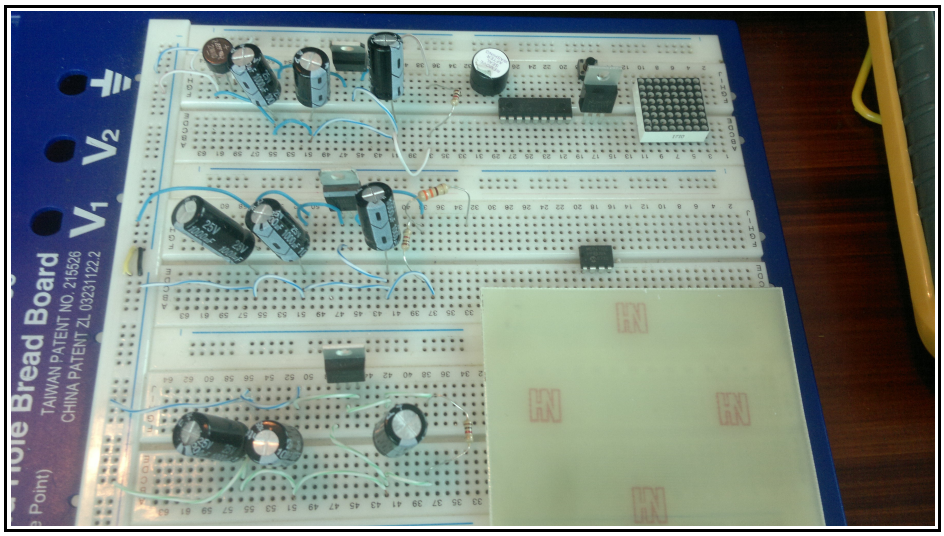


Figura 3.36: (a) Placa de fibra de vidrio y elementos de la fuente de voltaje.

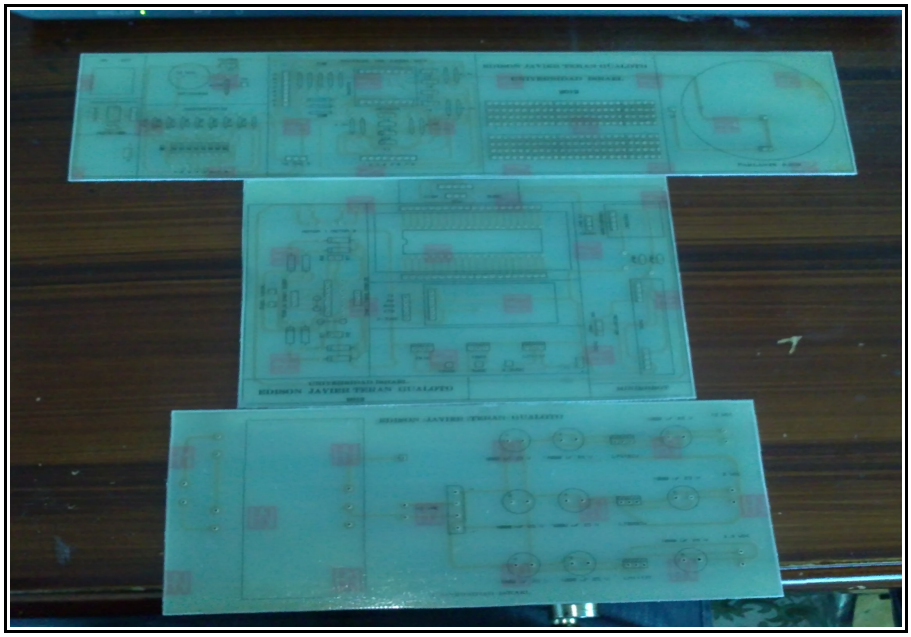


Figura 3.36: (b) Placas con huecos para montar los elementos.

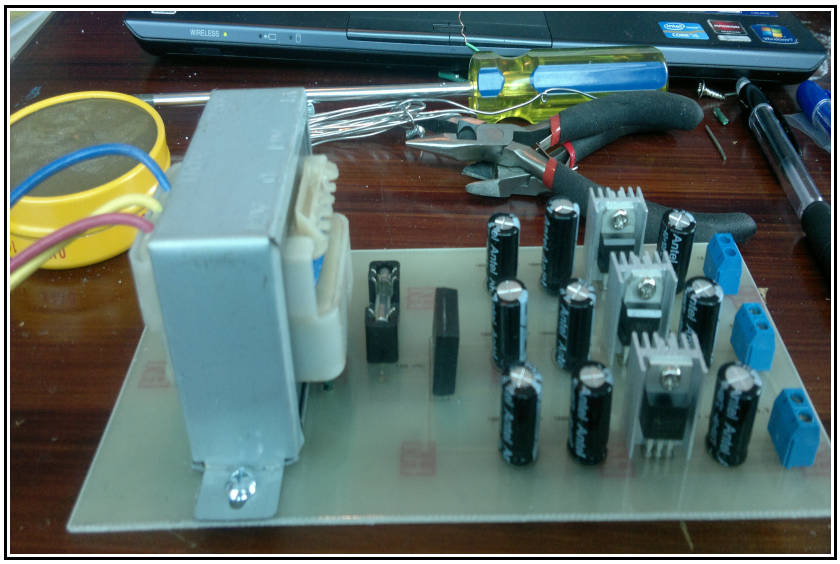


Figura 3.36: (c) Montaje final de la Fuente de Voltaje.

Una vez explicado el proceso de montaje de la fuente de VDC se realizó el montaje de los elementos en la placa del Entrenador Lógico y en la placa del Mini Robot. En La figura 3.37 se presenta el montaje final del Entrenador Lógico, Mini Robot y la fuente de VDC.

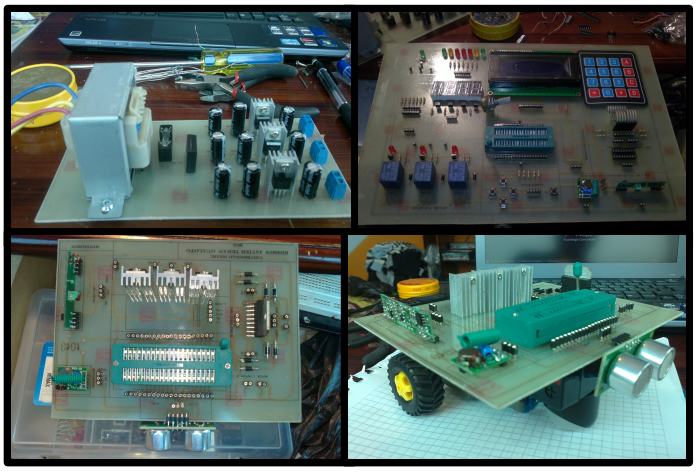


Figura 3.37: Placas finales del Entrenador Lógico.

3.6. Diseño del chasis del Entrenador Lógico

Una vez terminado el proceso de montaje se realizó el diseño del chasis del Entrenador Lógico y del Mini Robot, diseño que se lo realizó de acuerdo a las medidas de las placas.

El diseño y las medidas del chasis están dadas de acuerdo a la figura 3.38, las mismas están dadas en centímetros.

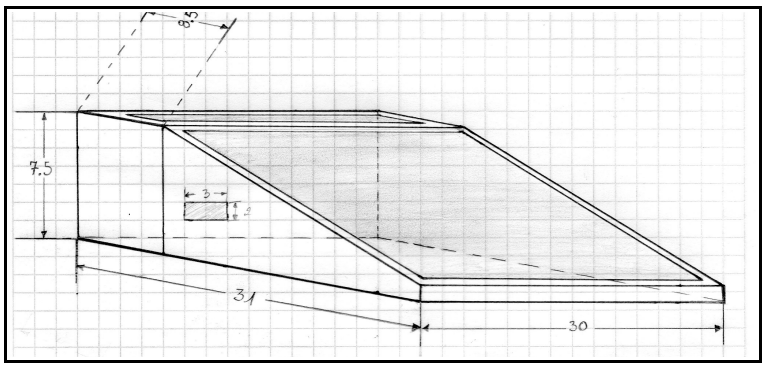


Figura 3.38: Diseño del chasis del Entrenador Lógico.

La fuente de VDC está dentro de la caja y la placa del Entrenador Lógico está situada en la parte sombreada del diseño.

La figura 3.39 muestra el Entrenador Lógico armado y terminado.

Figura 3.39: Entrenador Lógico armado y terminado (a).

Figura 3.39: Entrenador Lógico armado y terminado (b).

Figura 3.39: Entrenador Lógico armado y terminado (c).

3.7. Guía de Prácticas

La Facultad de Ingeniería Electrónica imparte las materias de; Sistemas Digitales, Microcontroladores, Sistemas de Comunicación y Diseño Electrónico, dichas materias son impartidas a estudiantes a partir del séptimo nivel de su carrera profesional.

De acuerdo al pensum de estudios de dichas materias, se ha elaborado un diseño de 10 prácticas que fomentará el aprendizaje.

Las Prácticas que se realizaron son las siguientes:

1. Control de un interruptor a través de RF.
2. Control de giro de un motor de DC a través de RF.
3. Sistema de clave inalámbrica a través de RF.
4. J2ME aplicación ejecutable para el celular.
5. Android aplicación ejecutable para el celular.
6. Control de un interruptor a través de BT.
7. Mini robot controlado por RF.
8. Mini robot controlado por BT través de J2ME.

9. Mini robot controlado por BT a través de Android.
10. Control de las instalaciones de un departamento a través de un teléfono celular con bluetooth.

Los circuitos de las simulaciones se lo realizó en Proteus 7, dichos archivos constan en el CD de prácticas, al igual que la programación de los PICs de cada una de las prácticas.

Estandarización del formato de la guía de Laboratorio.

Se analizaron varios formatos de guías para prácticas de Laboratorio, se logró una depuración de los mismos obteniendo un formato común y estándar para todas las guías que competen a los Laboratorios de éste documento. El formato estándar es el siguiente:

"TEMA"

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TÍTULO

2. OBJETIVOS

2.1 GENERAL

2.2 ESPECÍFICOS

3. MARCO TEÓRICO

4. LISTADO DE MATERIALES Y EQUIPOS

5. PROCEDIMIENTO

6. DIAGRAMAS Y FIGURAS

7. TABULACIONES Y RESULTADOS

8. CONCLUSIONES Y RECOMENDACIONES

9. BIBLIOGRAFÍA

10. ANEXOS

PRÁCTICA N° 1

CONTROL DE UN INTERRUPTOR A TRAVES DE RF

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Control de un interruptor a través de RF.

2. OBJETIVOS.

2.1. OBETIVO GENERAL

Diseñar un sistema de comunicación inalámbrica a través de radio frecuencia a 434 Mhz, mediante los Módulos TLP434 y RLP434.

2.2. OBJETIVOS ESPECÍFICOS

- Diseñar un sistema de control en Proteus 7.
- Realizar el programa en Basic para la comunicación inalámbrica.
- Armar el circuito planteado en el Entrenador Lógico.

3. MARCO TEÓRICO

4. LISTADO DE MATERIALES Y EQUIPOS.

- ✔ Entrenador Lógico.

5. PROCEDIMIENTO.

✔ Implementar el circuito de la figura 5.1 que muestra el transmisor y el receptor de la Práctica N°1.

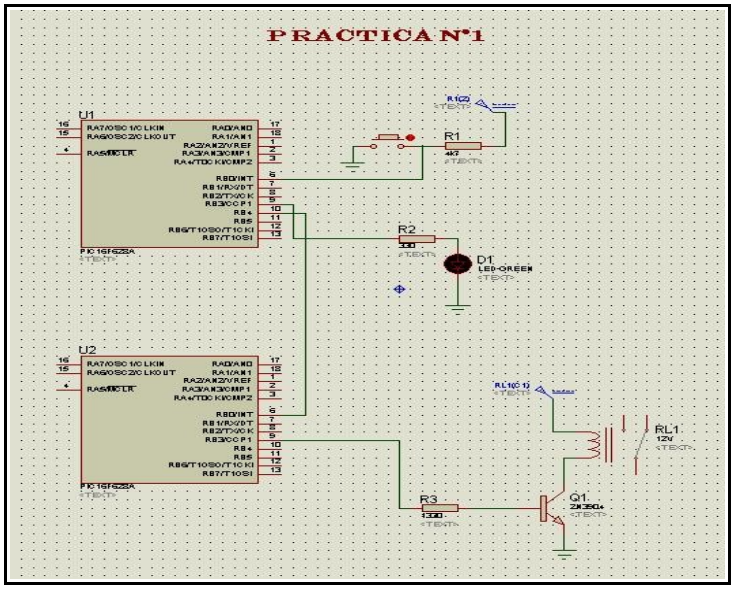


Figura 5.1: Circuito transmisor y receptor.

✔ Realizar la programación del transmisor y receptor utilizando el PIC 16F628A.

✔ Cargar los programas en los PIC.

✔ Comprobar el funcionamiento del circuito implementado.

6. DIAGRAMAS Y FIGURAS.

7. TABULACIÓN Y RESULTADOS.

PULSOS	RELE
1	
2	

Tabla 7.1: Estado del relé según la pulsación.

8. CONCLUSIONES.

9. BIBLIOGRAFÍA.

10. ANEXOS.

PRÁCTICA N° 2

CONTROL DE GIRO DE UN MOTOR DE DC A TRAVÉS DE RF

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Control de un giro de un motor de DC a través de RF.

2. OBJETIVOS.

2.1. OBETIVO GENERAL

Implementar un sistema que controle el sentido de giro de un motor de corriente continua a través de RF.

2.2. OBJETIVOS ESPECÍFICOS

- Diseñar la práctica planteada con un pic diferente al 16F628A.
- Realizar el programa en Basic para la comunicación inalámbrica.
- Armar el circuito planteado en el Entrenador Lógico.

3. MARCO TEÓRICO

4. LISTADO DE MATERIALES Y EQUIPOS.

- ✔ Entrenador Lógico.

5. PROCEDIMIENTO.

✔Implementar el circuito de la figura 5.1 que muestra el transmisor y el receptor de la Práctica N°1.

✔ Realizar la programación del transmisor y receptor utilizando el PIC 12F675.

✔ Cargar los programas en los PIC.

✔ Comprobar el funcionamiento del circuito implementado.

6. DIAGRAMAS Y FIGURAS.

7. TABULACIÓN Y RESULTADOS.

PULSOS	RELE
1	
2	
3	

Tabla 7.1: Estado del motor según la pulsación.

8. CONCLUSIONES.

9. BIBLIOGRAFÍA.

10. ANEXOS.

PRÁCTICA N° 3

SISTEMA DE CLAVE INALÁMBRICA A TRAVÉS DE RF

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Sistema de clave inalámbrica a través de RF.

2. OBJETIVOS.

2.1. OBETIVO GENERAL

Diseñar un sistema de clave inalámbrica a través de radiofrecuencia.

2.2. OBJETIVOS ESPECÍFICOS

- Aprender el funcionamiento de los módulos de RF en conjunto con un LCD 4x20.
- Armar el circuito planteado en el Entrenador Lógico.

3. MARCO TEÓRICO

4. LISTADO DE MATERIALES Y EQUIPOS.

- ✔ Entrenador Lógico.
- ✔ 1 PIC 12F675.
- ✔ 1 PIC 16F628A

5. PROCEDIMIENTO.

✔Implementar el circuito de la figura 5.1 que muestra el transmisor y el receptor de la

Práctica N°1.

✔ Realizar la programación del transmisor y receptor utilizando el PIC 12F675.

✔ Cargar los programas en los PIC.

✔ Comprobar el funcionamiento del circuito implementado.

6. DIAGRAMAS Y FIGURAS.

7. TABULACIÓN Y RESULTADOS.

CLAVE	ESTADO
P1,P1,P2,P3	
XXXX	

Tabla 7.1: Clave inalámbrica.

8. CONCLUSIONES.

9. BIBLIOGRAFÍA.

10. ANEXOS.

PRÁCTICA N° 4

J2ME APLICACIÓN EJECUTABLE PARA EL CELULAR

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: J2ME aplicación ejecutable para el celular.

2. OBJETIVOS

2.1. OBETIVO GENERAL

Diseñar una interfaz gráfica ejecutable para teléfonos celulares con sistema operativo Java.

2.2. OBJETIVOS ESPECÍFICOS

- Estudiar los comandos necesarios en la programación de J2ME con Netbeans.
- Instalar el archivo ejecutable en un teléfono celular.

3. MARCO TEÓRICO

4. LISTADO DE MATERIALES Y EQUIPOS.

- ✔ Entrenador Lógico.
- ✔ 1 Teléfono celular.

5. PROCEDIMIENTO.

✔Establecer una imagen de fondo para la interfaz gráfica del teléfono celular.

✔ Realizar la programación para el teléfono celular.

✔Copiar el archivo ejecutable.

✔ Comprobar el funcionamiento del programa en el teléfono celular.

6. DIAGRAMAS Y FIGURAS.

7. TABULACIÓN Y RESULTADOS.

ARCHIVO	ESTADO
.JAR	
.JAD	

Tabla 7.1: Funcionamiento del los archivos ejecutables.

8. CONCLUSIONES.

9. BIBLIOGRAFÍA.

10. ANEXOS.

PRÁCTICA N° 5

ANDROID APLICACIÓN EJECUTABLE PARA EL CELULAR

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Android aplicación ejecutable para el celular.

2. OBJETIVOS

2.1. OBETIVO GENERAL

Diseñar una interfaz gráfica ejecutable para teléfonos celulares con sistema operativo android.

2.2. OBJETIVOS ESPECÍFICOS

- Estudiar los comandos necesarios en la programación de Android.
- Instalar el archivo ejecutable en un teléfono celular.

3. MARCO TEÓRICO

4. LISTADO DE MATERIALES Y EQUIPOS.

- ✔ Entrenador Lógico.
- ✔ 1 Teléfono celular.

5. PROCEDIMIENTO.

✔Diseñar una interfaz gráfica con los datos del autor.

✔ Realizar la programación para el teléfono celular.

✔Copiar el archivo ejecutable en el teléfono celular.

✔ Comprobar el funcionamiento del programa en el teléfono celular.

6. DIAGRAMAS Y FIGURAS.

7. TABULACIÓN Y RESULTADOS.

ARCHIVO	ESTADO
.APK	

Tabla 7.1: Funcionamiento del archivo ejecutable.

8. CONCLUSIONES.

9. BIBLIOGRAFÍA.

10. ANEXOS.

PRÁCTICA N° 6

CONTROL DE UN INTERUPTOR A TRAVÉS DE BT

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Control de un interruptor a través de BT .

2. OBJETIVOS

2.1. OBETIVO GENERAL

Diseñar una interfaz gráfica capaz de controlar un interruptor a través de bluetooth por medio de J2ME.

2.2. OBJETIVOS ESPECÍFICOS

- Estudiar los comandos necesarios en la programación de J2ME con Netbeans para la comunicación inalámbrica.
- Comprobar el funcionamiento de la comunicación inalámbrica a través de bluetooth.

3. MARCO TEÓRICO

4. LISTADO DE MATERIALES Y EQUIPOS.

- ✔ Entrenador Lógico.
- ✔ 1 Teléfono celular.

5. PROCEDIMIENTO.

✔Realizar el diseño de la interfaz gráfica que se presenta en la pantalla principal de la aplicación en el teléfono celular.

✔ Realizar la programación para el teléfono celular.

✔Copiar el archivo ejecutable en el teléfono celular.

✔ Comprobar el funcionamiento del programa en el teléfono celular.

6. DIAGRAMAS Y FIGURAS.

7. TABULACIÓN Y RESULTADOS.

ARCHIVO	ESTADO
ACTIVADO	
DESACTIVADO	

Tabla 7.1: Funcionamiento del archivo ejecutable.

8. CONCLUSIONES.

9. BIBLIOGRAFÍA.

10. ANEXOS.

PRÁCTICA N° 7

MINI ROBOT CONTROLADO POR RF

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Minirobot controlado por RF.

2. OBJETIVOS

2.1. OBETIVO GENERAL

Diseñar e implementar un mini robot controlado por radio frecuencia.

2.2. OBJETIVOS ESPECÍFICOS

- Aplicar los conocimientos adquiridos en RF durante las prácticas anteriores.
- Comprobar el funcionamiento de la comunicación inalámbrica entre el Entrenador Lógico y el Minirobot.

3. MARCO TEÓRICO

4. LISTADO DE MATERIALES Y EQUIPOS.

- Entrenador Lógico.
- Minirobot.

5. PROCEDIMIENTO.

- Establecer los movimientos del Minirobot.
- Realizar la programación para el TX.
- Realizar la programación para el RX.
- Comprobar el funcionamiento del Entrenador Lógico y del Minirobot.

6. DIAGRAMAS Y FIGURAS.

7. TABULACIÓN Y RESULTADOS.

ARCHIVO	ESTADO
ADELANTE	
ATRAS	
DERECHA	
IZQUIERDA	
PARAR	

Tabla 7.1: Funcionamiento del archivo ejecutable.

8. CONCLUSIONES.

9. BIBLIOGRAFÍA.

10. ANEXOS.

PRÁCTICA N° 8

MINI ROBOT CONTROLADO POR BT A TRAVÉS DE J2ME

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Minirobot controlado por BT a través de J2ME.

2. OBJETIVOS

2.1. OBETIVO GENERAL

Diseñar e implementar un mini robot controlado por bluetooth a través de J2ME.

2.2. OBJETIVOS ESPECÍFICOS

- Desarrollar una interfaz gráfica, como aplicación java, para un teléfono celular capaz de controlar el mini robot.
- Comprobar el funcionamiento de la comunicación inalámbrica entre el teléfono celular y el Minirobot.

3. MARCO TEÓRICO

4. LISTADO DE MATERIALES Y EQUIPOS.

- Entrenador Lógico.
- Minirobot.
- Teléfono celular.
- Cables de par trenzado.

5. PROCEDIMIENTO.

- Establecer los movimientos del Minirobot.

  El Minirobot esta diseñado para realizar los siguientes movimientos:

  - Adelante.
  - Atrás.
  - Derecha.
  - Izquierda.
  - Parar.

- Realizar la programación para el TX.
- Realizar la programación para el RX.
- Comprobar el funcionamiento entre el teléfono celular y el Minirobot.

6. DIAGRAMAS Y FIGURAS.

7. TABULACIÓN Y RESULTADOS.

ARCHIVO	ESTADO
ADELANTE	
ATRAS	
DERECHA	
IZQUIERDA	
PARAR	

Tabla 7.1: Funcionamiento del archivo ejecutable .JAR.

8. CONCLUSIONES.

9. BIBLIOGRAFÍA.

10. ANEXOS.

PRÁCTICA N° 9

MINI ROBOT CONTROLADO POR BT A TRAVÉS DE ANDROID

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Minirobot controlado por BT a través de ANDROID.

2. OBJETIVOS

2.1. OBETIVO GENERAL

Diseñar e implementar un mini robot controlado por bluetooth a través de ANDROID.

2.2. OBJETIVOS ESPECIFICOS

- Desarrollar una interfaz gráfica, como aplicación Android, para un teléfono celular capaz de controlar el mini robot.
- Comprobar el funcionamiento de la comunicación inalámbrica entre el teléfono celular y el Minirobot.

3. MARCO TEÓRICO

4. LISTADO DE MATERIALES Y EQUIPOS.

- Entrenador Lógico.
- Minirobot.
- Teléfono celular.
- Cables de par trenzado.

5. PROCEDIMIENTO.

- Establecer los movimientos del Minirobot.

  El Minirobot esta diseñado para realizar los siguientes movimientos:

  - Adelante.
  - Atrás.
  - Derecha.
  - Izquierda.
  - Parar.

- Realizar la programación para el TX.
- Realizar la programación para el RX.
- Comprobar el funcionamiento entre el teléfono celular y el Minirobot.

6. DIAGRAMAS Y FIGURAS.

7. TABULACIÓN Y RESULTADOS.

ARCHIVO	ESTADO
ADELANTE	
ATRAS	
DERECHA	
IZQUIERDA	
PARAR	

Tabla 7.1: Funcionamiento del archivo ejecutable .APK.

8. CONCLUSIONES.

9. BIBLIOGRAFÍA.

10. ANEXOS.

PRÁCTICA N° 10

CONTROL DE LAS INSTALACIONES DE UN DEPARTAMENTO A TRAVÉS DE UN TELÉFONO CELULAR CON BLUETOOTH

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO:

Control de las instalaciones de un departamento a través de un teléfono celular con bluetooth.

2. OBJETIVOS

2.1. OBETIVO GENERAL

Diseñar e implementar el control de las instalaciones de un departamento por bluetooth a través de J2ME.

2.2. OBJETIVOS ESPECÍFICOS

- Desarrollar una interfaz gráfica, como aplicación java, para un teléfono celular capaz de controlar tres áreas del departamento.
- Comprobar el funcionamiento de la comunicación inalámbrica entre el teléfono celular y las áreas del departamento.

3. MARCO TEÓRICO

4. LISTADO DE MATERIALES Y EQUIPOS.

- Entrenador Lógico.

- Minirobot.
- Teléfono celular.
- Cables de par trenzado.

5. PROCEDIMIENTO.

- Establecer las áreas del departamento a controlar.
  1. Iluminación principal.
  2. Control de la alarma de seguridad.
  3. Sistema de ingreso automático.
- Realizar la programación para el TX.
- Realizar la programación para el RX.
- Comprobar el funcionamiento entre el teléfono celular y el Minirobot.

6. DIAGRAMAS Y FIGURAS.

7. TABULACIÓN Y RESULTADOS.

ARCHIVO	ESTADO
LUCES	
ALARMA	
SIST.INGRESO	

Tabla 7.1: Funcionamiento del archivo ejecutable .JAR.

8. CONCLUSIONES.

9. BIBLIOGRAFÍA.

10. ANEXOS.

Desarrollo de las Prácticas.

Se realizó un solucionario como parte del presente proyecto donde se encuentra todas las prácticas planteadas, desarrolladas con sus respectiva programación, y simulaciones.

3.8. Programación Interfaz gráfica.

Introducción

Aquí se describe los pilares sobre los que se asienta la plataforma J2ME para que el estudiante domine las clases que componen esta edición de Java para que pueda realizar rápidamente sus propias aplicaciones para teléfonos que dispongan de esta tecnología. Para la realización de dichas aplicaciones se usará software de libre distribución disponible en la página oficial de Sun Microsystems .Se usará también emuladores dónde se podrá ejecutar y depurar los programas, evitando así la descarga insatisfactoria de la aplicación en un verdadero terminal.

J2ME es la versión de Java orientada a los dispositivos móviles. Debido a que éstos tienen una potencia de cálculo baja e interfaces de usuario pobres, es necesaria una versión específica de Java destinada a estos dispositivos, ya que el resto de versiones de Java, J2SE o J2EE, no encajan dentro de este esquema. J2ME es por tanto, una versión “reducida” de J2SE.

Configuraciones

Una configuración es el conjunto mínimo de APIs Java que permiten desarrollar aplicacio-

nes para un grupo de dispositivos. Estas APIs describen las características básicas, comunes a todos los dispositivos, que son:

• Características soportadas del lenguaje de programación Java.

• Características soportadas por la Máquina Virtual Java.

• Bibliotecas básicas de Java y APIs soportadas.

Existen dos configuraciones en J2ME:

CLDC, orientada a dispositivos con limitaciones computacionales y de memoria y CDC, orientada a dispositivos con no tantas limitaciones.

La CDC está basada en J2SE v1.3 e incluye varios paquetes Java de la edición estándar. Las peculiaridades de la CDC están contenidas principalmente en el paquete javax.microedition.io, que incluye soporte para comunicaciones http y basadas en datagramas. La Tabla 3.6 muestra las librerías incluidas en la CDC.

Nombre de Paquete CDC	Descripción
java.io	Clases e interfaces estándar de E/S.
java.lang	Clases básicas del lenguaje.
java.lang.ref	Clases de referencia.
java.lang.reflect	Clases e interfaces de reflection.
java.math	Paquete de matemáticas.
java.net	Clases e interfaces de red.
java.security	Clases e interfaces de seguridad
java.security.cert	Clases de certificados de seguridad.
java.text	Paquete de texto.
java.util	Clases de utilidades estándar.
java.util.jar	Clases y utilidades para archivos JAR.
java.util.zip	Clases y utilidades para archivos ZIP y comprimidos.
javax.microedition.io	Clases e interfaces para conexión genérica CDC.

Tabla 3.6: Librerías de CDC.

La CLDC está orientada a dispositivos dotados de conexión y con limitaciones en cuanto a capacidad gráfica, cómputo y memoria. Un ejemplo de estos dispositivos son: teléfonos móviles, buscapersonas (pagers), PDAs, organizadores personales, etc.

CLDC está orientado a dispositivos con ciertas restricciones. 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 deben cumplir los siguientes requisitos:

1. Disponer entre 160 Kb y 512 Kb de memoria total disponible.

2. Procesador de 16 o 32 bits con al menos 25 Mhz de velocidad.

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

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

La tabla 3.7 muestra las liberías de la CLDC.

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

Tabla 3.6: Librerías de CLDC.

Perfiles

El perfil es el que define las APIs que controlan el ciclo de vida de la aplicación, interfaz de usuario, etc. Más concretamente, un perfil es un conjunto de APIs orientado a un ámbito de aplicación determinado. Los perfiles identifican un grupo de dispositivos por la funcionalidad que proporcionan (electrodomésticos, teléfonos móviles, etc.) y el tipo de aplicaciones que se ejecutarán en ellos. Las librerías de la interfaz gráfica son un componente muy importante en la definición de un perfil.

J2ME y las comunicaciones

Una característica importante que deben tener los dispositivos que hagan uso de J2ME, más específicamente CLDC/MIDP es que necesitan poseer conexión a algún tipo de red, por lo que la comunicación de estos dispositivos cobra una gran importancia. Ahora se mostrará cómo participan las distintas tecnologías en estos dispositivos y cómo influyen en el uso de la tecnología J2ME. Para ello se analiza un dispositivo en especial: los teléfonos móviles. De aquí en adelante todo el estudio y la creación de aplicaciones se reali-

zarán para este dispositivo. Esto es así debido a la rápida evolución que han tenido los teléfonos móviles en el sector de las comunicaciones, lo que ha facilitado el desarrollo, por parte de algunas empresas, de herramientas que se usará para crear las aplicaciones.

Desarrollo

El proceso de creación de MIDlets se puede realizar básicamente de dos formas:

• A través de la línea de comandos

• A través de un entorno visual:

Los MIDlets creados serán ejecutados en dispositivos MID (Mobile Information Device) y no en la máquina donde se los desarrolle. Por esta razón, sea cual sea el método de creación que se use, se tendrá que hacer uso de algún emulador para realizar las pruebas de la aplicación. Este emulador puede representar a un dispositivo genérico o puede ser de algún modelo de MID específico.

Las etapas básicas que han de realizarse con este objetivo:

1. Desarrollo
2. Compilación
3. Preverificación
4. Empaquetamiento
5. Ejecución
6. Depuración

Estructura de los MIDlets

Los MIDlets tienen la siguiente estructura:

```
import javax.microedition.midlet.*
public class MiMidlet extends MIDlet
public MiMidlet() {
/* Este es el constructor de clase. Aquí se
inicializa las variables.
```

```
*/
}
public startApp(){
/* Aquí se incluye  el código que  el
MIDlet ejecuta cuando se activa
*/
}
public pauseApp(){
/* Aquí se incluye  el código que  el
MIDlet ejecuta cuando entra en el estado de pausa
(Opcional)
*/
}
public destroyApp(){
/* Aquí se incluye  el código que  el
MIDlet ejecuta cuando sea destruido. Normalmente
aquí se liberarán los recursos ocupados por el
MIDlet como memoria, etc. (Opcional)
*/
}
}
```

Ejemplo:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class HolaMundo extends MIDlet{
private Display pantalla;
```

```
private Form formulario = null;
public HolaMundo(){
pantalla = Display.getDisplay(this);
formulario = new Form(“Hola Mundo”);
}
public void startApp(){
pantalla.setCurrent(formulario);
}
public void pauseApp(){
}
public void destroyApp(boolean unconditional){
pantalla = null;
formulario = null;
notifyDestroyed();
}
}
```

Estos métodos son los que obligatoriamente tienen que poseer todos los MIDlets ya que la clase que se ha creado tiene que heredar de la clase MIDlet y ésta posee tres métodos abstractos: startApp(), pauseApp() y destroyApp() que han de ser implementados por cualquier MIDlet.

CAPITULO IV

PRUEBAS Y VALIDACIÓN

4.1 GUIA DE PRÁCTICAS DESARROLLADAS

Introducción

El presente capítulo trata sobre la implementación y desarrollo de cada una de las prácticas planteadas para el Entrenador Lógico.

Las prácticas descritas están totalmente desarrolladas de acuerdo a un formato establecido en coordinación con la Facultad de Ingeniería Electrónica de la Universidad Israel en el cual consta la elaboración de hardware y software para el correcto aprendizaje y entendimiento de las comunicaciones inalámbricas a través de radiofrecuencia y bluetooth.

PRÁCTICA N° 1
CONTROL DE UN INTERRUPTOR A TRAVES DE RF

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Control de un interruptor a través de RF.

2. OBJETIVOS.

2.1. OBETIVO GENERAL

Diseñar un sistema de comunicación inalámbrica a través de radio frecuencia a 434 Mhz, mediante los Módulos TLP434 y RLP434.

2.2. OBJETIVOS ESPECÍFICOS

- Diseñar un sistema de control en Proteus 7.
- Realizar el programa en Basic para la comunicación inalámbrica.
- Armar el circuito planteado en el Entrenador Lógico.

3. MARCO TEÓRICO

Radiofrecuencia.

En la actualidad diversos productos de consumo propio e industriales utilizan la energía electromagnética. Hoy en día la energía de radiofrecuencia, un tipo de energía electromagnética, está aumentando su importancia a nivel mundial incluyendo ondas de radio y microondas, las cuales son utilizadas en comunicación y radiodifusión.

Las ondas de radio, son formas de energía electromagnética comúnmente identificadas por las siglas RF por el término de radiofrecuencia. Las emisiones de

RF y los fenómenos asociados pueden ser discutidos en términos de energía, radiación o campos.

Ya que la radiación es definida como la propagación de energía a través del espacio en forma de ondas o partículas, la radiación electromagnética se puede describir como ondas de energía eléctrica y magnética moviéndose conjuntamente a través del espacio.

Las ondas de radiofrecuencia y las microondas son generadas por el movimiento de cargas eléctricas en algún material conductor, o bien en una antena, logrando penetrar las nubes, la niebla y las paredes. Por ello su uso para la difusión de radio, televisión, telefonía celular, etc., las cuales pueden ser recibidas por una antena de techo, de radio de automóvil o una antena de teléfono celular.

Los módulos de radiofrecuencia TLP Y RLP 434 comprenden la región del espectro electromagnético de ondas UHF entre 300MHz y 3000MHz, además de trabajar con modulación ASK y FSK.

Modulación ASK

Este tipo de modulación es la más sencilla para la transmisión de datos digitales y es conocida como ASK por sus siglas en inglés Amplitude Shift Keying (Modulación por desplazamiento de amplitud). Es una forma de modulación en la cual se representan los datos digitales como variaciones en la amplitud de la onda portadora.

Modulación FSK

Al igual que la modulación ASK, la modulación FSK es una conmutación sencilla y es conocida de esta forma por sus siglas en inglés Frequency Shift Keying (Modulación por desplazamiento de frecuencia).

4. LISTADO DE MATERIALES Y EQUIPOS.

- ✔ Entrenador Lógico.
- ✔ 2 PIC 16F628A.

✔ Cables de par trenzado.

5. PROCEDIMIENTO.

✔ Implementar el circuito de la figura 5.1 que muestra el transmisor y el receptor de la Práctica N°1.

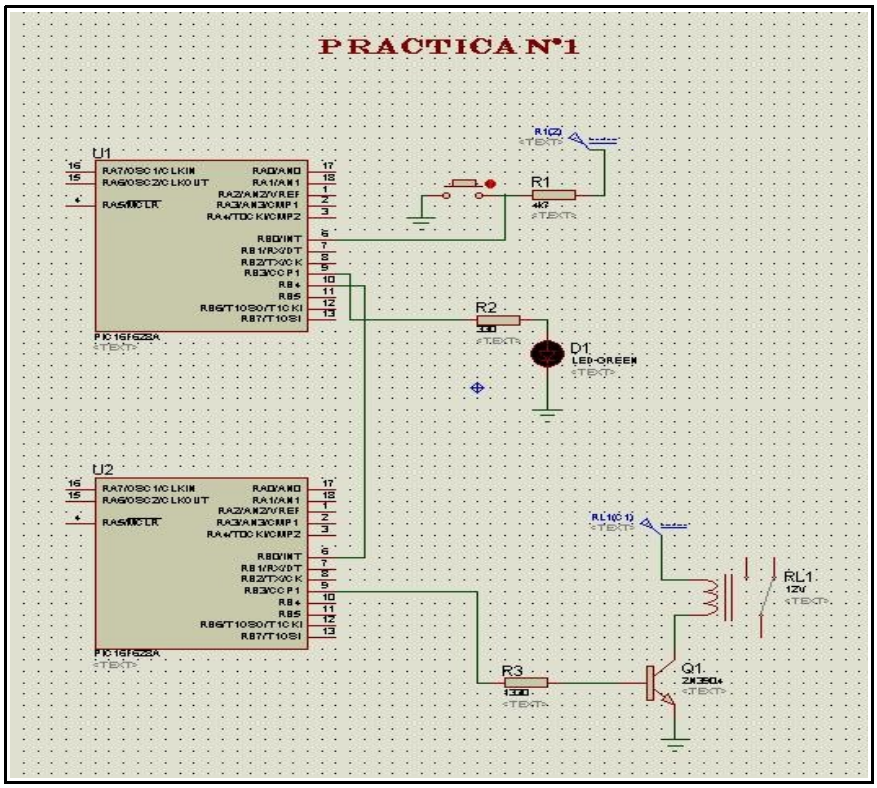


Figura 5.1: Circuito transmisor y receptor.

✔ Realizar la programación del transmisor y receptor utilizando el PIC 16F628A.

PROGRAMACIÒN DEL PIC
TX

```
'*****************************************************************
'*  Name    : PRACTICA Nº1 TX
'*  Author  : EDISON JAVIER TERAN GUALOTO
'*  Notice  : Copyright (c) 2012 EDISON TERAN
'*          : All Rights Reserved
'*  Date    : 27/01/2012
'*  Version : 1.0
'*  Notes   : PRÀCTICA Nº1
'*          :
'*****************************************************************
;libreria para radio frecuencia, incluye los modos de comunicaciòn
include "modedefs.bas"
;ASIGNACION DE NOMBRES A LOS PUERTOS
PULSO       VAR PORTB.0
INDICADOR   VAR PORTB.3
TRANSMISOR  VAR PORTB.4
;SUBRUTINA PARA COMPROBAR SI SE PRESIONÒ EL PUERTB.0
  TX:
    IF PULSO=0 THEN ENVIAR
  GOTO TX
;SUBRUTINA PARA EL ENVIO DE DATOS
  ENVIAR:
;ANTIREBOTE
    IF PULSO=0 THEN ENVIAR
    PAUSE 200
;ENVIO DE DATOS POR EL PUERTO TRANSMISOR B.4
    SEROUT TRANSMISOR,N2400,["X"]

;CONFIRMACION DE ENVIO
    HIGH INDICADOR
    PAUSE 120
    LOW INDICADOR
    pause 120
    HIGH INDICADOR
    PAUSE 120
    LOW INDICADOR
  GOTO TX
```

PROGRAMACIÒN DEL PIC
RX

```
'*****************************************************************
'*  Name    : PREACTICA Nº1 RX
'*  Author  : EDISON JAVIER TERAN GUALOTO
'*  Notice  : Copyright (c) 2012 EDISON TERAN
'*          : All Rights Reserved
'*  Date    : 27/01/2012
'*  Version : 1.0
'*  Notes   :
'*          :
'*****************************************************************
;libreria para radio frecuencia, incluye los modos de comunicaciòn
include "modedefs.bas"

;ASIGNACION DE NOMBRES A LOS PUERTOS
```

```
DATOS       VAR BYTE
RELE        VAR PORTB.3
RECEPTOR    VAR PORTB.0
CONT        VAR BYTE
CONT=0
;RECIBIR DATOS
INICIO:
  SERIN receptor,N2400,DATOS
  IF DATOS="X" THEN ABRIR
GOTO INICIO
;CODIGO PARA PRENDER Y APAGAR EL RELE EN CADA PULSACIÒN
  ABRIR:
    CONT=CONT+1
    IF CONT>1 THEN GOSUB CERO
    IF CONT=0 THEN LRELE
    IF CONT=1 THEN HRELE
    HIGH RELE
    PAUSE 5000
    LOW RELE
  GOTO INICIO
;SUBRUTINA DE APAGADO DEL RELÈ
       LRELE:
       LOW RELE
       GOTO INICIO
;SUBRUTINA DE ENCENDIDO DEL RELÈ
       HRELE:
       HIGH RELE
       GOTO INICIO
;ENCERDAO DE LA VARIABLE CONT
CERO:
CONT=0
RETURN
```

✔ Cargar los programas en los PIC.

Los PIC se graban utilizando el software winpic800.

✔ Comprobar el funcionamiento del circuito implementado.

6. DIAGRAMAS Y FIGURAS.

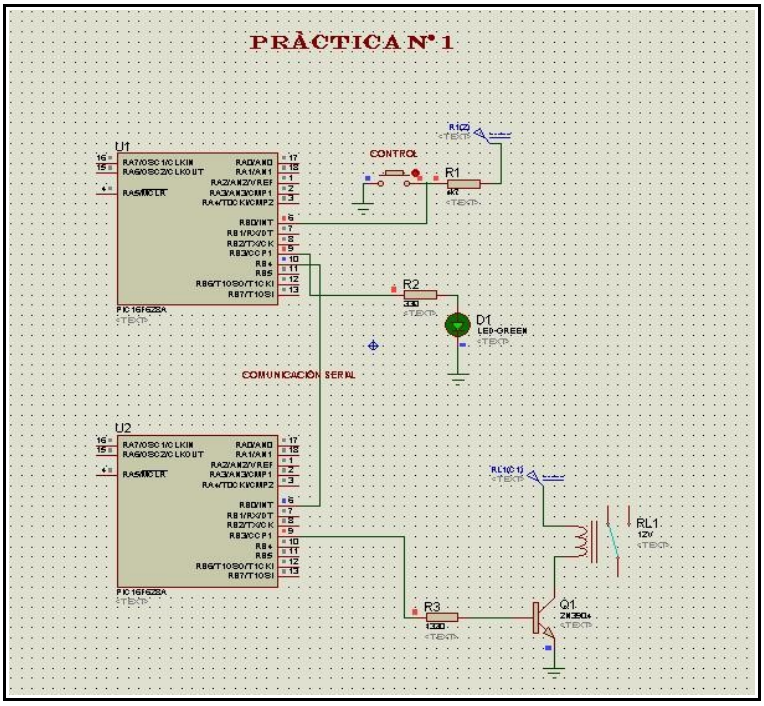


Figura 6.1: Práctica N° 1 en estado activado.

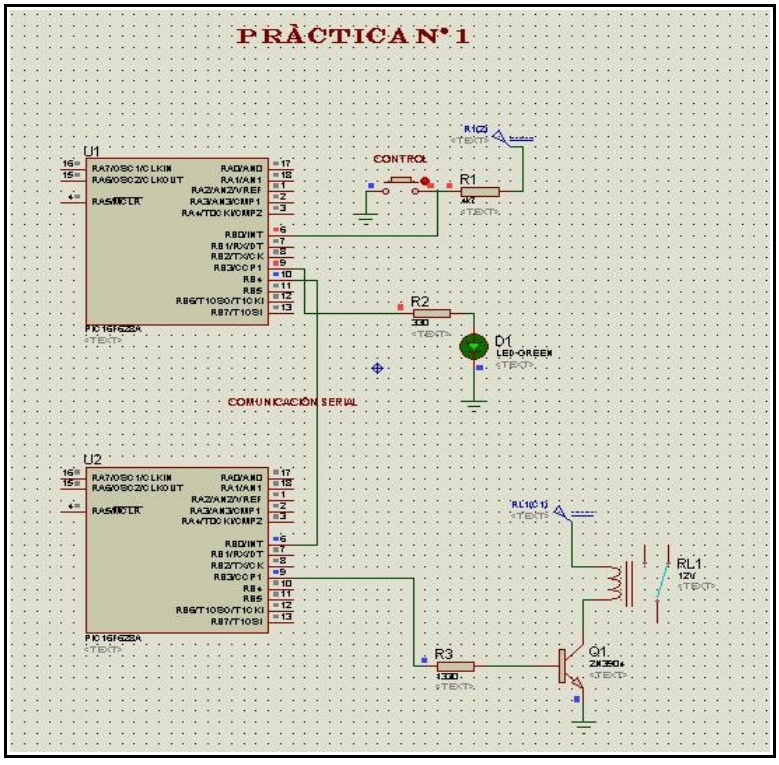


Figura 6.2: Práctica N°1 en estado desactivado.

7. TABULACIÓN Y RESULTADOS.

PULSOS	RELE
1	ACTIVADO
2	DESACTIVADO

Tabla 7.1: Estado del relé según la pulsación.

8. CONCLUSIONES.

- Los módulos de RF que posee el Entrenador Lógico trabajan a una frecuencia de 434Mhz.
- Los módulos de RF simulan una comunicación serial la cual luego la transmite de forma inalámbrica.
- En Micro Code Studio las declaraciones SERIN y SEROUT proporcionan una comunicación serial.
- El Entrenador Lógico trabaja con normalidad y precisión con la práctica planteada.

9. BIBLIOGRAFÍA.

- http://es.wikipedia.org/wiki/Radiofrecuencia
- http://www.forosdeelectronica.com/f17/manejo-modulos-tlp434-rlp434-441/
- http://www.ideastechnology.com/index.php?option=com_content&view=article&id=17%3Aradiocontrol-con-el-tlp434-y-rlp434&catid=27%3Aproyectos-y-disenos&Itemid=2&lang=en

10. ANEXOS.

PRÁCTICA N° 2
CONTROL DE GIRO DE UN MOTOR DE DC A TRAVÉS DE RF

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Control de un giro de un motor de DC a través de RF.

2. OBJETIVOS.

2.1. OBETIVO GENERAL

Implementar un sistema que controle el sentido de giro de un motor de corriente continua a través de RF.

2.2. OBJETIVOS ESPECIFÍCOS

- Diseñar la práctica planteada con un pic diferente al 16F628A.
- Realizar el programa en Basic para la comunicación inalámbrica.
- Armar el circuito planteado en el Entrenador Lógico.

3. MARCO TEÓRICO

CONTROL DE UN MOTOR MEDIANTE RELÉ

En muchos proyectos de Tecnología es necesario controlar el giro, en ambos sentidos, de un pequeño motor eléctrico de corriente continua.

Dicho control puede hacerse con una llave de cruce o con un conmutador doble, pero también se puede hacer con un relé, como se muestra a continuación.

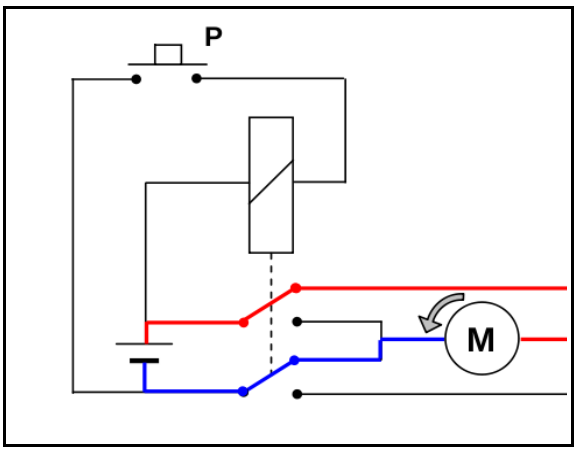


Figura 3.1: Giro del motor en sentido antihorario.

La Fig. 3.1 La bobina del relé se ha conectado a la pila a través de un pulsador NA (normalmente abierto) que designamos con la letra P. El motor se ha conectado a los contactos fijos del relé del mismo modo que si se tratase de un conmutador doble. Los dos polos del relé se conectan a los borne de la pila.

En esta situación al motor le llega la corriente por el borne derecho y le sale por el izquierdo, girando en sentido antihorario (Fig. 3.1).

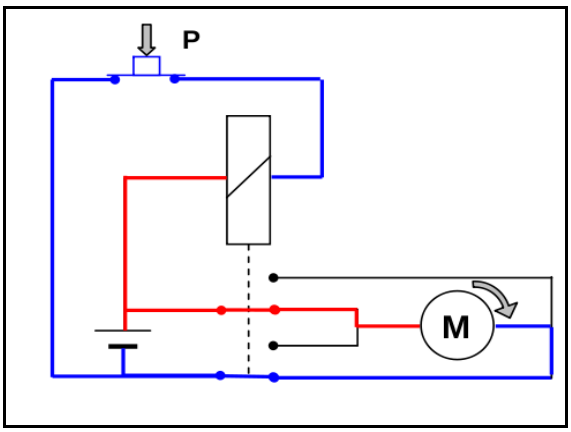


Figura 3.2: Giro del motor en sentido horario.

Al accionar el pulsador P en la figura 3.2 se suministra corriente a la bobina del relé, haciendo ésta que los contactos móviles cambien de posición, con lo cual la corriente le llega al motor por su borne izquierdo y le sale por el derecho, girando en sentido horario.

4. LISTADO DE MATERIALES Y EQUIPOS.

- ✔ Entrenador Lógico.
- ✔ 2 PIC 12F675.
- ✔ Cables de par trenzado.

5. PROCEDIMIENTO.

✔Implementar el circuito de la figura 5.1 que muestra el transmisor y el receptor de la Práctica N°1.

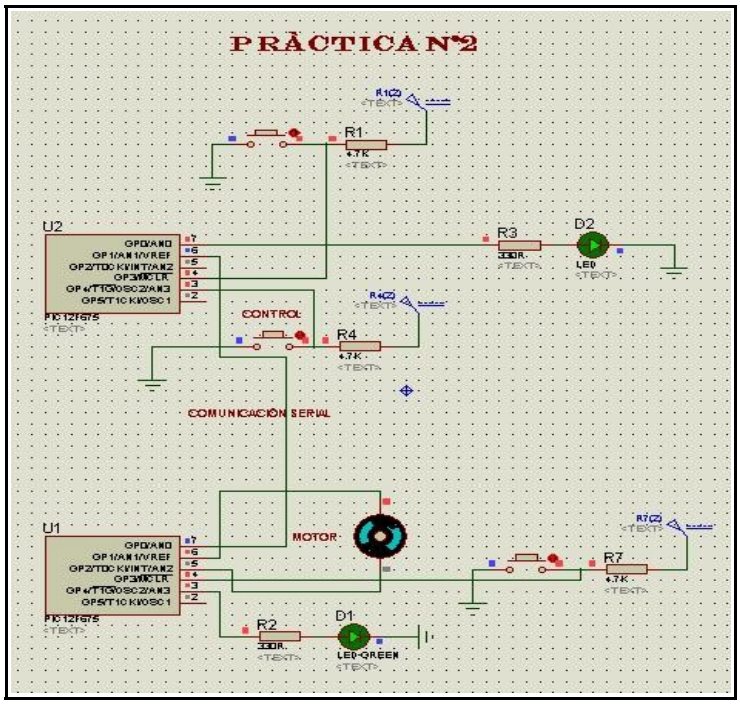


Figura 5.1: Circuito transmisor y receptor.

✔ Realizar la programación del transmisor y receptor utilizando el PIC 12F675.

PROGRAMACIÒN DEL PIC
TX

```
'*****************************************************************
'*  Name    : PRACTICA Nº2 TX
'*  Author  : EDISON JAVIER TERAN GUALOTO
'*  Notice  : Copyright (c) 2012 EDISON TERAN
'*          : All Rights Reserved
'*  Date    : 30/01/2012
'*  Version : 1.0
'*  Notes   :
'*          :
'*****************************************************************
;libreria para radio frecuencia, incluye los modos de comunicaciòn
include "modedefs.bas"
CMCON=%111
ANSEL=0
;ASIGNACION DE NOMBRES A LOS PUERTOS
GIRO          VAR GPIO.4
TRANSMISOR    VAR GPIO.1
INDICADOR     VAR GPIO.0
;SUBRUTINA PARA TRANSMITIR
  TX:
    IF GIRO=0   THEN GIRO1
  GOTO TX
;SUBRUTINA PARA EL ENVIO DE DATOS
  GIRO1:
;ANTIREBOTE
    IF GIRO=0 THEN GIRO1
    PAUSE 200
;ENVIO DE DATOS POR EL PUERTO TRANSMISOR
    SEROUT TRANSMISOR,N2400,["X"]
;CONFIRMACION DE ENVIO
    HIGH INDICADOR
    PAUSE 120
    LOW INDICADOR
    pause 120
    HIGH INDICADOR
    PAUSE 120
    LOW INDICADOR
  GOTO TX
```

PROGRAMACIÒN DEL PIC
RX

```
'*****************************************************************
'*  Name    : PRACTICA Nº2 RX                                    *
'*  Author  : EDISON JAVIER TERAN GUALOTO                          *
'*  Notice  : Copyright (c) 2012 EDISON TERAN                    *
'*          : All Rights Reserved                                *
'*  Date    : 30/01/2012                                         *
'*  Version : 1.0                                               *
'*  Notes   :                                                   *
'*          :                                                  *
'*****************************************************************
;libreria para radio frecuencia, incluye los modos de comunicaciòn
include "modedefs.bas"
```

```
CMCON=7
ANSEL=0
;ASIGNACION DE NOMBRES A LOS PUERTOS Y VARIABLES
DATOS       VAR BYTE
CONT        VAR BYTE
RECEPTOR    VAR GPIO.0
LED         VAR GPIO.4
POSMOT      VAR GPIO.1
NEGMOT      VAR GPIO.2
;INICIA EL CONTADOR A CERO
CONT=0
;RECIBIR DATOS
INICIO:
  SERIN receptor,N2400,DATOS
  IF DATOS="X" THEN ABRIR
GOTO INICIO
;CODIGO PARA PRENDER Y APAGAR EL RELE EN CADA PULSACIÒN
  ABRIR:
   HIGH LED
   PAUSE 120
   LOW LED
   PAUSE 120
   HIGH LED
   PAUSE 120
   LOW LED
;CODIGO PARA EL GIRO DEL MOTOR
    CONT=CONT+1
    IF CONT>2 THEN GOSUB CERO
    IF CONT=0 THEN DERECHA
    IF CONT=1 THEN IZQUIERDA
    IF CONT=2 THEN PARAR
  GOTO INICIO
;SUBRUTINA DE GIRO A LA IZQUIERDA
       IZQUIERDA:
       HIGH POSMOT
       LOW  NEGMOT
       GOTO INICIO
;SUBRUTINA DE GIRO A LA DERECHA
       DERECHA:
       HIGH NEGMOT
       LOW  POSMOT
       GOTO INICIO
;SUBRUTINA PARA PARAR AL MOTOR
       PARAR:
       LOW POSMOT
       LOW NEGMOT
       GOTO INICIO
 ;ENCERADO DE LA VARIABLE CONT
CERO:
CONT=0
        RETURN
```

✔ Cargar los programas en los PIC.

Los PIC se graban utilizando el software winpic800.

✔ Comprobar el funcionamiento del circuito implementado.

6. DIAGRAMAS Y FIGURAS.

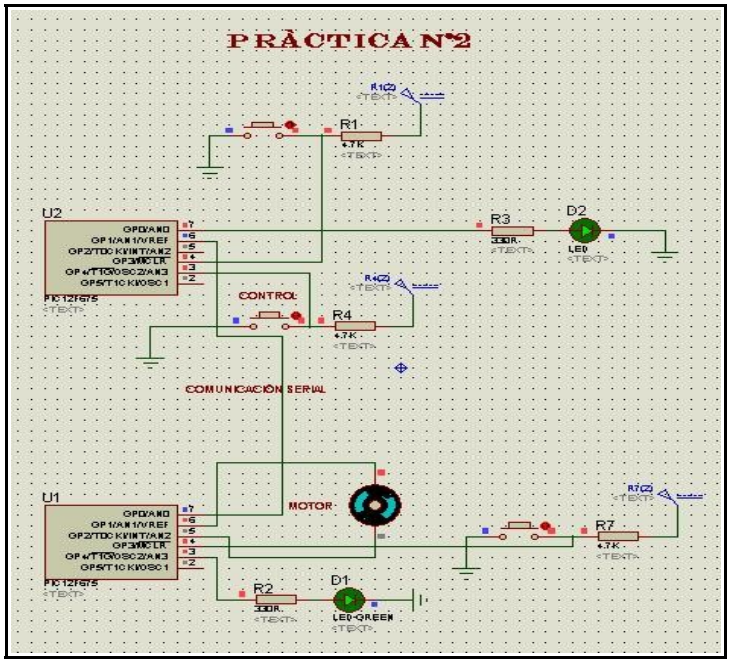


Figura 6.1: Práctica N°2, motor en sentido horario.

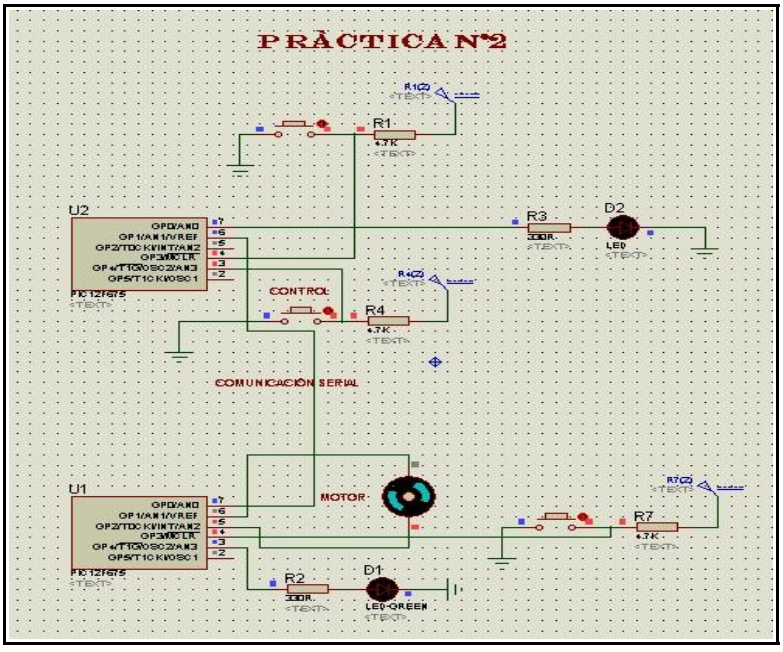


Figura 6.2: Práctica N°2, motor apagado.

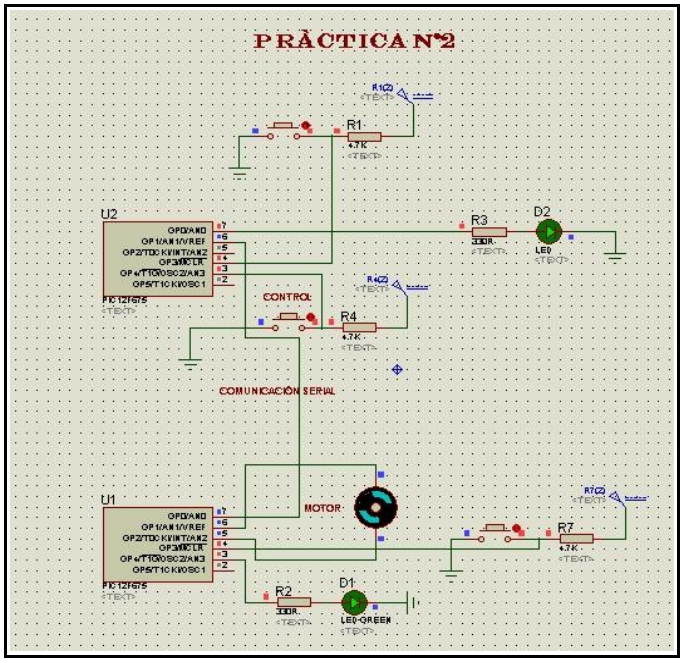


Figura 6.3: Práctica N°2, motor en sentido antihorario.

7. TABULACIÓN Y RESULTADOS.

PULSOS	RELE
1	HORARIO
2	ANTIHORARIO
3	PARA

Tabla 7.1: Estado del motor según la pulsación.

8. CONCLUSIONES.

- El circuito de control de giro funciona correctamente y para optimizar esta práctica es necesario utilizar un puente H.
- El PIC 12f675 necesita un pulsador como MCLC y dasctivar los comparadores analogos para su correcto funcionamiento.
- En Micro Code Studio las declaraciones SERIN y SEROUT proporcionan una comunicación serial.
- El Entrenador Lógico trabaja con normalidad y precisión con la práctica planteada.

9. BIBLIOGRAFÍA.

- http://www.google.com/search?client=ubuntu&channel=fs&q=CIRCUITOS+DE+CONTROL+RELE&ie=utf-8&oe=utf-8

- http://r-luis.xbot.es/ebasica2/mcc_02.html

10. ANEXOS.

PRÁCTICA N° 3
SISTEMA DE CLAVE INALAMBRICA A TRAVÉS DE RF

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Sistema de clave inalámbrica a través de RF.

2. OBJETIVOS.

2.1. OBETIVO GENERAL

Diseñar un sistema de clave inalámbrica a través de radiofrecuencia.

2.2. OBJETIVOS ESPECÍFICOS

- Aprender el funcionamiento de los módulos de RF en conjunto con un LCD 4x20.
- Armar el circuito planteado en el Entrenador Lógico.

3. MARCO TEÓRICO

Una forma sencilla de saber lo que pasa en el microcontrolador es usar un LCD, mediante su uso se puede comprobar el estado de los registros internos del micro, verificar el funcionamiento de la electrónica y los programas que hagamos, por ejemplo si queremos comprobar que la lectura de un ADC en nuestro programa es correcta, pues no tenemos más opciones que usar un LCD o el PC para poder visualizar lo que está leyendo el micro.

Un LCD es un dispositivo que muestra caracteres alfanuméricos en una pantalla de varias líneas, todo LCD lleva un microcontrolador interno que se encarga de gobernar su funcionamiento (el más común es el Hitachi 44780), este microcontrolador tiene unos pines para comunicarse con el mundo exterior y así poder realizar las distintas operaciones sobre la pantalla del LCD, lo que hace que manejar la pantalla sea

muy sencillo y que merezca la pena añadirlo a cualquier proyecto mientras se realiza. En la figura 3.1 se muestra el esquema de un LCD.

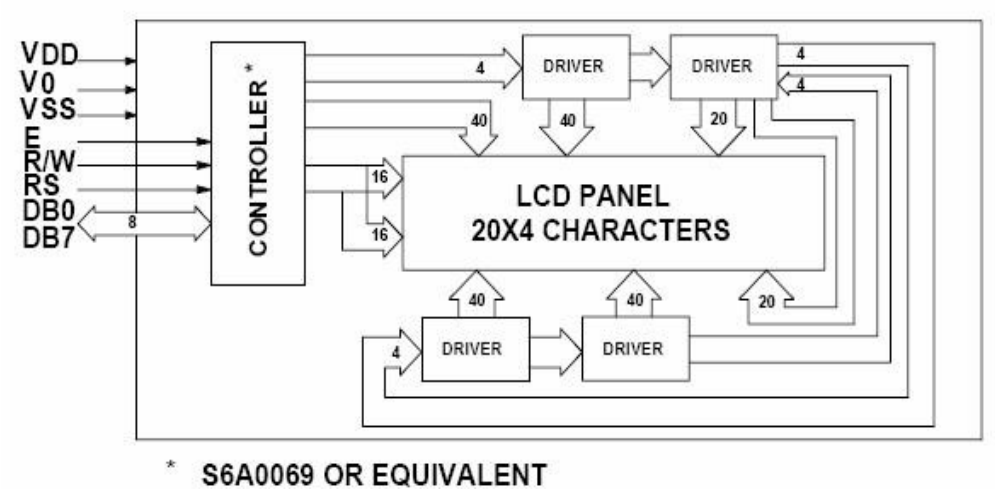


Figura 3.1: Esquema de un LCD.

El LCD dispone de los siguientes pines.

Pin No.	Symbol	Level	Description
1	V_{SS}	0V	Ground
2	V_{DD}	5.0V	Supply Voltage for logic
3	V0	(Variable)	Operating voltage for LCD
4	RS	H/L	H: DATA, L: Instruction code
5	R/W	H/L	H: Read(MPU→Module) L: Write(MPU→Module)
6	E	H,H→L	Chip enable signal
7	DB0	H/L	Data bit 0
8	DB1	H/L	Data bit 1
9	DB2	H/L	Data bit 2
10	DB3	H/L	Data bit 3
11	DB4	H/L	Data bit 4
12	DB5	H/L	Data bit 5
13	DB6	H/L	Data bit 6
14	DB7	H/L	Data bit 7
15	LED(+)		Anode of LED Backlight
16	LED(-)		Cathode of LED Backlight

Tabla 3.1: Distribución de pines de un LCD.

4. LISTADO DE MATERIALES Y EQUIPOS.

- ✔ Entrenador Lógico.
- ✔ 1 PIC 12F675.
- ✔ 1 PIC 16F628A
- ✔ Cables de par trenzado.

5. PROCEDIMIENTO.

✔Implementar el circuito de la figura 5.1 que muestra el transmisor y el receptor de la Práctica N°1.

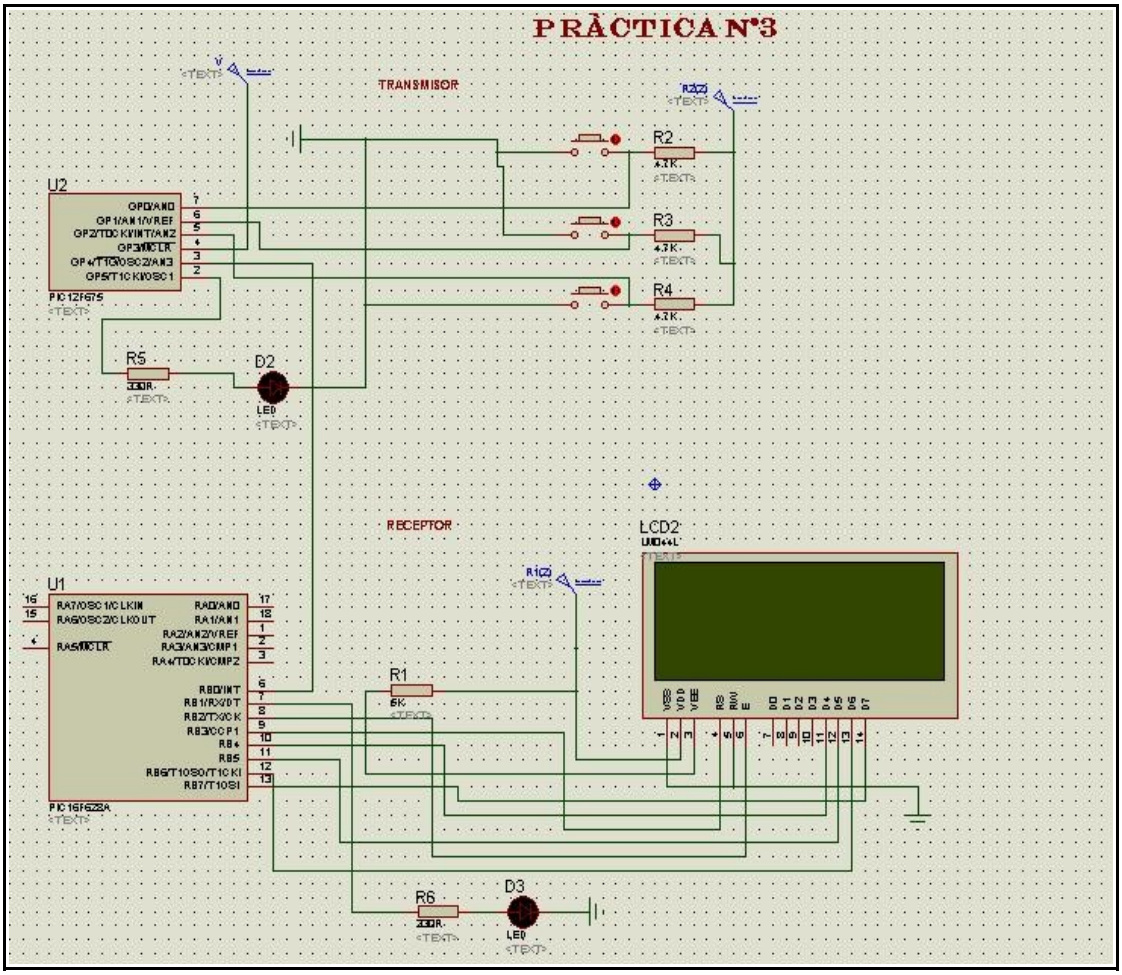


Figura 5.1: Circuito transmisor y receptor.

✔ Realizar la programación del transmisor y receptor utilizando el PIC 12F675.

PROGRAMACIÒN DEL PIC

TX

```
'*****************************************************************
'*  Name    : CLAVE INALAMBRICA TX
'*  Author  : EDISON JAVIER TERAN GUALOTO
'*  Notice  : Copyright (c) 2012 EDISON TERAN
'*          : All Rights Reserved
'*  Date    : 02/02/2012
'*  Version : 1.0
'*  Notes   :
'*          :
'*****************************************************************
;libreria para radio frecuencia, incluye los modos de comunicaciòn
include "modedefs.bas"
CMCON=%111
ANSEL=0
;ASIGNACION DE NOMBRES A LOS PUERTOS
P1   VAR GPIO.0
P2   VAR GPIO.1
P3   VAR GPIO.2
TX1  VAR GPIO.4
LED  VAR GPIO.5
;SUBRUTINA PARA TRANSMITIR
  TX:
    IF P1=0 THEN UNO
    IF P2=0 THEN ERROR
    if p3=0 then ERROR
  GOTO TX
;SUBRUTINA PARA EL ENVIO DE DATOS
 UNO:
;ANTIREBOTE
    GOSUB TECLA
    IF P1=0 THEN DOS
    IF P2=0 THEN ERROR
    IF P3=0 THEN ERROR
 GOTO UNO
 DOS:
;ANTIREBOTE
    GOSUB TECLA
    IF P1=0 THEN ERROR
    IF P2=0 THEN TRES
    IF P3=0 THEN ERROR
 GOTO DOS
 TRES:
    GOSUB TECLA
    IF P1=0 THEN ERROR
    IF P2=0 THEN ERROR
    IF P3=0 THEN TRANSMITIR
 GOTO TRES
 TRANSMITIR:
```

```
   GOSUB TECLA
   SEROUT TX1,N2400,["X"]
   HIGH LED
   PAUSE 120
   LOW LED
   pause 120
   HIGH LED
   PAUSE 120
   LOW LED
 GOTO TX
 ERROR:
   GOSUB TECLA
   SEROUT TX1,N2400,["Y"]
   HIGH LED
   PAUSE 1500
   LOW LED
 GOTO TX
;PROGRAMA PARA EL ANTIREBOTE
TECLA:
IF P1=0 THEN TECLA
IF P2=0 THEN TECLA
IF P3=0 THEN TECLA
PAUSE 100
RETURN
```

PROGRAMACIÒN DEL PIC

RX

```
'*****************************************************************
'*  Name    : CLAVE INALAMBRICA RX                                *
'*  Author  : EDISON JAVIER TERAN GUALOTO                         *
'*  Notice  : Copyright (c) 2012 EDISON TERAN                     *
'*          : All Rights Reserved                                 *
'*  Date    : 03/02/2012                                          *
'*  Version : 1.0                                                 *
'*  Notes   :                                                     *
'*          :                                                     *
'*****************************************************************
;libreria para radio frecuencia, incluye los modos de comunicaciòn
include "modedefs.bas"
RECEPTOR VAR PORTB.0
LED VAR PORTB.1
DEFINE LCD_LINES 4
DEFINE LCD_DREG PORTB ; define pines del LCD B4 a B7
DEFINE LCD_DBIT 4 ; empezando desde el Puerto B4 hasta el B7
DEFINE LCD_RSREG PORTB ;define el puerto B para conectar el bit RS
DEFINE LCD_RSBIT 3 ;este es el puerto B3
DEFINE LCD_EREG PORTB ;define el puerto B para conectar el bit Enable
DEFINE LCD_EBIT 2 ;este es el puerto B2
PAUSE 200 ;retardo para esperar que funcione el LCD
x VAR BYTE ;crear la variable x de 255
abc VAR BYTE ;crear la variable abc de 255
DATOS VAR BYTE

INICIO:
   LCDOUT $FE,1 ;limpiar pantalla
   LCDOUT,$FE,$80
```

```
    LCDOUT "  CLAVE INALAMBRICA"
    LCDOUT,$FE,$C0
    LCDOUT "  ================"
  SERIN receptor,N2400,DATOS
  IF DATOS="X" THEN ABRIR
  IF DATOS="Y" THEN ERROR
GOTO INICIO

ABRIR :
    LCDOUT,$FE,$94
    LCDOUT "ABRIENDO"
    LCDOUT $FE,$A0
    FOR X=0 TO 5
    LOOKUP x,["......."],abc ;tomar caracter por caracter y guardar en abc
    LCDOUT, abc ;sacar en LCD el contenido de abc
    PAUSE 400 ;esperar 400 mls
    NEXT ;siguiente repetición
    HIGH LED
    PAUSE 1000
    LOW LED
    LCDOUT,$FE,$D8
    LCDOUT "OK"
    PAUSE 1500
GOTO INICIO

ERROR:
    LCDOUT,$FE,$94
    LCDOUT "ERROR"
    LCDOUT,$FE,$D4
    LCDOUT "CLAVE INCORRECTA"
    PAUSE 1500
GOTO INICIO
```

✔ Cargar los programas en los PIC.

Los PIC se graban utilizando el software winpic800.

✔ Comprobar el funcionamiento del circuito implementado.

6. DIAGRAMAS Y FIGURAS.

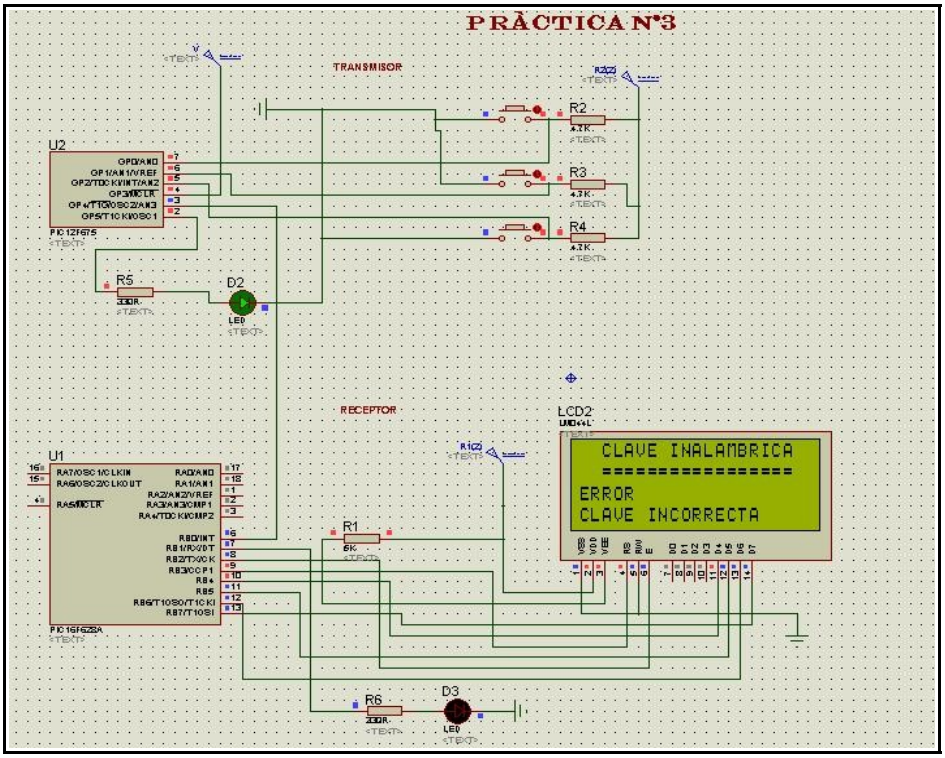


Figura 6.1: Práctica N°3, clave incorrecta.

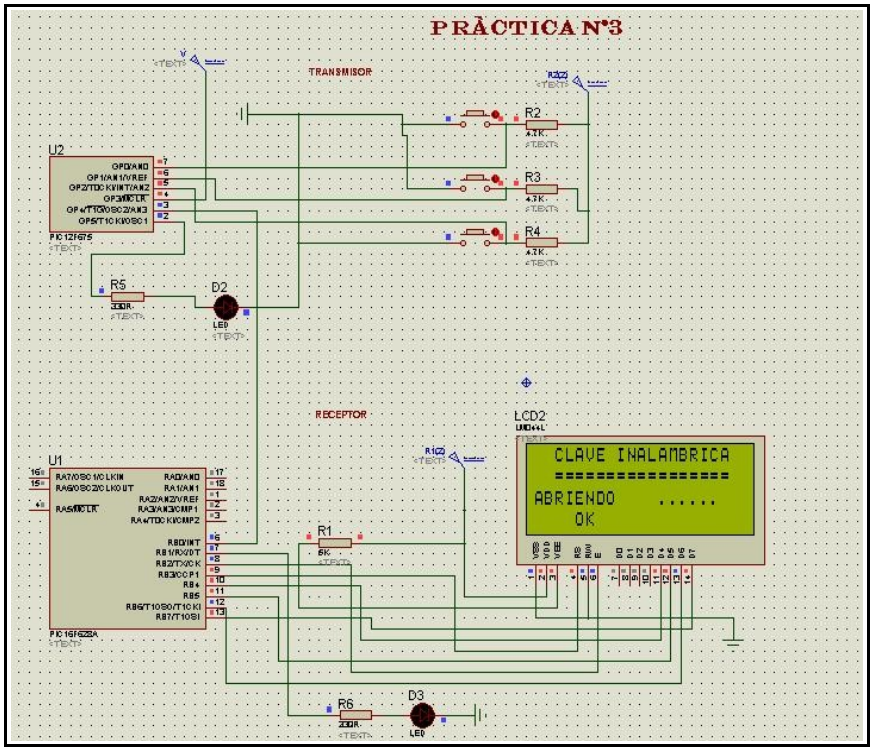


Figura 6.2: Práctica N°3, clave correcta.

7. TABULACIÓN Y RESULTADOS.

CLAVE	ESTADO
P1,P1,P2,P3	CORECTO
XXXX	INCORRECTO

Tabla 7.1: Clave inalámbrica.

8. CONCLUSIONES.

- El circuito de clave inalámbrica se la puede programar con uno o mas pulsadores y su funcionamiento no varía.

- El Entrenador Lógico trabaja con normalidad y precisión con la práctica planteada.

9. BIBLIOGRAFÍA.

- jmnlab.com/lcd/lcd.html

- http://www.jakeselectronics.net/pro-simplelcd.php

10. ANEXOS.

PRÁCTICA N° 4
J2ME APLICACIÓN EJECUTABLE PARA EL CELULAR

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: J2ME aplicación ejecutable para el celular.

2. OBJETIVOS

2.1. OBETIVO GENERAL

Diseñar una interfaz gráfica ejecutable para teléfonos celulares con sistema operativo Java.

2.2. OBJETIVOS ESPECÍFICOS

- Estudiar los comandos necesarios en la programación de J2ME con Netbeans.
- Instalar el archivo ejecutable en un teléfono celular.

3. MARCO TEÓRICO

Java es un lenguaje de programación creado por la empresa norteamericana Sun Microsystems a mediados de los años 90's. Actualmente es un lenguaje muy extendido por el mundo entero, más que lenguaje es una serie de tecnologías que compiten con los grandes imperios comerciales como Microsoft entre otros. Sun dividió el lenguaje en varias versiones para diferentes tipos de aplicaciones, estas versiones son las siguientes:

- Plataforma de Java Edición Empresarial (java Platform EE)
- Plataforma de Java Edición Estandar (java Platform SE)

- Plataforma de Java Edición Micro (java Platform ME)

Aquí se usó única y exclusivamente la plataforma de Java ME. Esta plataforma esta diseñada para crear aplicaciones Java principalmente en:

- Teléfonos celulares.
- Agendas electrónicas o también llamadas PDA's Personal Digital Assistant (Asistente Digital Personal).

En la actualidad existen entornos integrados de desarrollo (IDE) para programar en casi todos los lenguajes, con un IDE el proceso de programación se puede realizar de una forma rápida, ya que con estos es posible:

- Editar el código fuente, es decir, escribir el programa deseado.
- Compilar el código fuente para saber si existen errores de sintaxis.
- Depurar el programa, en caso de que el programa tenga una ejecución que no es la correcta.
- Ejecutar programa.

Java no es la excepción, existen varios IDEs para programar en este lenguaje, uno de ellos es el NetBeans que es un producto gratuito y sin restricciones de uso que goza de una gran popularidad a lo largo del mundo y que su patrocinador oficial es Sun Microsystem, los creadores de Java.

NetBeans, está diseñado a base de componentes llamados módulos, un módulo de NetBeans permite editar, compilar, depurar y ejecutar programas para teléfonos celulares y PDAs.

NetBeans es solo el IDE, para su funcionamiento requiere el compilador de Java y todas las herramientas de este lenguaje. Entonces para trabajar con NetBeans primero hay que instalar el compilador de java y todos sus archivos asociados.

Software necesario para programar teléfonos celulares

1. Java SE Development Kit (JDK 6)

Este software contiene todas las herramientas para desarrollar aplicaciones de escritorio para java y es requisito indispensable para NetBeans. El programa se puede bajar desde la página de Sun, la dirección es la siguiente:

http://java.sun.com/javase/downloads/index.jsp

Ahí se elige el JDK 6 Update 13.

4. LISTADO DE MATERIALES Y EQUIPOS.

- ✔ Entrenador Lógico.
- ✔ 1 Teléfono celular.
- ✔ NetBeans.
- ✔ Cable de datos del celular.

5. PROCEDIMIENTO.

✔Para empezar con la programación hay que tener un diseño de la interfaz gráfica establecido, para este proyecto se presenta una portada con los datos del autor con un fondo del escudo de la Universidad Israel, la imagen de fondo es la que se presenta en la figura 5.1.

Figura 5.1: Imagen de fondo de la aplicación.

✔ Realizar la programación para el teléfono celular.

PROGRAMACIÒN TELÈFONO CELULAR

MIDLET INTERFAZ_GRÀFICA

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class INTERFAZ_GRAFICA extends MIDlet implements CommandListener {
private Display display;
private SSCanvas screen;
public INTERFAZ_GRAFICA() {
display=Display.getDisplay(this);
screen=new SSCanvas();
```

```
screen.setCommandListener(this);
}
public void startApp() throws MIDletStateChangeException {
    display.setCurrent(screen);
}
public void pauseApp() {}
public void destroyApp(boolean unconditional) {}
public void commandAction(Command c, Displayable s) {

}
}
class SSCanvas extends Canvas {

public void paint(Graphics g) {
    Image img=null;
// Borrar la pantalla
g.setColor(255,255,255);
g.fillRect (0, 0, getWidth(), getHeight());

// Poner texto
Font fuente = Font.getFont (Font.FACE_PROPORTIONAL, Font.STYLE_BOLD,
Font.SIZE_MEDIUM);
g.setFont(fuente);
try {
     setFullScreenMode(true);
    img = Image.createImage("/logo.png");
} catch (Exception e) {
System.err.println("error: " + e);
}
g.drawImage (img, getWidth()/2,getHeight()/2, Graphics.HCENTER|Graphics.VCENTER);
g.setColor(0,0,0);
g.drawString("UNIVERSIDAD ISRAEL", getWidth()/2, 60,Graphics.BASELINE|
Graphics.HCENTER);
g.drawString("JAVIER TERÁN", getWidth()/2, 365,Graphics.BASELINE|Graphics.HCENTER);
g.drawString("INGENIERIA", getWidth()/2, 200,Graphics.BASELINE|Graphics.HCENTER);
g.drawString("ELECTRONICA", getWidth()/2, 225,Graphics.BASELINE|Graphics.HCENTER);
g.drawString("2012", getWidth()/2, 560,Graphics.BASELINE|Graphics.HCENTER);
}
}
```

✔Copiar el archivo ejecutable.

El archivo ejecutable se encuentra en la carpeta “dist” que esta dentro de la carpeta donde se genera el proyeto.

✔ Comprobar el funcionamiento del programa en el teléfono celular.

6. DIAGRAMAS Y FIGURAS.

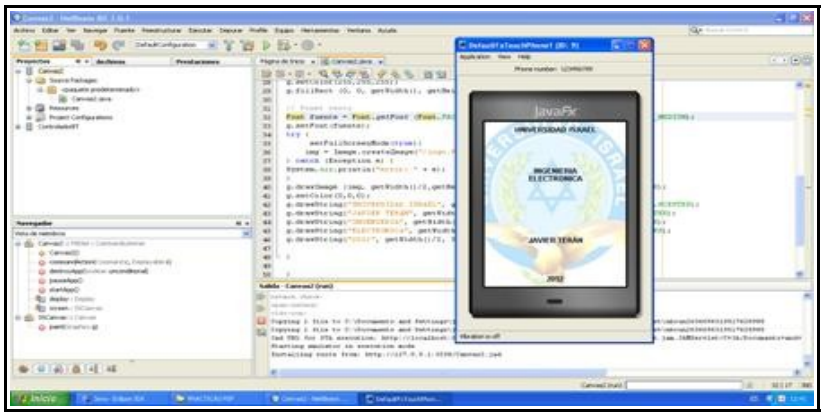


Figura 6.1: Práctica N°4, simulación de la interfaz gráfica.

7. TABULACIÓN Y RESULTADOS.

ARCHIVO	ESTADO
.JAR	CORECTO
.JAD	CORRECTO

Tabla 7.1: Funcionamiento del los archivos ejecutables.

8. CONCLUSIONES.

- NetBeans es un programa que genera archivos ejecutables para teléfonos celuares mediante programación JAVA.
- La programación es en base a clases y objetos construidos dentro de un mismo proyecto.
- NetBeans a través de su simulador muestra el programa ejecutado sin necesidad de instalarlo en el teléfono celular.

9. BIBLIOGRAFÍA.

- http://www.programacion-j2me.blogspot.com
- Programación de juegos para móviles con J2ME autor: Alberto Garcia.

10. ANEXOS.

PRÁCTICA N° 5
ANDROID APLICACIÓN EJECUTABLE PARA EL CELULAR

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Android aplicación ejecutable para el celular.

2. OBJETIVOS

2.1. OBETIVO GENERAL

Diseñar una interfaz gráfica ejecutable para teléfonos celulares con sistema operativo android.

2.2. OBJETIVOS ESPECÍFICOS

- Estudiar los comandos necesarios en la programación de Android.
- Instalar el archivo ejecutable en un teléfono celular.

3. MARCO TEÓRICO

Android es un sistema operativo móvil basado en Linux, que junto con aplicaciones middleware, está enfocado para ser utilizado en dispositivos móviles como teléfonos inteligentes, tablets, Google TV y otros dispositivos. Es desarrollado por la Open Handset Alliance, la cual es liderada por Google.

Fue desarrollado inicialmente por Android Inc., una firma comprada por Google en 2005. Es el principal producto de la Open Handset Alliance, un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio. Las unidades vendidas de teléfonos inteligentes con Android se ubican en el primer puesto en los Estados Unidos, en el segundo y tercer trimestres de 2010, con una cuota de mercado de 43,6% en el tercer trimestre.

Tiene una gran comunidad de desarrolladores escribiendo aplicaciones para extender la funcionalidad de los dispositivos. A la fecha, se han sobrepasado las 400.000 aplicaciones (de las cuales, dos tercios son gratuitas) disponibles para la tienda de aplicaciones oficial de Android: Android Market, sin tener en cuenta aplicaciones de otras tiendas no oficiales para Android, como la App Store de Amazon o la tienda de aplicaciones Samsung Apps de Samsung. Android Market es la tienda de aplicaciones en línea administrada por Google, aunque existe la posibilidad de obtener software externamente. Los programas están escritos en el lenguaje de programación Java. No obstante, no es un sistema operativo libre de malware, aunque la mayoría de ello es descargado de sitios de terceros.

Características de Android

Android se podría decir que es hijo de Linux, que no es otra cosa que otro sistema operativo diseñado para ordenadores. La principal característica de Linux, heredada por Android, radica en el sistema abierto de programación, es decir, contiene un código libre, que permite a quien desee transformarlo, adaptarlo o modificarlo, dependiendo de sus necesidades.

Principales ventajas de Android.

Al ser un código libre o abierto, la modificación, mejora y solución de problemas es más rápida y eficaz, ya que todo el mundo puede acceder al mismo, al contrario que sucede con otros sistemas operativos. Este último caso exige estar pendiente que la compañía o empresa responsable del sistema operativo del celular en cuestión, realice las mejoras o reparaciones necesarias, que en algunos casos pueden demorarse incluso algunos meses.

Móviles mas económicos.

Las especificaciones concretas de Android, revierten en una serie de beneficios al comprar un celular con este sistema operativo.

Android, al ser un sistema basado en Linux, abierto y libre, no conlleva pagos de licencias, y por consiguiente, abarata los costes, lo que finalmente se traduce en unos móviles con un precio de venta inferior.

- Ha sido desarrollado principalmente para smartphones de gama media-alta y con pantalla táctil, y ofrece rapidez de reacción y una gran exactitud en el

manejo del teclado virtual.

Android es un sistema operativo en continua evolución y mejora, lo que significa que muchas de las ventajas que aquí se exponen irán rápidamente en aumento:

- Resuelve cualquier web en flash, lo que significa que es posible ver vídeos y acceder a juegos normalmente.
- Aplicaciones en Android Market: contiene miles de aplicaciones y no deja de crecer con nuevos contenidos para el celular, y no tiene restricciones.
- Android se instala en la mayoría de marcas y operadoras, al contrario que otros sistemas operativos, y ofrece al usuario la posibilidad de elegir el móvil que más le guste.

4. LISTADO DE MATERIALES Y EQUIPOS.

- ✔ Entrenador Lógico.
- ✔ 1 Teléfono celular.
- ✔ Eclipse.
- ✔ Cable de datos del celular.

5. PROCEDIMIENTO.

✔La interfaz gráfica se presenta con los datos del autor.

Figura 5.1: Imagen de fondo de la aplicación.

✔ Realizar la programación para el teléfono celular.

PROGRAMACIÒN TELÈFONO CELULAR

```
package com.example.android.BluetoothChat;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int app_icon=0x7f020000;
    }
    public static final class id {
        public static final int button_scan=0x7f060006;
        public static final int button_send=0x7f060009;
        public static final int discoverable=0x7f060016;
        public static final int edit_text_out=0x7f060008;
        public static final int in=0x7f060007;
        public static final int insecure_connect_scan=0x7f060015;
        public static final int new_devices=0x7f060005;
        public static final int paired_devices=0x7f060003;
        public static final int secure_connect_scan=0x7f060014;
        public static final int textView1=0x7f06000a;
        public static final int textView2=0x7f06000c;
        public static final int textView3=0x7f06000e;
        public static final int textView4=0x7f060010;
        public static final int textView5=0x7f060012;
        public static final int title_left_text=0x7f060000;
        public static final int title_new_devices=0x7f060004;
        public static final int title_paired_devices=0x7f060002;
        public static final int title_right_text=0x7f060001;
        public static final int toggleButton1=0x7f06000b;
        public static final int toggleButton2=0x7f06000d;
        public static final int toggleButton3=0x7f06000f;
        public static final int toggleButton4=0x7f060011;
        public static final int toggleButton5=0x7f060013;
    }
    public static final class layout {
        public static final int custom_title=0x7f030000;
        public static final int device_list=0x7f030001;
        public static final int device_name=0x7f030002;
        public static final int main=0x7f030003;
        public static final int message=0x7f030004;
    }
    public static final class menu {
        public static final int option_menu=0x7f050000;
    }
    public static final class string {
        public static final int Interruptor1=0x7f040013;
        public static final int Interruptor2=0x7f040014;
        public static final int Interruptor3=0x7f040015;
        public static final int Interruptor4=0x7f040016;
        public static final int Interruptor5=0x7f040017;
        public static final int app_name=0x7f040000;
```

```
        public static final int bt_not_enabled_leaving=0x7f040003;
        public static final int button_scan=0x7f04000d;
        public static final int discoverable=0x7f040010;
        public static final int insecure_connect=0x7f04000f;
        public static final int no1=0x7f040012;
        public static final int none_found=0x7f04000a;
        public static final int none_paired=0x7f040009;
        public static final int not_connected=0x7f040002;
        /**   DeviceListActivity
         */
        public static final int scanning=0x7f040007;
        /**  Options Menu
         */
        public static final int secure_connect=0x7f04000e;
        public static final int select_device=0x7f040008;
        /**   BluetoothChat
         */
        public static final int send=0x7f040001;
        public static final int si1=0x7f040011;
        public static final int title_connected_to=0x7f040005;
        public static final int title_connecting=0x7f040004;
        public static final int title_not_connected=0x7f040006;
        public static final int title_other_devices=0x7f04000c;
        public static final int title_paired_devices=0x7f04000b;
    }
}
```

✔Copiar el archivo ejecutable.

El archivo ejecutable se encuentra en la carpeta “dist” que esta dentro de la carpeta donde se genera el proyecto y es de extensión .APK.

✔ Comprobar el funcionamiento del programa en el teléfono celular.

6. DIAGRAMAS Y FIGURAS.

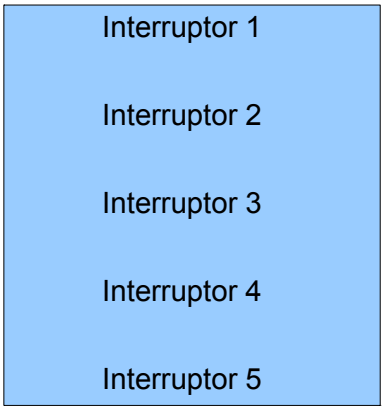


Figura 6.1: Práctica N°5, interfaz gráfica.

7. TABULACIÓN Y RESULTADOS.

ARCHIVO	ESTADO
.APK	CORECTO

Tabla 7.1: Funcionamiento del archivo ejecutable.

8. CONCLUSIONES.

- Eclipse es un programa que genera archivos ejecutables para teléfonos celulares mediante programación JAVA y su archivo ejecutable es .APK.

- La programación es en base a clases y objetos construidos dentro de un mismo proyecto.

- Eclipse a través de su simulador muestra el programa ejecutado sin necesidad de instalarlo en el teléfono celular.

9. BIBLIOGRAFÍA.

- http://es.wikipedia.org/wiki/Android

- http://alejandro-esparza-tudela.suite101.net/que-es-android-en-celula-res-a61379.

10. ANEXOS.

PRÁCTICA N° 6
CONTROL DE UN INTERUPTOR A TRAVÉS DE BT

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Control de un interruptor a través de BT .

2. OBJETIVOS

2.1. OBETIVO GENERAL

Diseñar una interfaz gráfica capaz de controlar un interruptor a través de bluetooth por medio de J2ME.

2.2. OBJETIVOS ESPECIFÍCOS

- Estudiar los comandos necesarios en la programación de J2ME con Netbeans para la comunicación inalámbrica.
- Comprobar el funcionamiento de la comunicación inalámbrica a través de bluetooth.

3. MARCO TEÓRICO

BLUETOOTH

La tecnología Bluetooth es una especificación abierta para la comunicación inalámbrica (WIRELESS) de datos y voz. Está basada en un enlace de radio de bajo costo y corto alcance, implementado en un circuito integrado de 9 x 9 mm, proporcionando conexiones instantáneas (ad hoc) para entornos de comunicaciones tanto móviles como estáticos. En definitiva, Bluetooth pretende ser una especificación global para la conectividad inalámbrica.

El principal objetivo de esta tecnología, es la posibilidad de reemplazar los muchos

cables propietarios que conectan unos dispositivos con otros por medio de un enlace radio universal de corto alcance. Por ejemplo, la tecnología de radio Bluetooth implementada en el teléfono celular y en el ordenador portátil reemplazaría el molesto cable utilizado hoy en día para conectar ambos aparatos. Las impresoras, las agendas electrónicas, los PDA, los faxes, los teclados, los joysticks y prácticamente cualquier otro dispositivo digital son susceptibles de formar parte de un sistema Bluetooth.

Pero más allá de reemplazar, los incómodos cables, la tecnología Bluetooth ofrece un puente a las redes de datos existentes, una interfaz con el exterior y un mecanismo para formar en el momento, pequeños grupos de dispositivos conectados entre sí de forma privada fuera de cualquier estructura fija de red.

Integrado en un pequeño transmisor de radiofrecuencia que permite conectar entre sí todo tipo de dispositivos electrónicos (teléfonos, ordenadores, impresoras, faxes, etc) situados dentro de un radio limitado de 10 metros (ampliable a 100, aunque con mayor distorsión) sin necesidad de utilizar cables.

El transmisor está integrado en un pequeño microchip de 9x9 milímetros y opera en una frecuencia de banda global (2,4 GHz, utilizada en muchos países para usos médicos y científicos) que asegura la compatibilidad universal. Los dispositivos que incorporan Bluetooth se reconocen y se hablan de la misma forma que lo hace un ordenador con su impresora. El canal permanece abierto y no requiere la intervención directa y constante del usuario cada vez que se quiere enviar algo.

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

Las diferentes partes del sistema Bluetooth son:

- Una unidad de radio
- Una unidad de control del enlace
- Gestión del enlace
- Funciones software

4. LISTADO DE MATERIALES Y EQUIPOS.

- ✔ Entrenador Lógico.
- ✔ 1 Teléfono celular.
- ✔ NetBeans.
- ✔ Cable de datos del celular.

5. PROCEDIMIENTO.

✔Para empezar con la programación hay que tener un diseño de la interfaz gráfica establecido, para este proyecto se presenta una portada con los datos del autor con un fondo del escudo de la Universidad Israel, la imagen de fondo es la que se presenta en la figura 5.1.

Figura 5.1: Imagen de fondo de la aplicación.

✔ Realizar la programación para el teléfono celular.

PROGRAMACIÒN TELÈFONO CELULAR

MIDLET CONTROL_BT

```
import javax.bluetooth.ServiceRecord;
import javax.microedition.midlet.*;
/**
 * @author JAVTER
public class ControlBT extends MIDlet implements Runnable {
    public void startApp() {
        new Portada(this);
        new Thread(this).start();
    }
    public void pauseApp() {
    }
    public void destroyApp(boolean unconditional) {
    }
    public void exitMidlet() {
```

```
        destroyApp(true);
        notifyDestroyed();
    }
    public void run() {
        try {
            Thread.sleep(3000);
        } catch (InterruptedException ex) {
        }
          Rele bluetooth = new Rele(this);
        bluetooth.makeConnections
       ("btspp://00195DEE405B:1;authenticate=false;encrypt=false;master=false",
       ServiceRecord.NOAUTHENTICATE_NOENCRYPT);
    }
}
```

CLASE PORTADA

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.MIDlet;
public class Portada {
    private Display display;
    private MyCanvas canvas = new MyCanvas();
    private MIDlet midlet;
    public Portada(MIDlet midletIn) {
        midlet =  midletIn;
        display = Display.getDisplay(midlet);
        display.setCurrent(canvas);
    }
    public Display getDisplay() {
        return display;
    }
}
class MyCanvas extends Canvas {
    private Image image = null;
    public MyCanvas() {
        try {
            image = Image.createImage("/PORTADA.PNG");
        } catch (Exception error) {
       Alert alert = new Alert("Failure", "Can't open image file.", null, null);
            alert.setTimeout(Alert.FOREVER);
        }
    }
    protected void paint(Graphics graphics) {
        graphics.setColor(255,255,255);//RGB
        graphics.fillRect (0, 0, getWidth(), getHeight());
        if (image != null) {
            graphics.drawImage(image, getWidth() / 2, getHeight() / 2, Graphics.
                    HCENTER | Graphics.VCENTER);
        }
        setFullScreenMode(true);
    }
}
```

CLASE RELE

```
import javax.microedition.lcdui.*;
import java.util.Vector;
import javax.bluetooth.LocalDevice;
```

```
public class Rele implements ConexionHandlerListener, CommandListener {
    private Alert alert;
    private Display display;
    private ControlBT mIDlet;
    private Command salir;
    private List listaOpciones;
    private Command atraz;
    private final Vector handlers;
    private volatile int numReceivedMessages = 0;
    private volatile int numSentMessages = 0;
    String receive = "";
    Image img;
    private int maxConnections;
    private int sendMessageId = 0;
    private String sendcommand;
    private String tipo="";
    public Rele(ControlBT mIDletIn) {
        init();
        mIDlet = mIDletIn;
        display = Display.getDisplay(mIDlet);
        display.setCurrent(listaOpciones);
        handlers = new Vector();
        String value =
            LocalDevice.getProperty(
                        "bluetooth.connected.devices.max");
        try
        {
            maxConnections = Integer.parseInt(value);
              }
        catch (NumberFormatException e)
        {
            maxConnections = 0;
        }
    }
    private void init() {
        salir = new Command("Salir", Command.EXIT, 2);
        atraz = new Command("Atrás", Command.BACK, 1);
        listaOpciones = new List("PRACTICA Nº6", List.IMPLICIT);
        listaOpciones.append("CONTROL", null);
        listaOpciones.addCommand(salir);
        listaOpciones.setCommandListener(this);
    }
    public void presentarAlerta(String resultado) {
        if (resultado.startsWith("Error:")) {
            alert = new Alert("Alerta", resultado, null, AlertType.WARNING);
        } else if (resultado.startsWith("Ok:")) {
            alert = new Alert("Alerta", resultado, null, AlertType.INFO);
        }
        alert.setTimeout(Alert.FOREVER);
        alert.addCommand(new Command("Salir", Command.EXIT, 2));
        display.setCurrent(alert, listaOpciones);
    }
    public void commandAction(Command c, Displayable d) {
        if (c == listaOpciones.SELECT_COMMAND) {
       String label = listaOpciones.getString(listaOpciones.getSelectedIndex());
```

```
            //System.out.println(label);
            if (label.equals("CONTROL")) {
                tipo="1";
                listaOpciones.deleteAll();
                listaOpciones.append("ACTIVAR", null);
                listaOpciones.append("DESACTIVAR", null);
                listaOpciones.addCommand(atraz);
            } else if (label.equals("ACTIVAR")) {
                send("si"+tipo);
            } else if (label.equals("DESACTIVAR")) {
                send("no"+tipo);
            }
        } else if (c == atraz) {
            listaOpciones.deleteAll();
            listaOpciones.removeCommand(atraz);
            listaOpciones.append("CONTROL", null);
            listaOpciones.addCommand(salir);
            listaOpciones.setCommandListener(this);
        } else if (c == salir) {
            ((ControlBT) mIDlet).exitMidlet();
        }
    }
    public void handleOpen(ConexionHandler handler) {
        handlers.addElement(handler);
        // for the first open connection
        if (handlers.size() == 1) {
//            removeCommand(searchCommand);
//            removeCommand(sendCommand);
//            addCommand(sendCommand);
        }
        // Remove the 'Add connection' command
        // when the device already has open the
        // maximum number of connections it can
        // support.
        if (handlers.size() >= maxConnections) {
//            removeCommand(addConnectionCommand);
        }

        //texto2="Connection opened";
        String str = Integer.toString(handlers.size());
        //texto1=str;
        //fondoambiente=1;
        listaOpciones.append("Conectado", null);
    }

    public void handleOpenError(
            ConexionHandler handler,
            String errorMessage) {
        //midlet.noencontrado("El dispositivo no esta dentro del area de servicio");
        listaOpciones.append("Dispositivo no encontrado",null);
    }

    public void handleReceivedMessage(
            ConexionHandler handler,
            byte[] messageBytes) {
```

```
        numReceivedMessages++;

        String message = new String(messageBytes);
        receive = message;

        // texto2="# messages read: " + numReceivedMessages + " " +
        //     "sent: " + numSentMessages;

        if (message.substring(4, 5).equals("0")) {
           //   receive="off";
           //    foco.selFondo(0);
        }

        // Broadcast message to all clients
//        for (int i=0; i < handlers.size(); i++)
//        {
//           ConexionHandler h =
//              (ConexionHandler) handlers.elementAt(i);

//           Integer id = new Integer(sendMessageId++);
//           try
//           {
//              h.queueMessageForSending(id, messageBytes);
//           }
//           catch (IllegalArgumentException e)
//           {
//              String errorMessage =
//                 "IllegalArgumentException while trying to " +
//                 "send message: " + e.getMessage();
//              handleError(handler, errorMessage);
//           }
//        }
        listaOpciones.append("recibiendo datos", null);
    }

    public void handleQueuedMessageWasSent(
           ConexionHandler handler,
           Integer id) {
        numSentMessages++;

        //texto2="# messages read: " +numReceivedMessages + " " + "sent: " + numSentMessages;
    }

    public void handleClose(ConexionHandler handler) {
        removeHandler(handler);
        if (handlers.size() == 0) {
//           removeCommand(sendCommand);
//           addCommand(searchCommand);
        }

        // If the number of currently open connections
        // drops below the maximum number that this
        // device could have open, restore
        // 'Add connection' to the screen commands.
        if (handlers.size() < maxConnections) {
```

```
//            removeCommand(addConnectionCommand);
//            addCommand(addConnectionCommand);
        }
        //texto2="Connection closed";
        closeAll();
    }
    public void handleErrorClose(ConexionHandler handler,
            String errorMessage) {
        removeHandler(handler);
        if (handlers.size() == 0) {
//            removeCommand(sendCommand);
//            addCommand(searchCommand);
        }

     //   texto2 = "Error (close): '" + errorMessage + "'";
    }
    public void handleError(ConexionHandler handler,
            String errorMessage) {
        //texto2 = "Error: '" + errorMessage + "'";
    }
    public void makeConnections(String URL, int security) {
        ConexionHandler newHandler =
                new ConexionHandler(
                this,
                URL,
                security);
        newHandler.start(); // start reader & writer
    }
    private void removeHandler(ConexionHandler handler) {
        if (handlers.contains(handler)) {
            handlers.removeElement(handler);
            String str = Integer.toString(handlers.size());
          //  texto1 = str;
            if (handlers.size() == 0) {
            }
        }
    }

    private void send(String sendData) {
        Integer id = new Integer(sendMessageId++);

        for (int i = 0; i < handlers.size(); i++) {
            ConexionHandler handler =
                    (ConexionHandler) handlers.elementAt(i);
            try {
                handler.queueMessageForSending(
                        id,
                        sendData.getBytes());
            //    texto1 = "Queued a send message request";
            } catch (IllegalArgumentException e) {
                // Message length longer than
                // ServerConnectionHandler.MAX_MESSAGE_LENGTH

                String errorMessage =
                        "IllegalArgumentException while trying "
```

```
                    + "to send a message: " + e.getMessage();
                //handleError(handler, errorMessage);
            }
        }
    }

    void closeAll() {
        for (int i = 0; i < handlers.size(); i++) {
            ConexionHandler handler =
                    (ConexionHandler) handlers.elementAt(i);
            handler.close();
            removeHandler(handler);
        }
    }
}
```

CLASE CONEXIÓN HANDLER

```
import java.io.InputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.util.Hashtable;
import java.util.Enumeration;
import javax.bluetooth.ServiceRecord;
import javax.microedition.io.Connector;
import javax.microedition.io.StreamConnection;
import javax.microedition.io.StreamConnectionNotifier;
public class ConexionHandler
    implements Runnable
{
    private final static byte ZERO = (byte) '0';
    private final static int LENGTH_MAX_DIGITS = 5;

    // this is an arbitrarily chosen value:
    private final static int MAX_MESSAGE_LENGTH =
                        65536 - LENGTH_MAX_DIGITS;

    //private final ServiceRecord serviceRecord;
    private final int requiredSecurity;
    private final ConexionHandlerListener listener;
    private final Hashtable sendMessages = new Hashtable();
    private StreamConnection connection;
    private OutputStream out;
    private InputStream in;
    private volatile boolean aborting;
    private Writer writer;
    private String url;
    public ConexionHandler(
            ConexionHandlerListener listener,
            String URL,
            int requiredSecurity)
    {
        this.listener = listener;
        this.url=URL;
        //this.serviceRecord = serviceRecord;
        this.requiredSecurity = requiredSecurity;
        aborting = false;
```

```
        connection = null;
        out = null;
        in = null;
        listener = null;

        // the caller must call method 'start'
        // to start the reader and writer
    }

//    public ServiceRecord getServiceRecord()
//    {
//        return serviceRecord;
//    }
    public synchronized void start()
    {
        Thread thread = new Thread(this);
        thread.start();
    }
    public void close()
    {
        if (!aborting)
        {
            synchronized(this)
            {
                aborting = true;
            }
            synchronized(sendMessages)
            {
                sendMessages.notify();
            }

            if (out != null)
            {
                try
                {
                    out.close();
                    synchronized (this)
                    {
                        out = null;
                    }
                }
                catch(IOException e)
                {
                    // there is nothing we can do: ignore it
                }
            }

            if (in != null)
            {
                try
                {
                    in.close();
                    synchronized (this)
                    {
```

```
                in = null;
            }
        }
        catch(IOException e)
        {
            // there is nothing we can do: ignore it
        }
    }

    if (connection != null)
    {
        try
        {
            connection.close();
            synchronized (this)
            {
                connection = null;
            }
        }
        catch (IOException e)
        {
            // there is nothing we can do: ignore it
        }
    }
  }
}
public void queueMessageForSending(Integer id, byte[] data)
{
  if (data.length > MAX_MESSAGE_LENGTH)
  {
    throw new IllegalArgumentException(
                "Message too long: limit is " +
                MAX_MESSAGE_LENGTH + " bytes");
  }
  synchronized(sendMessages)
  {
    sendMessages.put(id, data);
    sendMessages.notify();
  }
}
public void run()
{
  // the reader
  // 1. open the connection and streams, start the writer
  //String url = null;
  try
  {
    // 'must be master': false
    //url = serviceRecord.getConnectionURL(
    //                requiredSecurity,
    //                false);
    connection = (StreamConnection) Connector.open(url);
    in = connection.openInputStream();
    out = connection.openOutputStream();
```

```
//          LogScreen.log("Opened connection & streams to: '" +
//                  url + "'\n");
            // start the writer
            Writer writer = new Writer(this);
            Thread writeThread = new Thread(writer);
            writeThread.start();
//          LogScreen.log("Started a reader & writer for: '" +
//                  url + "'\n");
            // open succeeded, inform listener
            listener.handleOpen(this);
        }
        catch(IOException e)
        {
            // open failed, close any connections/streams, and
            // inform listener that the open failed

 //         LogScreen.log("Failed to open " +
 //                 "connection or streams for '" +
 //                  url + "' , Error: " +
 //                  e.getMessage());
            close();
            listener.handleOpenError(
                    this,
                    "IOException: '" + e.getMessage() + "'");
            return;
        }
        catch (SecurityException e)
        {
            // open failed, close any connections/streams, and
            // inform listener that the open failed

 //         LogScreen.log("Failed to open " +
 //                 "connection or streams for '" +
 //                  url + "' , Error: " +
 //                  e.getMessage());
            close();

            listener.handleOpenError(
                    this,
                    "SecurityException: '" + e.getMessage() + "'");
            return;
        }
        // 2. wait to receive and read messages
        while (!aborting)
        {
            int length = 0;
            try
            {
                byte[] lengthBuf = new byte[LENGTH_MAX_DIGITS];
                readFully(in, lengthBuf);
                length = readLength(lengthBuf);
                byte[] temp = new byte[length];
                readFully(in, temp);

                listener.handleReceivedMessage(this, temp);
```

```
        }
        catch (IOException e)
        {
            close();
            if (length == 0)
            {
              listener.handleClose(this);
            }
            else
            {
              // we were in the middle of reading...
              listener.handleErrorClose(this, e.getMessage());
            }
        }
    }
}

private static void readFully(InputStream in, byte[] buffer)
    throws IOException
{
    int bytesRead = 0;

    while (bytesRead < buffer.length)
    {
        int count = in.read(buffer,
                            bytesRead,
                            buffer.length - bytesRead);
        if (count == -1)
        {
            throw new IOException("Input stream closed");
        }
        bytesRead += count;
    }
}
private static int readLength(byte[] buffer)
{
    int value = 0;

    for (int i = 0; i < LENGTH_MAX_DIGITS; ++i)
    {
        value *= 10;
        value += buffer[i] - ZERO;
    }
    return value;
}
private void sendMessage(OutputStream out, byte[] data)
    throws IOException
{
    int i;
    byte[] buf=new byte[2+data.length];
    for (i=0;i<data.length;i++){
        buf[i]=data[i];
    }
    buf[i]=13;
    buf[i+1]=10;
```

```
    out.write(buf);
    out.flush();
}
private static void writeLength(int value, byte[] buffer)
{
    for (int i = LENGTH_MAX_DIGITS -1; i >= 0; --i)
    {
        buffer[i] = (byte) (ZERO + value % 10);
        value = value / 10;
    }
}
private class Writer
    implements Runnable
{
    private final ConexionHandler handler;
    Writer(ConexionHandler handler)
    {
        this.handler = handler;
    }
    public void run()
    {
        while (!aborting)
        {
            synchronized(sendMessages)
            {
                Enumeration e = sendMessages.keys();
                if (e.hasMoreElements())
                {
                    // send any pending messages
                    Integer id = (Integer) e.nextElement();
                    byte[] sendData =
                          (byte[]) sendMessages.get(id);
                    try
                    {
                        sendMessage(out, sendData);

                        // remove sent message from queue
                        sendMessages.remove(id);

                        // inform listener that it was sent
                        listener.handleQueuedMessageWasSent(
                                handler,
                                id);
                    }
                    catch (IOException ex)
                    {
                        close(); // stop the networking thread

                        // inform that we got an error close
                        listener.handleErrorClose(handler,
                                        ex.getMessage());
                    }
                }
```

```
                if (sendMessages.isEmpty())
                {
                    try
                    {
                        sendMessages.wait();
                    }
                    catch (InterruptedException ex)
                    {
                        // this can't happen in MIDP: ignore it
                    }
                }
            }
        }
    }
}
```

CLASE CONEXIÓN HANDLER LISTENER

```
public interface ConexionHandlerListener
{
    // The handler's open succeeded. It can now be used for sending
    // and receiving messages.
    public void handleOpen(ConexionHandler handler);

    // The hadler's open failed. It has closed any connections or
    // streams that were open. The handler should not be used anymore,
    // and should be discarded.
    public void handleOpenError(ConexionHandler handler,
                                String errorMessage);
    public void handleQueuedMessageWasSent(ConexionHandler handler, Integer id);
    // The handler got an inbound message.
    public void handleReceivedMessage(
                ConexionHandler handler,
                byte[] messageBytes);
    // A message that had been previously queued for sending
    // (identified by id) by the handler, has been sent successfully.
    // The handler has closed its connections and streams.
    // The handler should not be used anymore, and should be discarded.
    // Only handlers which have previously called 'handleOpen' may
    // call 'handleClose', and only just once.
    public void handleClose(ConexionHandler handler);
    // An error related to the handler occurred. The handler
    // has closed the connection, and the handler should no
    // longer be used.
    public void handleErrorClose(ConexionHandler handler,
                                 String errorMessage);

}
```

✔Copiar el archivo ejecutable.

El archivo ejecutable se encuentra en la carpeta “dist” que esta dentro de la carpeta donde se genera el proyeto.

✔ Comprobar el funcionamiento del programa en el teléfono celular.

6. DIAGRAMAS Y FIGURAS.

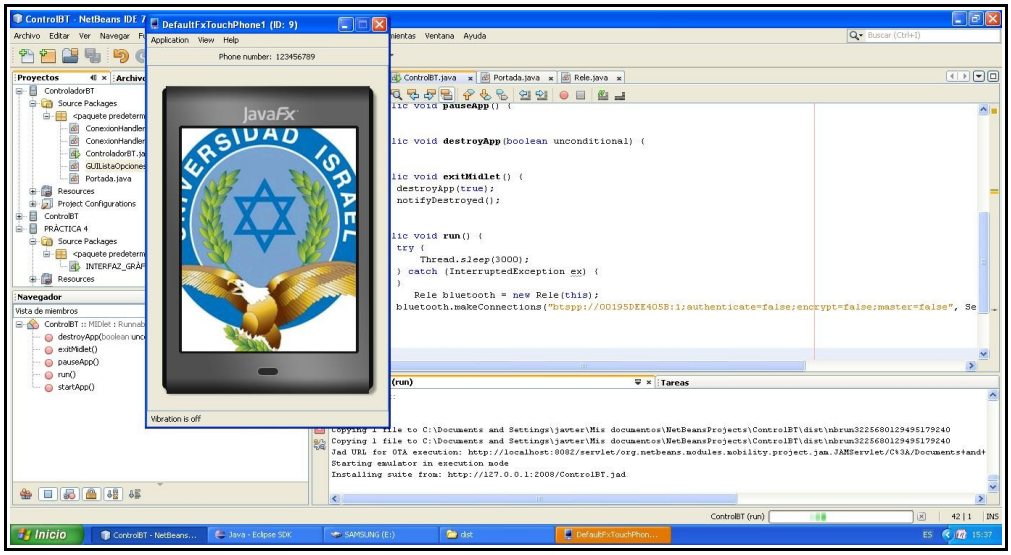


Figura 6.1: Práctica N°6, simulación de la interfaz gráfica, pantalla1.

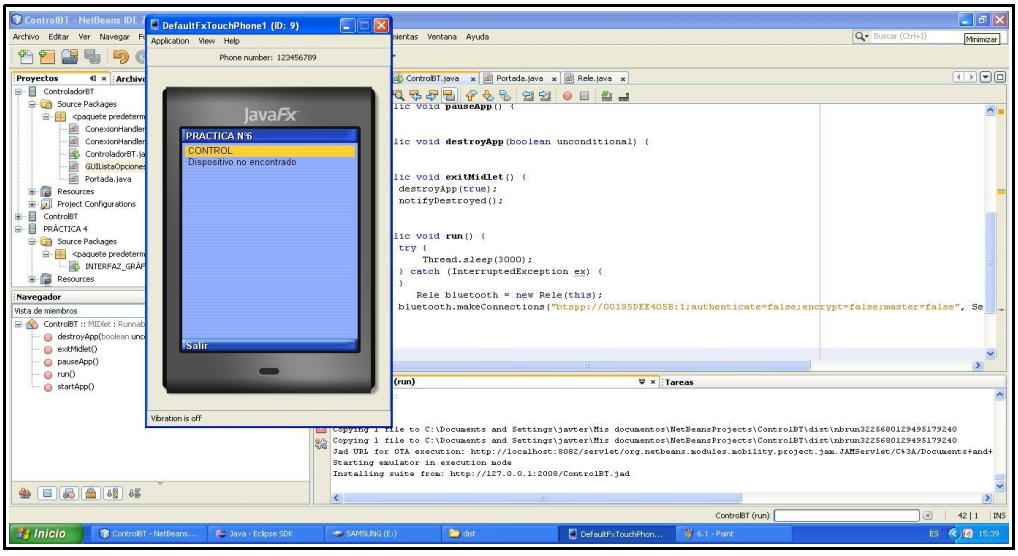


Figura 6.2: Práctica N°6, simulación de la interfaz gráfica, pantalla2.

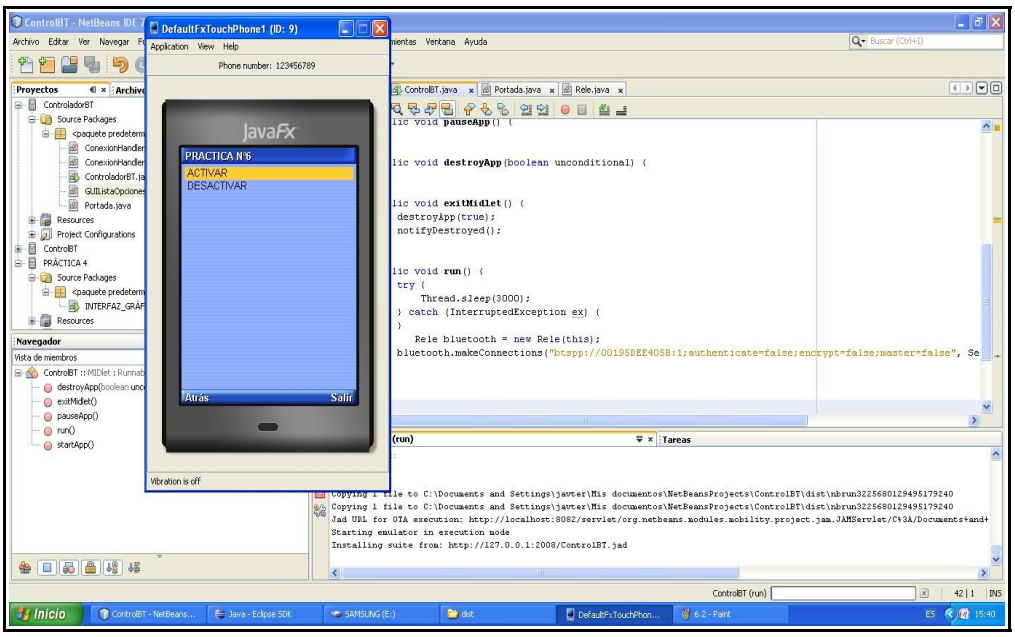


Figura 6.3: Práctica N°6, simulación de la interfaz gráfica, pantalla3.

7. TABULACIÓN Y RESULTADOS.

ARCHIVO	ESTADO
ACTIVADO	CORECTO
DESACTIVADO	CORRECTO

Tabla 7.1: Funcionamiento del archivo ejecutable.

8. CONCLUSIONES.

- El módulo bluetooth se conecta a través del celular y su alcance máximo es de 10 metros en linea de vista.
- Para poder realizar la comunicación se utiliza la clase javax.bluetooth.LocalDevice
- El Entrenador Lógico provee de 3,3 VDC al módulo bluetooth para su correcto funcionamiento.

9. BIBLIOGRAFÍA.

- http://sourceforge.net/projects/bluetoothled/files/

- http://www.google.com/search?client=ubuntu&channel=fs&q=comunicacion+bt+j2me&ie=utf-8&oe=utf-8#hl=es&client=ubuntu&hs=lcr&channel=fs&sa=X&ei=RQRAT_uxLMavg-wes2YSbCA&sqi=2&ved=0CBgQBSgA&q=comandos+de+comunicaci%C3%B3n+en+netbeans&spell=1&bav=on.2,or.r_gc.r_pw.,cf.osb&fp=141a0420e47bc3e0&biw=1368&bih=606

10. ANEXOS.

PRÁCTICA N° 7
MINI ROBOT CONTROLADO POR RF

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Minirobot controlado por RF.

2. OBJETIVOS

2.1. OBETIVO GENERAL

Diseñar e implementar un mini robot controlado por radio frecuencia.

2.2. OBJETIVOS ESPECÍFICOS

- Aplicar los conocimientos adquiridos en RF durante las prácticas anteriores.
- Comprobar el funcionamiento de la comunicación inalámbrica entre el Entrenador Lógico y el Minirobot.

3. MARCO TEÓRICO

Un Puente H o Puente en H es un circuito electrónico que permite a un motor eléctrico DC girar en ambos sentidos, avance y retroceso. Son ampliamente usados en robótica y como convertidores de potencia. Los puentes H están disponibles como circuitos integrados, pero también pueden construirse a partir de componentes discretos. La figura 3.1 muestra la estructura interna de un puente H.

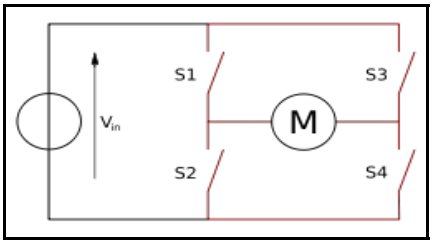


Figura 3.1: Estructura interna, Puente H.

Los 2 estados básicos del circuito.

El término "puente H" proviene de la típica representación gráfica del circuito. Un puente H se construye con 4 interruptores (mecánicos o mediante transistores). Cuando los interruptores S1 y S4 están cerrados (y S2 y S3 abiertos) se aplica una tensión positiva en el motor, haciéndolo girar en un sentido. Abriendo los interruptores S1 y S4 (y cerrando S2 y S3), el voltaje se invierte, permitiendo el giro en sentido inverso del motor.

Con la nomenclatura que se está usando, los interruptores S1 y S2 nunca podrán estar cerrados al mismo tiempo, porque esto cortocircuitaría la fuente de tensión. Lo mismo sucede con S3 y S4.

4. LISTADO DE MATERIALES Y EQUIPOS.

- Entrenador Lógico.
- Minirobot.
- Cables de par trenzado.

5. PROCEDIMIENTO.

- Establecer los movimientos del Minirobot.

El Minirobot esta diseñado para realizar los siguientes movimientos:

- Adelante.
- Atrás.
- Derecha.
- Izquierda.
- Parar.

- Realizar la programación para el TX.

PROGRAMACIÒN DEL PIC
TX

```
'*****************************************************************
'*  Name    : MINIROBOT_TX
'*  Author  : EDISON JAVIER TERAN GUALOTO
'*  Notice  : Copyright (c) 2012 EDISON TERAN
'*          : All Rights Reserved
'*  Date    : 23/02/2012
'*  Version : 1.0
'*  Notes   :
'*          :
'*****************************************************************
;libreria para radio frecuencia, incluye los modos de comunicaciòn
include "modedefs.bas"
CMCON=7
TX1 VAR PORTB.0       ;ASIGNAR VARIABLES A LOS PUERTOS
LED VAR PORTB.1
P1  VAR PORTA.0
P2  VAR PORTA.1
P3  VAR PORTA.2
P4  VAR PORTA.3
P5  VAR PORTA.4
DEFINE LCD_LINES 4
DEFINE LCD_DREG PORTB ; define pines del LCD B4 a B7
DEFINE LCD_DBIT 4 ; empezando desde el Puerto B4 hasta el B7
DEFINE LCD_RSREG PORTB ;define el puerto B para conectar el bit RS
DEFINE LCD_RSBIT 3 ;este es el puerto B3
DEFINE LCD_EREG PORTB ;define el puerto B para conectar el bit Enable
DEFINE LCD_EBIT 2 ;este es el puerto B2
PAUSE 200 ;retardo para esperar que funcione el LCD
   LCDOUT $FE,1 ;limpiar pantalla
   LCDOUT,$FE,$80
   LCDOUT "      MINIROBOT"
   LCDOUT,$FE,$C0
   LCDOUT "      ========="
   LCDOUT,$FE,$95
   LCDOUT "CONTROLADO POR RF"
;SUBRUTINA DE INICIO DETECTA EL PULSADOR PRESIONADO
INICIO:
      IF P1=0 THEN ADELANTE
      IF P2=0 THEN DERECHA
      IF P3=0 THEN ATRAS
      IF P4=0 THEN IZQUIERDA
      IF P5=0 THEN PARAR
GOTO INICIO
ADELANTE:   ;SI PRESIONA LA LETRA W
   GOSUB TECLA
   GOSUB LED1
   SEROUT TX1,N2400,["W"]
   GOSUB LCD
   LCDOUT $FE,$D4
   LCDOUT "ADELANTE"
```

```
GOTO INICIO
ATRAS:      ;SI PRESIONA LA LETRA X
   GOSUB TECLA
   GOSUB LED1
   SEROUT TX1,N2400,["X"]
   GOSUB LCD
   LCDOUT $FE,$D4
   LCDOUT "ATRAS"
GOTO INICIO
DERECHA:    ;SI PRESIONA LA LETRA Y
   GOSUB TECLA
   GOSUB LED1
   SEROUT TX1,N2400,["Y"]
   GOSUB LCD
   LCDOUT $FE,$D4
   LCDOUT "DERECHA"
GOTO INICIO
IZQUIERDA:  ;SI PRESIONA LA LETRA Z
   GOSUB TECLA
   GOSUB LED1
   SEROUT TX1,N2400,["Z"]
   GOSUB LCD
   LCDOUT $FE,$D4
   LCDOUT "IZQUIERDA"
GOTO INICIO
PARAR:      ;SI PRESIONA LA LETRA S
   GOSUB TECLA
   GOSUB LED1
   SEROUT TX1,N2400,["S"]
   GOSUB LCD
   LCDOUT $FE,$D4
   LCDOUT "PARAR"
GOTO INICIO
TECLA:     ;SUBRUTINA DE ANTIREBOTE
   IF P1=0 THEN TECLA
   IF P2=0 THEN TECLA
   IF P3=0 THEN TECLA
   IF P4=0 THEN TECLA
   IF P5=0 THEN TECLA
   PAUSE 100
RETURN
LED1:      ;SUBRUTINA DE ENCENDIDO DE LED
   HIGH LED
   PAUSE 120
   LOW LED
   pause 120
   HIGH LED
   PAUSE 120
   LOW LED
RETURN
LCD:      ;SUBRUTINA DE BORRADO DE PANTALLA
   LCDOUT $FE,1 ;limpiar pantalla
   LCDOUT,$FE,$80
   LCDOUT "      MINIROBOT"
   LCDOUT,$FE,$C0
```

```
   LCDOUT "      ========="
   LCDOUT,$FE,$95
   LCDOUT "CONTROLADO POR RF"
RETURN
```

- Realizar la programación para el RX.

PROGRAMACIÒN DEL PIC
RX

```
'*****************************************************************
'*  Name    : MINIROBOT_RX
'*  Author  : EDISON JAVIER TERAN GUALOTO
'*  Notice  : Copyright (c) 2012 EDISON TERAN
'*          : All Rights Reserved
'*  Date    : 23/02/2012
'*  Version : 1.0
'*  Notes   :
'*          :
'*****************************************************************
;libreria para radio frecuencia, incluye los modos de comunicaciòn
include "modedefs.bas"
;ASIGNAMOS NOMBRES A LOS PUERTOS
RECEPTOR  VAR PORTB.0
IN1       VAR PORTB.1
IN2       VAR PORTB.2
IN3       VAR PORTB.3
IN4       VAR PORTB.4
ENA       VAR PORTB.5
ENB       VAR PORTB.6
LED       VAR PORTB.7
;VARIABLE PARA ALMACENAR DATOS RECIBIDOS
DATOS     VAR BYTE
;SUBRUTINA INICIO LEE DATOS RECIBIDOS
INICIO:
  SERIN receptor,N2400,DATOS
  IF DATOS="W" THEN ADELANTE
  IF DATOS="X" THEN ATRAS
  IF DATOS="Y" THEN DERECHA
  IF DATOS="Z" THEN IZQUIERDA
  IF DATOS="S" THEN PARAR
GOTO INICIO
;SI SE RECIBIO LA LETRA W
ADELANTE:
   HIGH ENA
   HIGH ENB
   HIGH IN1
   LOW  IN2
   HIGH IN3
   LOW  IN4
GOTO INICIO
;SI SE RECIBIO LA LETRA X
ATRAS:
   HIGH ENA
```

```
    HIGH ENB
    HIGH IN2
    LOW  IN1
    HIGH IN4
    LOW  IN3
GOTO INICIO
;SI SE RECIBIO LA LETRA Y
DERECHA:
    LOW  ENA
    HIGH ENB
    LOW  IN2
    LOW  IN1
    low  IN4
    high IN3
GOTO INICIO
;SI SE RECIBIO LA LETRA Z
IZQUIERDA:
    HIGH ENA
    LOW  ENB
    low  IN2
    high IN1
    LOW  IN4
    LOW  IN3
GOTO INICIO
;SI SE RECIBIO LA LETRA S
PARAR:
    LOW ENA
    LOW  ENB
    LOW IN2
    LOW  IN1
    LOW  IN4
    LOW  IN3
GOTO INICIO
```

- Comprobar el funcionamiento del Entrenador Lógico y del Minirobot.

El Entrenador Lógico establece una comunicación óptima.

6. DIAGRAMAS Y FIGURAS.

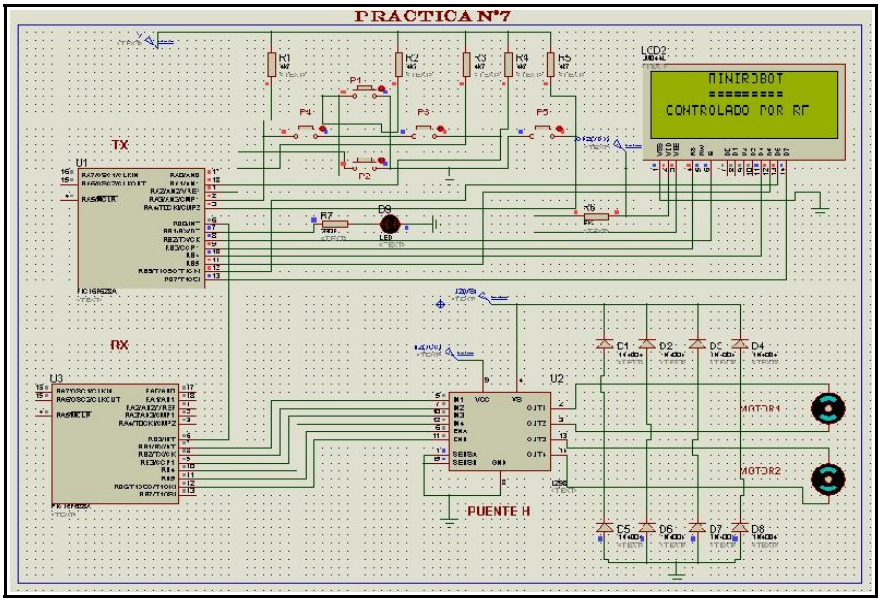


Figura 6.1: Práctica N°7,TX y RX.

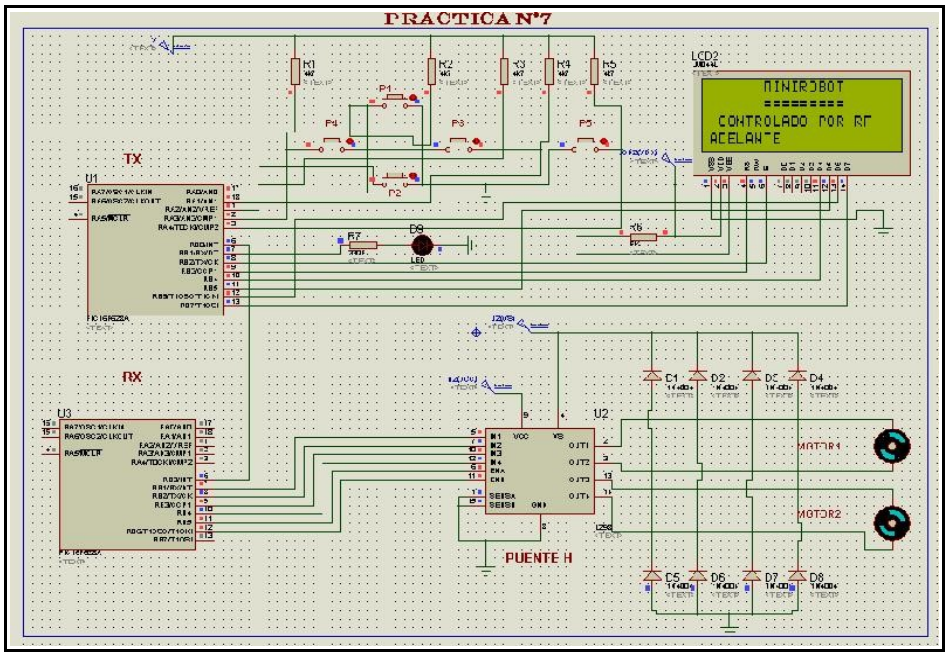


Figura 6.2: Práctica N°7, adelante.

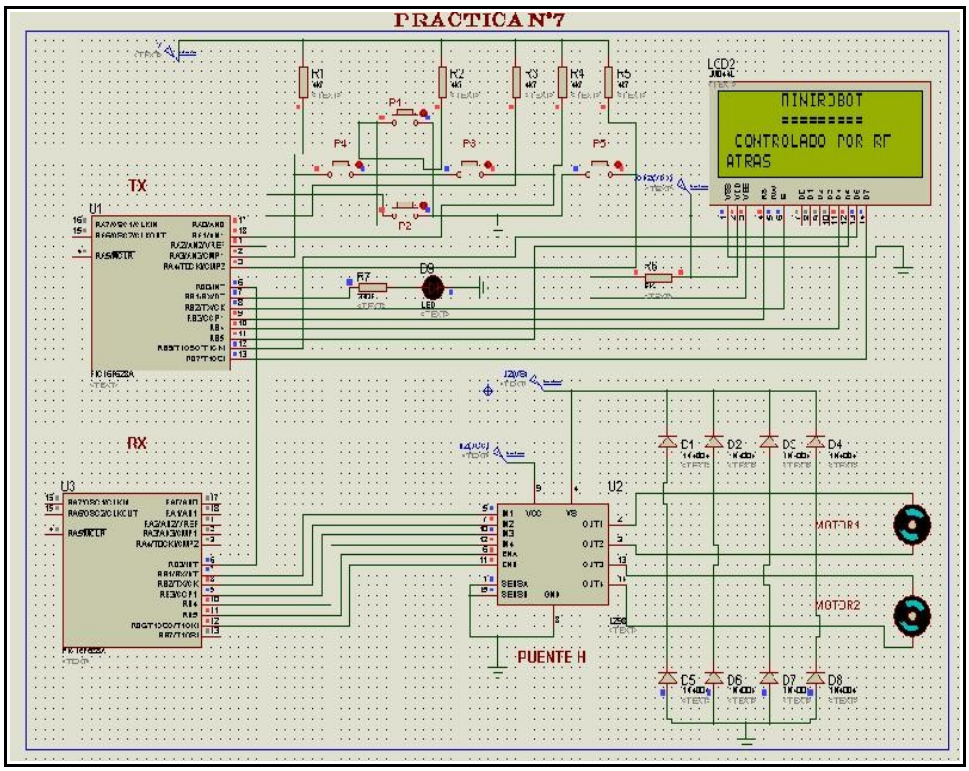


Figura 6.3: Práctica N°7, atrás.

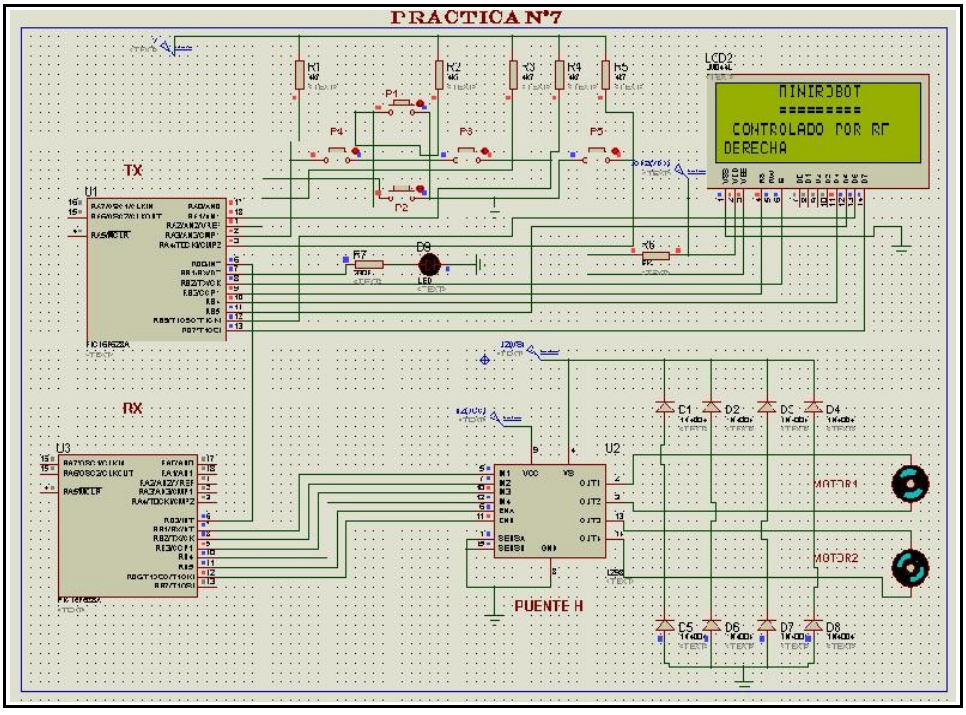


Figura 6.4: Práctica N°7, derecha.

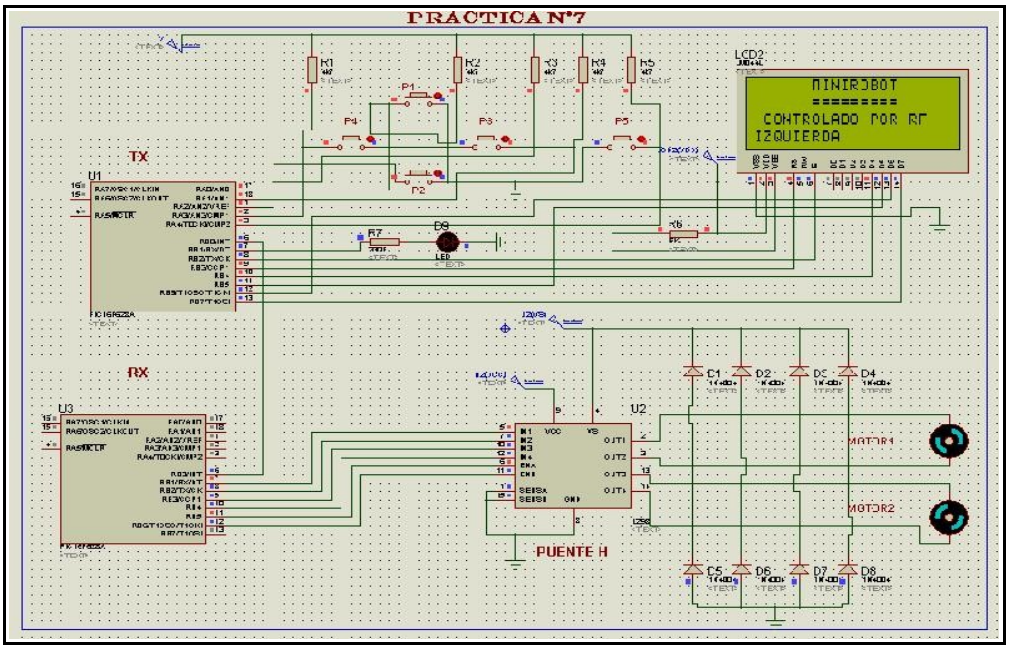


Figura 6.5: Práctica N°7, izquierda.

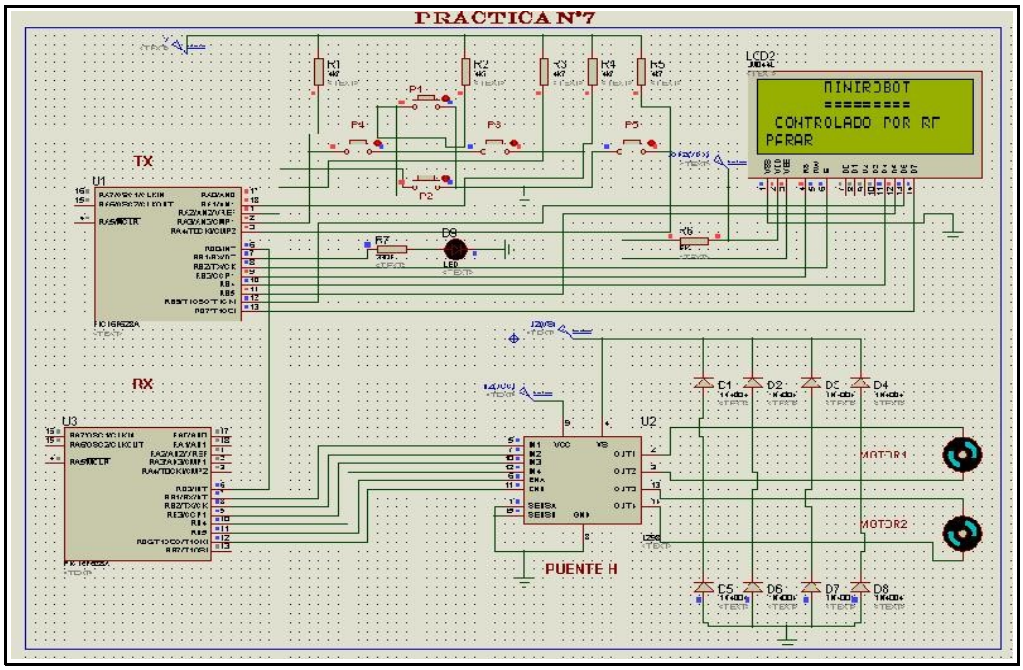


Figura 6.6: Práctica N°7, parar.

7. TABULACIÓN Y RESULTADOS.

ARCHIVO	ESTADO
ADELANTE	CORECTO
ATRAS	CORECTO
DERECHA	CORECTO
IZQUIERDA	CORECTO
PARAR	CORECTO

Tabla 7.1: Funcionamiento del archivo ejecutable.

8. CONCLUSIONES.

- Se implementó la comunicación en base a módulos de RF de 434 Mhz.
- El funcionamiento es en base a una simulación de la comunicación serial que se la realiza en el MCS.
- El Entrenador Lógico provee de los recursos tecnológicos necesarios para el desarrollo de la comunicación inalámbrica.

9. BIBLIOGRAFÍA.

- http://es.wikipedia.org/wiki/Puente_H_%28electr%C3%B3nica

10. ANEXOS.

PRÁCTICA N° 8
MINI ROBOT CONTROLADO POR BT A TRAVÉS DE J2ME

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Minirobot controlado por BT a través de J2ME.

2. OBJETIVOS

2.1. OBETIVO GENERAL

Diseñar e implementar un mini robot controlado por bluetooth a través de J2ME.

2.2. OBJETIVOS ESPECÍFICOS

- Desarrollar una interfaz gráfica, como aplicación java, para un teléfono celular capaz de controlar el mini robot.
- Comprobar el funcionamiento de la comunicación inalámbrica entre el teléfono celular y el Minirobot.

3. MARCO TEÓRICO

BLUETOOTH

La tecnología Bluetooth es una especificación abierta para la comunicación inalámbrica (WIRELESS) de datos y voz. Está basada en un enlace de radio de bajo costo y corto alcance, implementado en un circuito integrado de 9 x 9 mm, proporcionando conexiones instantáneas (ad hoc) para entornos de comunicaciones tanto móviles como estáticos. En definitiva, Bluetooth pretende ser una especificación global para la conectividad inalámbrica.

El principal objetivo de esta tecnología, es la posibilidad de reemplazar los muchos cables propietarios que conectan unos dispositivos con otros por medio de un enlace

radio universal de corto alcance. Por ejemplo, la tecnología de radio Bluetooth implementada en el teléfono celular y en el ordenador portátil reemplazaría el molesto cable utilizado hoy en día para conectar ambos aparatos. Las impresoras, las agendas electrónicas, los PDA, los faxes, los teclados, los joysticks y prácticamente cualquier otro dispositivo digital son susceptibles de formar parte de un sistema Bluetooth.

Pero más allá de reemplazar, los incómodos cables, la tecnología Bluetooth ofrece un puente a las redes de datos existentes, una interfaz con el exterior y un mecanismo para formar en el momento, pequeños grupos de dispositivos conectados entre sí de forma privada fuera de cualquier estructura fija de red.

Integrado en un pequeño transmisor de radiofrecuencia que permite conectar entre sí todo tipo de dispositivos electrónicos (teléfonos, ordenadores, impresoras, faxes, etc) situados dentro de un radio limitado de 10 metros (ampliable a 100, aunque con mayor distorsión) sin necesidad de utilizar cables.

El transmisor está integrado en un pequeño microchip de 9x9 milímetros y opera en una frecuencia de banda global (2,4 GHz, utilizada en muchos países para usos médicos y científicos) que asegura la compatibilidad universal. Los dispositivos que incorporan Bluetooth se reconocen y se hablan de la misma forma que lo hace un ordenador con su impresora. El canal permanece abierto y no requiere la intervención directa y constante del usuario cada vez que se quiere enviar algo.

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

Las diferentes partes del sistema Bluetooth son:

- Una unidad de radio
- Una unidad de control del enlace
- Gestión del enlace
- Funciones software

4. LISTADO DE MATERIALES Y EQUIPOS.

- Entrenador Lógico.
- Minirobot.
- Teléfono celular.
- Cables de par trenzado.

5. PROCEDIMIENTO.

- Establecer los movimientos del Minirobot.

  El Minirobot esta diseñado para realizar los siguientes movimientos:

  - Adelante.
  - Atrás.
  - Derecha.
  - Izquierda.
  - Parar.

- Realizar la programación

PROGRAMACIÒN TELÈFONO CELULAR
MIDLET CONTROLADOR_BT

```
import javax.bluetooth.ServiceRecord;
import javax.microedition.midlet.*;

/**
 * @author JAVTER
 */
public class CONTROLADORBT extends MIDlet implements Runnable {

    public void startApp() {
        new Portada(this);
        new Thread(this).start();
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void exitMidlet() {
        destroyApp(true);
        notifyDestroyed();
    }

    public void run() {
        try {
            Thread.sleep(3000);
        } catch (InterruptedException ex) {
        }
           GUIListaOpciones bluetooth = new GUIListaOpciones(this);
        bluetooth.makeConnections("btspp://00195DEE405B:1;authenticate=false;encrypt=false;mast
er=false", ServiceRecord.NOAUTHENTICATE_NOENCRYPT);

    }
}
```

CLASE PORTADA

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.MIDlet;
public class Portada {
    private Display display;
    private MyCanvas canvas = new MyCanvas();
    private MIDlet midlet;
    public Portada(MIDlet midletIn) {
        midlet =  midletIn;
        display = Display.getDisplay(midlet);
        display.setCurrent(canvas);
    }
    public Display getDisplay() {
        return display;
    }
}
class MyCanvas extends Canvas {
    private Image image = null;
```

```
    public MyCanvas() {
        try {
            image = Image.createImage("/logox1.PNG");
        } catch (Exception error) {
            Alert alert = new Alert("Failure", "Can't open image file.", null, null);
            alert.setTimeout(Alert.FOREVER);
        }
    }
    protected void paint(Graphics graphics) {
        graphics.setColor(255,255,255);//RGB
        graphics.fillRect (0, 0, getWidth(), getHeight());
        if (image != null) {
            graphics.drawImage(image, getWidth() / 2, getHeight() / 2, Graphics.HCENTER |
Graphics.VCENTER);
        }
        setFullScreenMode(true);
    }
}
```

CLASE GUILISTA OPCIONES

```
import javax.microedition.lcdui.*;
import java.util.Vector;
import javax.bluetooth.LocalDevice;
public class GUIListaOpciones implements ConexionHandlerListener, CommandListener {
    private Alert alert;
    private Display display;
    private CONTROLADORBT mIDlet;
    private Command salir;
    private List listaOpciones;
    private Command atraz;
    private final Vector handlers;
    private volatile int numReceivedMessages = 0;
    private volatile int numSentMessages = 0;
    String receive = "";
    private int maxConnections;
    private int sendMessageId = 0;
    private String sendcommand;
    private String tipo="";
    public GUIListaOpciones(CONTROLADORBT mIDletIn) {
        init();
        mIDlet = mIDletIn;
        display = Display.getDisplay(mIDlet);
        display.setCurrent(listaOpciones);
        handlers = new Vector();
       String value =
            LocalDevice.getProperty(
                        "bluetooth.connected.devices.max");
        try
        {
            maxConnections = Integer.parseInt(value);
        }
        catch (NumberFormatException e)
        {
            maxConnections = 0;
        }
    }
```

```
private void init() {
    salir = new Command("Salir", Command.EXIT, 2);
    atraz = new Command("Atrás", Command.BACK, 1);
    listaOpciones = new List("MINIROBOT BT:", List.IMPLICIT);
    listaOpciones.append("ADELANTE", null);
    listaOpciones.append("ATRAS", null);
    listaOpciones.append("DERECHA", null);
    listaOpciones.append("IZQUIERDA", null);
    listaOpciones.append("PARAR", null);
    listaOpciones.addCommand(salir);
    listaOpciones.setCommandListener(this);
}
public void presentarAlerta(String resultado) {
    if (resultado.startsWith("Error:")) {
        alert = new Alert("Alerta", resultado, null, AlertType.WARNING);
    } else if (resultado.startsWith("Ok:")) {
        alert = new Alert("Alerta", resultado, null, AlertType.INFO);
    }
    alert.setTimeout(Alert.FOREVER);
    alert.addCommand(new Command("Salir", Command.EXIT, 2));
    display.setCurrent(alert, listaOpciones);
}
public void commandAction(Command c, Displayable d) {
    if (c == listaOpciones.SELECT_COMMAND) {
        String label = listaOpciones.getString(listaOpciones.getSelectedIndex());
        //System.out.println(label);
        if (label.equals("ADELANTE")) {
            tipo="1";
            listaOpciones.deleteAll();
            listaOpciones.append("ON", null);
            listaOpciones.append("OFF", null);
            listaOpciones.addCommand(atraz);
        } else if (label.equals("ATRAS")) {
            tipo="2";
            listaOpciones.deleteAll();
            listaOpciones.append("ON", null);
            listaOpciones.append("OFF", null);
            listaOpciones.addCommand(atraz);
        } else if (label.equals("DERECHA")) {
            tipo="3";
            listaOpciones.deleteAll();
            listaOpciones.append("ON", null);
            listaOpciones.append("OFF", null);
            listaOpciones.addCommand(atraz);
        } else if (label.equals("IZQUIERDA")) {
            tipo="4";
            listaOpciones.deleteAll();
            listaOpciones.append("ON", null);
            listaOpciones.append("OFF", null);
            listaOpciones.addCommand(atraz);
        } else if (label.equals("PARAR")) {
            tipo="5";
            listaOpciones.deleteAll();
            listaOpciones.append("ON", null);
            listaOpciones.append("OFF", null);
```

```
                listaOpciones.addCommand(atraz);
            } else if (label.equals("ON")) {
                send("si"+tipo);
            } else if (label.equals("OFF")) {
                send("no"+tipo);
            }
        } else if (c == atraz) {
            listaOpciones.deleteAll();
            listaOpciones.removeCommand(atraz);
            listaOpciones.append("ADELANTE", null);
            listaOpciones.append("ATRAS", null);
            listaOpciones.append("DERECHA", null);
            listaOpciones.append("IZQUIERDA", null);
            listaOpciones.append("PARAR", null);
            listaOpciones.addCommand(salir);
            listaOpciones.setCommandListener(this);
        } else if (c == salir) {
            ((CONTROLADORBT) mIDlet).exitMidlet();
        }
    }
    public void handleOpen(ConexionHandler handler) {
        handlers.addElement(handler);
        // for the first open connection
        if (handlers.size() == 1) {
//            removeCommand(searchCommand);
//            removeCommand(sendCommand);
//            addCommand(sendCommand);
        }
        // Remove the 'Add connection' command
        // when the device already has open the
        // maximum number of connections it can
        // support.
        if (handlers.size() >= maxConnections) {
//            removeCommand(addConnectionCommand);
        }
        //texto2="Connection opened";
        String str = Integer.toString(handlers.size());
        //texto1=str;
        //fondoambiente=1;
        listaOpciones.append("Conectado", null);
    }
    public void handleOpenError(
            ConexionHandler handler,
            String errorMessage) {
        //midlet.noencontrado("El dispositivo no esta dentro del area de servicio");
        listaOpciones.append("Error de Conexion", null);
    }
    public void handleReceivedMessage(
            ConexionHandler handler,
            byte[] messageBytes) {
        numReceivedMessages++;
        String message = new String(messageBytes);
        receive = message;
        // texto2="# messages read: " + numReceivedMessages + " " +
        //     "sent: " + numSentMessages;
```

```
        if (message.substring(4, 5).equals("0")) {
            //   receive="off";
            //    foco.selFondo(0);
        }
        // Broadcast message to all clients
//        for (int i=0; i < handlers.size(); i++)
//        {
//            ConexionHandler h =
//                (ConexionHandler) handlers.elementAt(i);
//            Integer id = new Integer(sendMessageId++);
//            try
//            {
//                h.queueMessageForSending(id, messageBytes);
//            }
//            catch (IllegalArgumentException e)
//            {
//                String errorMessage =
//                    "IllegalArgumentException while trying to " +
//                    "send message: " + e.getMessage();
//                handleError(handler, errorMessage);
//            }
//        }
        listaOpciones.append("recibiendo datos", null);
    }
    public void handleQueuedMessageWasSent(
            ConexionHandler handler,
            Integer id) {
        numSentMessages++;
        //texto2="# messages read: " +numReceivedMessages + " " + "sent: " + numSentMessages;
    }
    public void handleClose(ConexionHandler handler) {
        removeHandler(handler);
        if (handlers.size() == 0) {
//            removeCommand(sendCommand);
//            addCommand(searchCommand);
        }
        // If the number of currently open connections
        // drops below the maximum number that this
        // device could have open, restore
        // 'Add connection' to the screen commands.
        if (handlers.size() < maxConnections) {
//            removeCommand(addConnectionCommand);
//            addCommand(addConnectionCommand);
        }
        //texto2="Connection closed";
        closeAll();
    }
    public void handleErrorClose(ConexionHandler handler,
            String errorMessage) {
        removeHandler(handler);
        if (handlers.size() == 0) {
//            removeCommand(sendCommand);
//            addCommand(searchCommand);
        }
     //   texto2 = "Error (close): '" + errorMessage + "'";
```

```
    }
    public void handleError(ConexionHandler handler,
            String errorMessage) {
        //texto2 = "Error: '" + errorMessage + "'";
    }
    public void makeConnections(String URL, int security) {
        ConexionHandler newHandler =
                new ConexionHandler(
                this,
                URL,
                security);
        newHandler.start(); // start reader & writer
    }
    private void removeHandler(ConexionHandler handler) {
        if (handlers.contains(handler)) {
            handlers.removeElement(handler);
            String str = Integer.toString(handlers.size());
          //  texto1 = str;
            if (handlers.size() == 0) {
            }
        }
    }
    private void send(String sendData) {
        Integer id = new Integer(sendMessageId++);
        for (int i = 0; i < handlers.size(); i++) {
            ConexionHandler handler =
                    (ConexionHandler) handlers.elementAt(i);
            try {
                handler.queueMessageForSending(
                        id,
                        sendData.getBytes());
            //    texto1 = "Queued a send message request";
            } catch (IllegalArgumentException e) {
                // Message length longer than
                // ServerConnectionHandler.MAX_MESSAGE_LENGTH
                String errorMessage =
                        "IllegalArgumentException while trying "
                        + "to send a message: " + e.getMessage();
                //handleError(handler, errorMessage);
            }
        }
    }
    void closeAll() {
        for (int i = 0; i < handlers.size(); i++) {
            ConexionHandler handler =
                    (ConexionHandler) handlers.elementAt(i);
            handler.close();
            removeHandler(handler);
        }
    }
}
```

CLASE CONEXIÓN HANDLER

```
import java.io.InputStream;
import java.io.IOException;
import java.io.OutputStream;
```

```
import java.util.Hashtable;
import java.util.Enumeration;
import javax.bluetooth.ServiceRecord;
import javax.microedition.io.Connector;
import javax.microedition.io.StreamConnection;
import javax.microedition.io.StreamConnectionNotifier;
public class ConexionHandler
    implements Runnable
{
    private final static byte ZERO = (byte) '0';
    private final static int LENGTH_MAX_DIGITS = 5;
    // this is an arbitrarily chosen value:
    private final static int MAX_MESSAGE_LENGTH =
                          65536 - LENGTH_MAX_DIGITS;
    //private final ServiceRecord serviceRecord;
    private final int requiredSecurity;
    private final ConexionHandlerListener listener;
    private final Hashtable sendMessages = new Hashtable();
    private StreamConnection connection;
    private OutputStream out;
    private InputStream in;
    private volatile boolean aborting;
    private Writer writer;
    private String url;
    public ConexionHandler(
            ConexionHandlerListener listener,
            String URL,
            int requiredSecurity)
    {
        this.listener = listener;
        this.url=URL;
        //this.serviceRecord = serviceRecord;
        this.requiredSecurity = requiredSecurity;
        aborting = false;
        connection = null;
        out = null;
        in = null;
        listener = null;
        // the caller must call method 'start'
        // to start the reader and writer
    }
//    public ServiceRecord getServiceRecord()
//    {
//        return serviceRecord;
//    }
    public synchronized void start()
    {
        Thread thread = new Thread(this);
        thread.start();
    }
    public void close()
    {
        if (!aborting)
        {
            synchronized(this)
```

```
        {
            aborting = true;
        }
        synchronized(sendMessages)
        {
            sendMessages.notify();
        }
        if (out != null)
        {
            try
            {
                out.close();
                synchronized (this)
                {
                    out = null;
                }
            }
            catch(IOException e)
            {
                // there is nothing we can do: ignore it
            }
        }
        if (in != null)
        {
            try
            {
                in.close();
                synchronized (this)
                {
                    in = null;
                }
            }
            catch(IOException e)
            {
                // there is nothing we can do: ignore it
            }
        }
        if (connection != null)
        {
            try
            {
                connection.close();
                synchronized (this)
                {
                    connection = null;
                }
            }
            catch (IOException e)
            {
                // there is nothing we can do: ignore it
            }
        }
    }
}
public void queueMessageForSending(Integer id, byte[] data)
```

```
    {
        if (data.length > MAX_MESSAGE_LENGTH)
        {
            throw new IllegalArgumentException(
                        "Message too long: limit is " +
                        MAX_MESSAGE_LENGTH + " bytes");
        }
        synchronized(sendMessages)
        {
            sendMessages.put(id, data);
            sendMessages.notify();
        }
    }
    public void run()
    {
        // the reader
        // 1. open the connection and streams, start the writer
        //String url = null;
        try
        {
            // 'must be master': false
            //url = serviceRecord.getConnectionURL(
            //                    requiredSecurity,
            //                    false);
            connection = (StreamConnection) Connector.open(url);
            in = connection.openInputStream();
            out = connection.openOutputStream();
//            LogScreen.log("Opened connection & streams to: '" +
//                    url + "'\n");
            // start the writer
            Writer writer = new Writer(this);
            Thread writeThread = new Thread(writer);
            writeThread.start();
//            LogScreen.log("Started a reader & writer for: '" +
//                    url + "'\n");
            // open succeeded, inform listener
            listener.handleOpen(this);
        }
        catch(IOException e)
        {
            // open failed, close any connections/streams, and
            // inform listener that the open failed
 //            LogScreen.log("Failed to open " +
 //                        "connection or streams for '" +
 //                         url + "' , Error: " +
 //                         e.getMessage());
            close();
            listener.handleOpenError(
                        this,
                        "IOException: '" + e.getMessage() + "'");
            return;
        }
        catch (SecurityException e)
        {
            // open failed, close any connections/streams, and
```

```
            // inform listener that the open failed
//            LogScreen.log("Failed to open " +
//                          "connection or streams for '" +
//                           url + "' , Error: " +
//                           e.getMessage());
            close();
            listener.handleOpenError(
                        this,
                        "SecurityException: '" + e.getMessage() + "'");
            return;
        }
        // 2. wait to receive and read messages
        while (!aborting)
        {
            int length = 0;
            try
            {
                byte[] lengthBuf = new byte[LENGTH_MAX_DIGITS];
                readFully(in, lengthBuf);
                length = readLength(lengthBuf);
                byte[] temp = new byte[length];
                readFully(in, temp);
                listener.handleReceivedMessage(this, temp);
            }
            catch (IOException e)
            {
                close();
                if (length == 0)
                {
                  listener.handleClose(this);
                }
                else
                {
                  // we were in the middle of reading...
                  listener.handleErrorClose(this, e.getMessage());
                }
            }
        }
    }
    private static void readFully(InputStream in, byte[] buffer)
        throws IOException
    {
        int bytesRead = 0;
        while (bytesRead < buffer.length)
        {
            int count = in.read(buffer,
                                bytesRead,
                                buffer.length - bytesRead);
            if (count == -1)
            {
                throw new IOException("Input stream closed");
            }
            bytesRead += count;
        }
    }
```

```
private static int readLength(byte[] buffer)
{
   int value = 0;
   for (int i = 0; i < LENGTH_MAX_DIGITS; ++i)
   {
      value *= 10;
      value += buffer[i] - ZERO;
   }
   return value;
}
private void sendMessage(OutputStream out, byte[] data)
   throws IOException
{
   int i;
   byte[] buf=new byte[2+data.length];
   for (i=0;i<data.length;i++){
      buf[i]=data[i];
   }
   buf[i]=13;
   buf[i+1]=10;
   out.write(buf);
   out.flush();
}
private static void writeLength(int value, byte[] buffer)
{
   for (int i = LENGTH_MAX_DIGITS -1; i >= 0; --i)
   {
      buffer[i] = (byte) (ZERO + value % 10);
      value = value / 10;
   }
}
private class Writer
   implements Runnable
{
   private final ConexionHandler handler;
   Writer(ConexionHandler handler)
   {
      this.handler = handler;
   }
   public void run()
   {
      while (!aborting)
      {
         synchronized(sendMessages)
         {
            Enumeration e = sendMessages.keys();
            if (e.hasMoreElements())
            {
               // send any pending messages
               Integer id = (Integer) e.nextElement();
               byte[] sendData =
                     (byte[]) sendMessages.get(id);
               try
               {
                  sendMessage(out, sendData);
```

```
                    // remove sent message from queue
                    sendMessages.remove(id);

                    // inform listener that it was sent
                    listener.handleQueuedMessageWasSent(
                            handler,
                            id);
                }
                catch (IOException ex)
                {
                    close(); // stop the networking thread

                    // inform that we got an error close
                    listener.handleErrorClose(handler,
                                    ex.getMessage());
                }
            }

            if (sendMessages.isEmpty())
            {
                try
                {
                    sendMessages.wait();
                }
                catch (InterruptedException ex)
                {
                    // this can't happen in MIDP: ignore it
                }
            }
        }
    }
  }
}
```

CLASE CONEXIÓN HANDLER LISTENER

```
public interface ConexionHandlerListener
{
   // The handler's open succeeded. It can now be used for sending
   // and receiving messages.
   public void handleOpen(ConexionHandler handler);
   // The hadler's open failed. It has closed any connections or
   // streams that were open. The handler should not be used anymore,
   // and should be discarded.
   public void handleOpenError(ConexionHandler handler,
                       String errorMessage);
   public void handleQueuedMessageWasSent(ConexionHandler handler, Integer id);
   // The handler got an inbound message.
   public void handleReceivedMessage(
               ConexionHandler handler,
               byte[] messageBytes);
   // A message that had been previously queued for sending
   // (identified by id) by the handler, has been sent successfully.
   // The handler has closed its connections and streams.
   // The handler should not be used anymore, and should be discarded.
```

```
    // Only handlers which have previously called 'handleOpen' may
    // call 'handleClose', and only just once.
    public void handleClose(ConexionHandler handler);
    // An error related to the handler occurred. The handler
    // has closed the connection, and the handler should no
    // longer be used.
    public void handleErrorClose(ConexionHandler handler,
                                 String errorMessage);

}
```

- Comprobar el funcionamiento entre el teléfono celular y el Minirobot.

Se establece una comunicación óptima.

6. DIAGRAMAS Y FIGURAS.

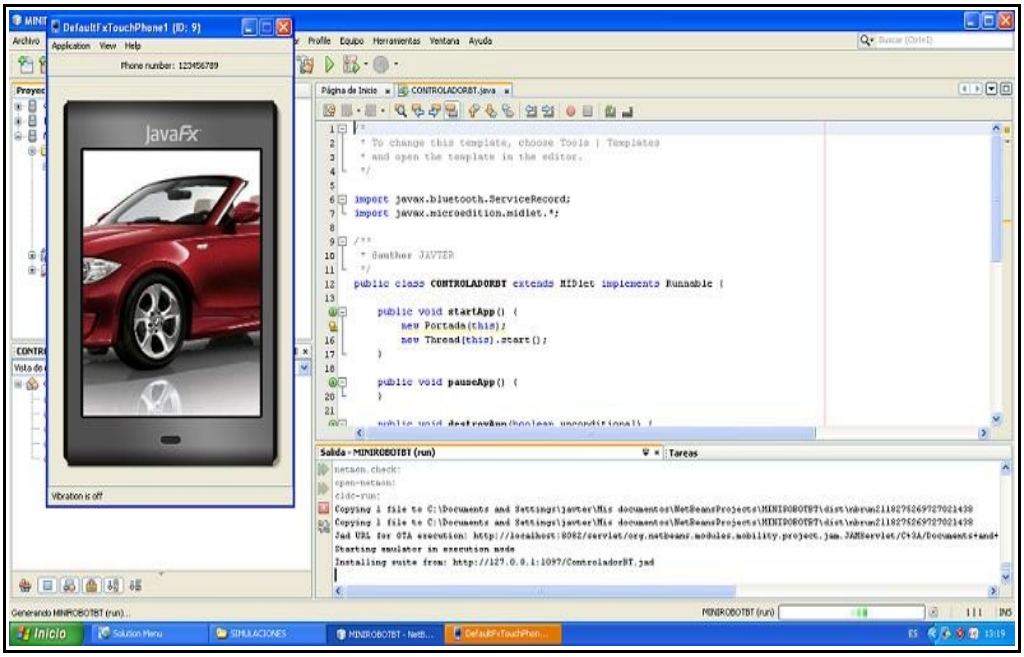


Figura 6.1: Práctica N°8, pantalla de inicio.

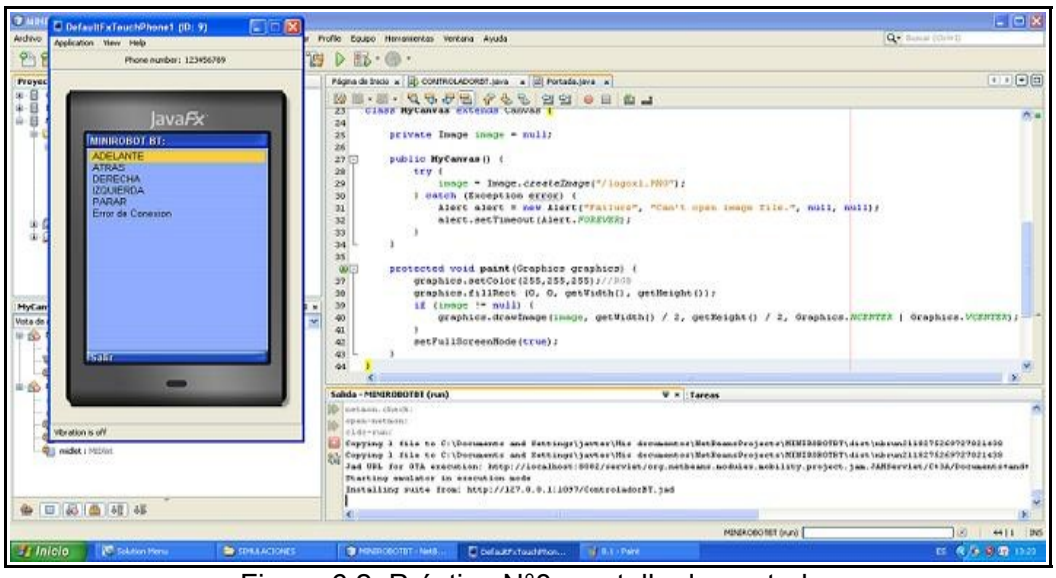


Figura 6.2: Práctica N°8, pantalla de control.

7. TABULACIÓN Y RESULTADOS.

ARCHIVO	ESTADO
ADELANTE	CORECTO
ATRAS	CORECTO
DERECHA	CORECTO
IZQUIERDA	CORECTO
PARAR	CORECTO

Tabla 7.1: Funcionamiento del archivo ejecutable .JAR.

8. CONCLUSIONES.

- Para la implementación se utilizó un módulo BT de clase 2 y su alcance máximo es de 10 metros en linea de vista.
- La programación en Netbeans se la realiza en base clases y solo se crea una aplicación ejecutable.
- La comunicación se realizó con éxito, entre el módulo y el teléfono celular.

9. BIBLIOGRAFÍA.

- http://sourceforge.net/projects/bluetoothled/files/
- http://www.google.com/search?client=ubuntu&channel=fs&q=comunicacion+bt+j2me&ie=utf-8&oe=utf-8#hl=es&client=ubuntu&hs=lcr&channel=fs&sa=X&ei=RQRAT_uxLMavg-

wes2YSbCA&sqi=2&ved=0CBgQBSgA&q=comandos+de+comunicaci
%C3%B3n+en+netbeans&spell=1&bav=on.2,or.r_gc.r_pw.,cf.osb&fp=141a04
20e47bc3e0&biw=1368&bih=606

10. ANEXOS.

PRÁCTICA N° 9
MINI ROBOT CONTROLADO POR BT A TRAVÉS DE ANDROID

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO: Minirobot controlado por BT a través de ANDROID.

2. OBJETIVOS

2.1. OBETIVO GENERAL

Diseñar e implementar un mini robot controlado por bluetooth a través de ANDROID.

2.2. OBJETIVOS ESPECÍFICOS

- Desarrollar una interfaz gráfica, como aplicación Android, para un teléfono celular capaz de controlar el mini robot.
- Comprobar el funcionamiento de la comunicación inalámbrica entre el teléfono celular y el Minirobot.

3. MARCO TEÓRICO

BLUETOOTH

La tecnología Bluetooth es una especificación abierta para la comunicación inalámbrica (WIRELESS) de datos y voz. Está basada en un enlace de radio de bajo costo y corto alcance, implementado en un circuito integrado de 9 x 9 mm, proporcionando conexiones instantáneas (ad hoc) para entornos de comunicaciones tanto móviles como estáticos. En definitiva, Bluetooth pretende ser una especificación global para la conectividad inalámbrica.

El principal objetivo de esta tecnología, es la posibilidad de reemplazar los muchos cables propietarios que conectan unos dispositivos con otros por medio de un enlace

radio universal de corto alcance. Por ejemplo, la tecnología de radio Bluetooth implementada en el teléfono celular y en el ordenador portátil reemplazaría el molesto cable utilizado hoy en día para conectar ambos aparatos. Las impresoras, las agendas electrónicas, los PDA, los faxes, los teclados, los joysticks y prácticamente cualquier otro dispositivo digital son susceptibles de formar parte de un sistema Bluetooth.

Pero más allá de reemplazar, los incómodos cables, la tecnología Bluetooth ofrece un puente a las redes de datos existentes, una interfaz con el exterior y un mecanismo para formar en el momento, pequeños grupos de dispositivos conectados entre sí de forma privada fuera de cualquier estructura fija de red.

Integrado en un pequeño transmisor de radiofrecuencia que permite conectar entre sí todo tipo de dispositivos electrónicos (teléfonos, ordenadores, impresoras, faxes, etc) situados dentro de un radio limitado de 10 metros (ampliable a 100, aunque con mayor distorsión) sin necesidad de utilizar cables.

El transmisor está integrado en un pequeño microchip de 9x9 milímetros y opera en una frecuencia de banda global (2,4 GHz, utilizada en muchos países para usos médicos y científicos) que asegura la compatibilidad universal. Los dispositivos que incorporan Bluetooth se reconocen y se hablan de la misma forma que lo hace un ordenador con su impresora. El canal permanece abierto y no requiere la intervención directa y constante del usuario cada vez que se quiere enviar algo.

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

Las diferentes partes del sistema Bluetooth son:

- Una unidad de radio
- Una unidad de control del enlace
- Gestión del enlace
- Funciones software

Android es un sistema operativo móvil basado en Linux, que junto con aplicaciones middleware, está enfocado para ser utilizado en dispositivos móviles como teléfonos

inteligentes, tablets, Google TV y otros dispositivos. Es desarrollado por la Open Handset Alliance, la cual es liderada por Google.

Fue desarrollado inicialmente por Android Inc., una firma comprada por Google en 2005. Es el principal producto de la Open Handset Alliance, un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio. Las unidades vendidas de teléfonos inteligentes con Android se ubican en el primer puesto en los Estados Unidos, en el segundo y tercer trimestres de 2010, con una cuota de mercado de 43,6% en el tercer trimestre.

Tiene una gran comunidad de desarrolladores escribiendo aplicaciones para extender la funcionalidad de los dispositivos. A la fecha, se han sobrepasado las 400.000 aplicaciones (de las cuales, dos tercios son gratuitas) disponibles para la tienda de aplicaciones oficial de Android: Android Market, sin tener en cuenta aplicaciones de otras tiendas no oficiales para Android, como la App Store de Amazon o la tienda de aplicaciones Samsung Apps de Samsung. Android Market es la tienda de aplicaciones en línea administrada por Google, aunque existe la posibilidad de obtener software externamente. Los programas están escritos en el lenguaje de programación Java. No obstante, no es un sistema operativo libre de malware, aunque la mayoría de ello es descargado de sitios de terceros.

Características de Android

Android se podría decir que es hijo de Linux, que no es otra cosa que otro sistema operativo diseñado para ordenadores. La principal característica de Linux, heredada por Android, radica en el sistema abierto de programación, es decir, contiene un código libre, que permite a quien desee transformarlo, adaptarlo o modificarlo, dependiendo de sus necesidades.

Principales ventajas de Android.

Al ser un código libre o abierto, la modificación, mejora y solución de problemas es más rápida y eficaz, ya que todo el mundo puede acceder al mismo, al contrario que sucede con otros sistemas operativos. Este último caso exige estar pendiente que la compañía o empresa responsable del sistema operativo del celular en cuestión, realice las mejoras o reparaciones necesarias, que en algunos casos pueden demorarse incluso algunos meses.

Móviles mas económicos.

Las especificaciones concretas de Android, revierten en una serie de beneficios al comprar un celular con este sistema operativo.

Android, al ser un sistema basado en Linux, abierto y libre, no conlleva pagos de licencias, y por consiguiente, abarata los costos, lo que finalmente se traduce en unos móviles con un precio de venta inferior.

- Ha sido desarrollado principalmente para *smartphones* de gama media-alta y con pantalla táctil, y ofrece rapidez de reacción y una gran exactitud en el manejo del teclado virtual.

Android es un sistema operativo en continua evolución y mejora, lo que significa que muchas de las ventajas que aquí se exponen irán rápidamente en aumento:

- Resuelve cualquier web en flash, lo que significa que es posible ver vídeos y acceder a juegos normalmente.
- Aplicaciones en Android Market: contiene miles de aplicaciones y no deja de crecer con nuevos contenidos para el celular, y no tiene restricciones.
- Android se instala en la mayoría de marcas y operadoras, al contrario que otros sistemas operativos, y ofrece al usuario la posibilidad de elegir el móvil que más le guste.

4. LISTADO DE MATERIALES Y EQUIPOS.

- Entrenador Lógico.
- Minirobot.
- Teléfono celular.
- Cables de par trenzado.

5. PROCEDIMIENTO.

- Establecer los movimientos del Minirobot.

El Minirobot esta diseñado para realizar los siguientes movimientos:

- Adelante.
- Atrás.
- Derecha.
- Izquierda.
- Parar.

- Realizar la programación

BLUETOOTH CHAT

```
package com.example.android.BluetoothChat;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.Window;
import android.view.View.OnClickListener;
import android.view.inputmethod.EditorInfo;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;
/**
 * This is the main Activity that displays the current chat session.
 */
public class BluetoothChat extends Activity {
    // Debugging
    private static final String TAG = "BluetoothChat";
    private static final boolean D = true;
    // Message types sent from the BluetoothChatService Handler
    public static final int MESSAGE_STATE_CHANGE = 1;
```

```
public static final int MESSAGE_READ = 2;
public static final int MESSAGE_WRITE = 3;
public static final int MESSAGE_DEVICE_NAME = 4;
public static final int MESSAGE_TOAST = 5;
// Key names received from the BluetoothChatService Handler
public static final String DEVICE_NAME = "device_name";
public static final String TOAST = "toast";
// Intent request codes
private static final int REQUEST_CONNECT_DEVICE_SECURE = 1;
private static final int REQUEST_CONNECT_DEVICE_INSECURE = 2;
private static final int REQUEST_ENABLE_BT = 3;
// Layout Views
private TextView mTitle;
private ListView mConversationView;
private EditText mOutEditText;
private Button mSendButton;
private ToggleButton mOn1,mOn2,mOn3,mOn4,mOn5;

// Name of the connected device
private String mConnectedDeviceName = null;
// Array adapter for the conversation thread
private ArrayAdapter<String> mConversationArrayAdapter;
// String buffer for outgoing messages
private StringBuffer mOutStringBuffer;
// Local Bluetooth adapter
private BluetoothAdapter mBluetoothAdapter = null;
// Member object for the chat services
private BluetoothChatService mChatService = null;


@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if(D) Log.e(TAG, "+++ ON CREATE +++");

    // Set up the window layout
    requestWindowFeature(Window.FEATURE_CUSTOM_TITLE);
    setContentView(R.layout.main);
    getWindow().setFeatureInt(Window.FEATURE_CUSTOM_TITLE, R.layout.custom_title);

    // Set up the custom title
    mTitle = (TextView) findViewById(R.id.title_left_text);
    mTitle.setText(R.string.app_name);
    mTitle = (TextView) findViewById(R.id.title_right_text);

    // Get local Bluetooth adapter
    mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

    // If the adapter is null, then Bluetooth is not supported
    if (mBluetoothAdapter == null) {
        Toast.makeText(this, "Bluetooth no esta disponible", Toast.LENGTH_LONG).show();
        finish();
        return;
    }else{
        Toast.makeText(this, "Bluetooth disponible", Toast.LENGTH_LONG).show();
```

```
    }
}

@Override
public void onStart() {
    super.onStart();
    if(D) Log.e(TAG, "++ ON START ++");

    // If BT is not on, request that it be enabled.
    // setupChat() will then be called during onActivityResult
    if (!mBluetoothAdapter.isEnabled()) {
        Intent enableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
    // Otherwise, setup the chat session
    } else {
        if (mChatService == null) setupChat();
    }
}

@Override
public synchronized void onResume() {
    super.onResume();
    if(D) Log.e(TAG, "+ ON RESUME +");

    // Performing this check in onResume() covers the case in which BT was
    // not enabled during onStart(), so we were paused to enable it...
    // onResume() will be called when ACTION_REQUEST_ENABLE activity returns.
    if (mChatService != null) {
        // Only if the state is STATE_NONE, do we know that we haven't started already
        if (mChatService.getState() == BluetoothChatService.STATE_NONE) {
          // Start the Bluetooth chat services
          mChatService.start();
        }
    }
}

private void setupChat() {
    Log.d(TAG, "setupChat()");

    // Initialize the array adapter for the conversation thread
    mConversationArrayAdapter = new ArrayAdapter<String>(this, R.layout.message);
    mConversationView = (ListView) findViewById(R.id.in);
    mConversationView.setAdapter(mConversationArrayAdapter);

    // Initialize the compose field with a listener for the return key
    mOutEditText = (EditText) findViewById(R.id.edit_text_out);
    mOutEditText.setOnEditorActionListener(mWriteListener);

    // Initialize the send button with a listener that for click events
    mSendButton = (Button) findViewById(R.id.button_send);

    mSendButton.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            // Send a message using content of the edit text widget
```

```
            TextView view = (TextView) findViewById(R.id.edit_text_out);
            String message = view.getText().toString();
            sendMessage(message);
        }
    });

    // Initialize the send button with a listener that for click events

    mOn1 = (ToggleButton) findViewById(R.id.toggleButton1);
    mOn1.setOnClickListener(new OnClickListener()
{
public void onClick(View v)
{
if (mOn1.getText().toString().equals("Sí")){
        try{
        //String message = "si1"+(char)13+(char)10;
                String message = "A"+(char)13;
        sendMessage(message);
        Toast.makeText(getBaseContext(),
                        "Interruptor 1 Encendido",
                        Toast.LENGTH_SHORT).show();
        }finally{

        }
}else{
        try{
                //String message = "no1"+(char)13+(char)10;
                String message = "a"+(char)13;
        sendMessage(message);
        Toast.makeText(getBaseContext(),
                        "Interruptor 1 Apagado",
                        Toast.LENGTH_SHORT).show();
}finally{
                        }}                  }       });

    mOn2 = (ToggleButton) findViewById(R.id.toggleButton2);
    mOn2.setOnClickListener(new OnClickListener()
{
public void onClick(View v)
{
if (mOn2.getText().toString().equals("Sí")){
        try{
        //String message = "si2"+(char)13+(char)10;
                String message = "B"+(char)13;
                sendMessage(message);
        Toast.makeText(getBaseContext(),
                        "Interruptor 2 Encendido",
                        Toast.LENGTH_SHORT).show();
        }finally{

        }
}else{
        try{
                //String message = "no2"+(char)13+(char)10;
                String message = "b"+(char)13;
```

```
        sendMessage(message);
        Toast.makeText(getBaseContext(),
                        "Interruptor 2 Apagado",
                        Toast.LENGTH_SHORT).show();
}finally{
                        }}                      }       });

      mOn3 = (ToggleButton) findViewById(R.id.toggleButton3);
      mOn3.setOnClickListener(new OnClickListener()
{
public void onClick(View v)
{
if (mOn3.getText().toString().equals("Sí")){
        try{
        //String message = "si3"+(char)13+(char)10;
                String message = "C"+(char)13;
                sendMessage(message);
        Toast.makeText(getBaseContext(),
                        "Interruptor 3 Encendido",
                        Toast.LENGTH_SHORT).show();
        }finally{
                        }
}else{
        try{
                //String message = "no3"+(char)13+(char)10;
                String message = "c"+(char)13;
        sendMessage(message);
        Toast.makeText(getBaseContext(),
                        "Interruptor 3 Apagado",
                        Toast.LENGTH_SHORT).show();
}finally{
                        }}                      }       });
      mOn4 = (ToggleButton) findViewById(R.id.toggleButton4);
      mOn4.setOnClickListener(new OnClickListener()
{
public void onClick(View v)
{
if (mOn4.getText().toString().equals("Sí")){
        try{
        //String message = "si4"+(char)13+(char)10;
                String message = "D"+(char)13;
                sendMessage(message);
        Toast.makeText(getBaseContext(),
                        "Interruptor 4 Encendido",
                        Toast.LENGTH_SHORT).show();
        }finally{

        }
}else{
        try{
                //String message = "no4"+(char)13+(char)10;
                String message = "d"+(char)13;
        sendMessage(message);
        Toast.makeText(getBaseContext(),
                        "Interruptor 4 Apagado",
```

```
                        Toast.LENGTH_SHORT).show();
}finally{

        }}                  }       });

      mOn5 = (ToggleButton) findViewById(R.id.toggleButton5);
      mOn5.setOnClickListener(new OnClickListener()
{
public void onClick(View v)
{
if (mOn5.getText().toString().equals("Sí")){
        try{
        //String message = "si5"+(char)13+(char)10;
                String message = "E"+(char)13;
                sendMessage(message);
        Toast.makeText(getBaseContext(),
                        "Interruptor 5 Encendido",
                        Toast.LENGTH_SHORT).show();
        }finally{
        }
}else{
        try{
                //String message = "no5"+(char)13+(char)10;
                String message = "e"+(char)13;
        sendMessage(message);
        Toast.makeText(getBaseContext(),
                        "Interruptor 5 Apagado",
                        Toast.LENGTH_SHORT).show();
}finally{
        }}                  }       });

      // Initialize the BluetoothChatService to perform bluetooth connections
      mChatService = new BluetoothChatService(this, mHandler);

      // Initialize the buffer for outgoing messages
      mOutStringBuffer = new StringBuffer("");
   }

   @Override
   public synchronized void onPause() {
      super.onPause();
      if(D) Log.e(TAG, "- ON PAUSE -");
   }

   @Override
   public void onStop() {
      super.onStop();
      if(D) Log.e(TAG, "-- ON STOP --");
   }

   @Override
   public void onDestroy() {
      super.onDestroy();
      // Stop the Bluetooth chat services
      if (mChatService != null) mChatService.stop();
```

```
        if(D) Log.e(TAG, "--- ON DESTROY ---");
    }

    private void ensureDiscoverable() {
        if(D) Log.d(TAG, "ensure discoverable");
        if (mBluetoothAdapter.getScanMode() !=
            BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE) {
            Intent discoverableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
            discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION,
300);
            startActivity(discoverableIntent);
        }
    }

    /**
     * Sends a message.
     * @param message  A string of text to send.
     */
    private void sendMessage(String message) {
        // Check that we're actually connected before trying anything
        if (mChatService.getState() != BluetoothChatService.STATE_CONNECTED) {
            Toast.makeText(this, R.string.not_connected, Toast.LENGTH_SHORT).show();
            return;
        }

        // Check that there's actually something to send
        if (message.length() > 0) {
            // Get the message bytes and tell the BluetoothChatService to write
            byte[] send = message.getBytes();
            mChatService.write(send);

            // Reset out string buffer to zero and clear the edit text field
            mOutStringBuffer.setLength(0);
            mOutEditText.setText(mOutStringBuffer);
        }
    }

    // The action listener for the EditText widget, to listen for the return key
    private TextView.OnEditorActionListener mWriteListener =
        new TextView.OnEditorActionListener() {
        public boolean onEditorAction(TextView view, int actionId, KeyEvent event) {
            // If the action is a key-up event on the return key, send the message
            if (actionId == EditorInfo.IME_NULL && event.getAction() == KeyEvent.ACTION_UP) {
                String message = view.getText().toString();
                sendMessage(message);
            }
            if(D) Log.i(TAG, "END onEditorAction");
            return true;
        }
    };

    // The Handler that gets information back from the BluetoothChatService
    private final Handler mHandler = new Handler() {
        @Override
```

```
    public void handleMessage(Message msg) {
        switch (msg.what) {
        case MESSAGE_STATE_CHANGE:
            if(D) Log.i(TAG, "MESSAGE_STATE_CHANGE: " + msg.arg1);
            switch (msg.arg1) {
            case BluetoothChatService.STATE_CONNECTED:
                mTitle.setText(R.string.title_connected_to);
                mTitle.append(mConnectedDeviceName);
                mConversationArrayAdapter.clear();
                break;
            case BluetoothChatService.STATE_CONNECTING:
                mTitle.setText(R.string.title_connecting);
                break;
            case BluetoothChatService.STATE_LISTEN:
            case BluetoothChatService.STATE_NONE:
                mTitle.setText(R.string.title_not_connected);
                break;
            }
            break;
        case MESSAGE_WRITE:
            byte[] writeBuf = (byte[]) msg.obj;
            // construct a string from the buffer
            String writeMessage = new String(writeBuf);
            mConversationArrayAdapter.add("Me:  " + writeMessage);
            break;
        case MESSAGE_READ:
            byte[] readBuf = (byte[]) msg.obj;
            // construct a string from the valid bytes in the buffer
            String readMessage = new String(readBuf, 0, msg.arg1);
            mConversationArrayAdapter.add(mConnectedDeviceName+":  " + readMessage);
            break;
        case MESSAGE_DEVICE_NAME:
            // save the connected device's name
            mConnectedDeviceName = msg.getData().getString(DEVICE_NAME);
            Toast.makeText(getApplicationContext(), "Connected to "
                           + mConnectedDeviceName, Toast.LENGTH_SHORT).show();
            break;
        case MESSAGE_TOAST:
            Toast.makeText(getApplicationContext(), msg.getData().getString(TOAST),
                           Toast.LENGTH_SHORT).show();
            break;
        }
    }
};

public void onActivityResult(int requestCode, int resultCode, Intent data) {
    if(D) Log.d(TAG, "onActivityResult " + resultCode);
    switch (requestCode) {
    case REQUEST_CONNECT_DEVICE_SECURE:
        // When DeviceListActivity returns with a device to connect
        if (resultCode == Activity.RESULT_OK) {
            connectDevice(data, true);
        }
        break;
    case REQUEST_CONNECT_DEVICE_INSECURE:
```

```
        // When DeviceListActivity returns with a device to connect
        if (resultCode == Activity.RESULT_OK) {
            connectDevice(data, false);
        }
        break;
    case REQUEST_ENABLE_BT:
        // When the request to enable Bluetooth returns
        if (resultCode == Activity.RESULT_OK) {
            // Bluetooth is now enabled, so set up a chat session
            setupChat();
        } else {
            // User did not enable Bluetooth or an error occured
            Log.d(TAG, "BT not enabled");
            Toast.makeText(this, R.string.bt_not_enabled_leaving, Toast.LENGTH_SHORT).show();
            finish();
        }
    }
}

private void connectDevice(Intent data, boolean secure) {
    // Get the device MAC address
    String address = data.getExtras()
        .getString(DeviceListActivity.EXTRA_DEVICE_ADDRESS);
    // Get the BLuetoothDevice object
    BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);
    // Attempt to connect to the device
    mChatService.connect(device, secure);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.option_menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    Intent serverIntent = null;
    switch (item.getItemId()) {
    case R.id.secure_connect_scan:
        // Launch the DeviceListActivity to see devices and do scan
        serverIntent = new Intent(this, DeviceListActivity.class);
        startActivityForResult(serverIntent, REQUEST_CONNECT_DEVICE_SECURE);
        return true;
    case R.id.insecure_connect_scan:
        // Launch the DeviceListActivity to see devices and do scan
        serverIntent = new Intent(this, DeviceListActivity.class);
        startActivityForResult(serverIntent, REQUEST_CONNECT_DEVICE_INSECURE);
        return true;
    case R.id.discoverable:
        // Ensure this device is discoverable by others
        ensureDiscoverable();
        return true;        }
    return false;    }}
```

BLUETOOTH CHAT SERVICE

```
package com.example.android.BluetoothChat;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.UUID;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothServerSocket;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;

/**
 * This class does all the work for setting up and managing Bluetooth
 * connections with other devices. It has a thread that listens for
 * incoming connections, a thread for connecting with a device, and a
 * thread for performing data transmissions when connected.
 */
public class BluetoothChatService {
    // Debugging
    private static final String TAG = "BluetoothChatService";
    private static final boolean D = true;

    // Name for the SDP record when creating server socket
    private static final String NAME_SECURE = "BluetoothChatSecure";
    private static final String NAME_INSECURE = "BluetoothChatInsecure";

    // Unique UUID for this application
    private static final UUID MY_UUID_SECURE =
        UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    private static final UUID MY_UUID_INSECURE =
        UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

    // Member fields
    private final BluetoothAdapter mAdapter;
    private final Handler mHandler;
    private AcceptThread mSecureAcceptThread;
    private AcceptThread mInsecureAcceptThread;
    private ConnectThread mConnectThread;
    private ConnectedThread mConnectedThread;
    private int mState;

    // Constants that indicate the current connection state
    public static final int STATE_NONE = 0;       // we're doing nothing
    public static final int STATE_LISTEN = 1;     // now listening for incoming connections
    public static final int STATE_CONNECTING = 2; // now initiating an outgoing connection
    public static final int STATE_CONNECTED = 3;  // now connected to a remote device

    /**
```

```
     * Constructor. Prepares a new BluetoothChat session.
     * @param context  The UI Activity Context
     * @param handler  A Handler to send messages back to the UI Activity
     */
    public BluetoothChatService(Context context, Handler handler) {
        mAdapter = BluetoothAdapter.getDefaultAdapter();
        mState = STATE_NONE;
        mHandler = handler;
    }

    /**
     * Set the current state of the chat connection
     * @param state  An integer defining the current connection state
     */
    private synchronized void setState(int state) {
        if (D) Log.d(TAG, "setState() " + mState + " -> " + state);
        mState = state;

        // Give the new state to the Handler so the UI Activity can update
        mHandler.obtainMessage(BluetoothChat.MESSAGE_STATE_CHANGE, state,
-1).sendToTarget();
    }

    /**
     * Return the current connection state. */
    public synchronized int getState() {
        return mState;
    }

    /**
     * Start the chat service. Specifically start AcceptThread to begin a
     * session in listening (server) mode. Called by the Activity onResume() */
    public synchronized void start() {
        if (D) Log.d(TAG, "start");

        // Cancel any thread attempting to make a connection
        if (mConnectThread != null) {mConnectThread.cancel(); mConnectThread = null;}

        // Cancel any thread currently running a connection
        if (mConnectedThread != null) {mConnectedThread.cancel(); mConnectedThread = null;}

        setState(STATE_LISTEN);

        // Start the thread to listen on a BluetoothServerSocket
        if (mSecureAcceptThread == null) {
            mSecureAcceptThread = new AcceptThread(true);
            mSecureAcceptThread.start();
        }
        if (mInsecureAcceptThread == null) {
            mInsecureAcceptThread = new AcceptThread(false);
            mInsecureAcceptThread.start();
        }
    }

    /**
```

```
 * Start the ConnectThread to initiate a connection to a remote device.
 * @param device  The BluetoothDevice to connect
 * @param secure Socket Security type - Secure (true) , Insecure (false)
 */
public synchronized void connect(BluetoothDevice device, boolean secure) {
    if (D) Log.d(TAG, "Estas conectado a: " + device);

    // Cancel any thread attempting to make a connection
    if (mState == STATE_CONNECTING) {
        if (mConnectThread != null) {mConnectThread.cancel(); mConnectThread = null;}
    }

    // Cancel any thread currently running a connection
    if (mConnectedThread != null) {mConnectedThread.cancel(); mConnectedThread = null;}

    // Start the thread to connect with the given device
    mConnectThread = new ConnectThread(device, secure);
    mConnectThread.start();
    setState(STATE_CONNECTING);
}

/**
 * Start the ConnectedThread to begin managing a Bluetooth connection
 * @param socket  The BluetoothSocket on which the connection was made
 * @param device  The BluetoothDevice that has been connected
 */
public synchronized void connected(BluetoothSocket socket, BluetoothDevice
        device, final String socketType) {
    if (D) Log.d(TAG, "connected, Socket Type:" + socketType);

    // Cancel the thread that completed the connection
    if (mConnectThread != null) {mConnectThread.cancel(); mConnectThread = null;}

    // Cancel any thread currently running a connection
    if (mConnectedThread != null) {mConnectedThread.cancel(); mConnectedThread = null;}

    // Cancel the accept thread because we only want to connect to one device
    if (mSecureAcceptThread != null) {
        mSecureAcceptThread.cancel();
        mSecureAcceptThread = null;
    }
    if (mInsecureAcceptThread != null) {
        mInsecureAcceptThread.cancel();
        mInsecureAcceptThread = null;
    }

    // Start the thread to manage the connection and perform transmissions
    mConnectedThread = new ConnectedThread(socket, socketType);
    mConnectedThread.start();

    // Send the name of the connected device back to the UI Activity
    Message msg = mHandler.obtainMessage(BluetoothChat.MESSAGE_DEVICE_NAME);
    Bundle bundle = new Bundle();
    bundle.putString(BluetoothChat.DEVICE_NAME, device.getName());
    msg.setData(bundle);
```

```
    mHandler.sendMessage(msg);

    setState(STATE_CONNECTED);
}

/**
 * Stop all threads
 */
public synchronized void stop() {
    if (D) Log.d(TAG, "stop");

    if (mConnectThread != null) {
        mConnectThread.cancel();
        mConnectThread = null;
    }

    if (mConnectedThread != null) {
        mConnectedThread.cancel();
        mConnectedThread = null;
    }

    if (mSecureAcceptThread != null) {
        mSecureAcceptThread.cancel();
        mSecureAcceptThread = null;
    }

    if (mInsecureAcceptThread != null) {
        mInsecureAcceptThread.cancel();
        mInsecureAcceptThread = null;
    }
    setState(STATE_NONE);
}
 * Write to the ConnectedThread in an unsynchronized manner
 * @param out The bytes to write
 * @see ConnectedThread#write(byte[])
 */
public void write(byte[] out) {
    // Create temporary object
    ConnectedThread r;
    // Synchronize a copy of the ConnectedThread
    synchronized (this) {
        if (mState != STATE_CONNECTED) return;
        r = mConnectedThread;
    }
    // Perform the write unsynchronized
    r.write(out);
}

/**
 * Indicate that the connection attempt failed and notify the UI Activity.
 */
private void connectionFailed() {
    // Send a failure message back to the Activity
    Message msg = mHandler.obtainMessage(BluetoothChat.MESSAGE_TOAST);
    Bundle bundle = new Bundle();
```

```
    bundle.putString(BluetoothChat.TOAST, "Unable to connect device");
    msg.setData(bundle);
    mHandler.sendMessage(msg);

    // Start the service over to restart listening mode
    BluetoothChatService.this.start();
}

/**
 * Indicate that the connection was lost and notify the UI Activity.
 */
private void connectionLost() {
    // Send a failure message back to the Activity
    Message msg = mHandler.obtainMessage(BluetoothChat.MESSAGE_TOAST);
    Bundle bundle = new Bundle();
    bundle.putString(BluetoothChat.TOAST, "Device connection was lost");
    msg.setData(bundle);
    mHandler.sendMessage(msg);

    // Start the service over to restart listening mode
    BluetoothChatService.this.start();
}
private class AcceptThread extends Thread {
    // The local server socket
    private final BluetoothServerSocket mmServerSocket;
    private String mSocketType;

    public AcceptThread(boolean secure) {
        BluetoothServerSocket tmp = null;
        mSocketType = secure ? "Secure":"Insecure";

        // Create a new listening server socket
        try {
            if (secure) {
                tmp = mAdapter.listenUsingRfcommWithServiceRecord(NAME_SECURE,
                    MY_UUID_SECURE);
            } else {
                tmp = mAdapter.listenUsingInsecureRfcommWithServiceRecord(
                        NAME_INSECURE, MY_UUID_INSECURE);
            }
        } catch (IOException e) {
            Log.e(TAG, "Socket Type: " + mSocketType + "listen() failed", e);
        }
        mmServerSocket = tmp;
    }

    public void run() {
        if (D) Log.d(TAG, "Socket Type: " + mSocketType +
                "BEGIN mAcceptThread" + this);
        setName("AcceptThread" + mSocketType);

        BluetoothSocket socket = null;

        // Listen to the server socket if we're not connected
        while (mState != STATE_CONNECTED) {
```

```
            try {
                // This is a blocking call and will only return on a
                // successful connection or an exception
                socket = mmServerSocket.accept();
            } catch (IOException e) {
                Log.e(TAG, "Socket Type: " + mSocketType + "accept() failed", e);
                break;
            }

            // If a connection was accepted
            if (socket != null) {
                synchronized (BluetoothChatService.this) {
                    switch (mState) {
                    case STATE_LISTEN:
                    case STATE_CONNECTING:
                        // Situation normal. Start the connected thread.
                        connected(socket, socket.getRemoteDevice(),
                                mSocketType);
                        break;
                    case STATE_NONE:
                    case STATE_CONNECTED:
                        // Either not ready or already connected. Terminate new socket.
                        try {
                            socket.close();
                        } catch (IOException e) {
                            Log.e(TAG, "Could not close unwanted socket", e);
                        }
                        break;
                    }                }              }         }
        if (D) Log.i(TAG, "END mAcceptThread, socket Type: " + mSocketType);
    }
    public void cancel() {
        if (D) Log.d(TAG, "Socket Type" + mSocketType + "cancel " + this);
        try {
            mmServerSocket.close();
        } catch (IOException e) {
            Log.e(TAG, "Socket Type" + mSocketType + "close() of server failed", e);
        }       }    }
private class ConnectThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final BluetoothDevice mmDevice;
    private String mSocketType;

    public ConnectThread(BluetoothDevice device, boolean secure) {
        mmDevice = device;
        BluetoothSocket tmp = null;
        mSocketType = secure ? "Secure" : "Insecure";

        // Get a BluetoothSocket for a connection with the
        // given BluetoothDevice
        try {
            if (secure) {
                tmp = device.createRfcommSocketToServiceRecord(
                        MY_UUID_SECURE);
            } else {
```

```
                tmp = device.createInsecureRfcommSocketToServiceRecord(
                        MY_UUID_INSECURE);
            }
        } catch (IOException e) {
            Log.e(TAG, "Socket Type: " + mSocketType + "create() failed", e);
        }
        mmSocket = tmp;
    }
    public void run() {
        Log.i(TAG, "BEGIN mConnectThread SocketType:" + mSocketType);
        setName("ConnectThread" + mSocketType);

        // Always cancel discovery because it will slow down a connection
        mAdapter.cancelDiscovery();

        // Make a connection to the BluetoothSocket
        try {
            // This is a blocking call and will only return on a
            // successful connection or an exception
            mmSocket.connect();
        } catch (IOException e) {
            // Close the socket
            try {
                mmSocket.close();
            } catch (IOException e2) {
                Log.e(TAG, "unable to close() " + mSocketType +
                        " socket during connection failure", e2);
            }
            connectionFailed();
            return;
        }

        // Reset the ConnectThread because we're done
        synchronized (BluetoothChatService.this) {
            mConnectThread = null;
        }
        // Start the connected thread
        connected(mmSocket, mmDevice, mSocketType);
    }
    public void cancel() {
        try {
            mmSocket.close();
        } catch (IOException e) {
            Log.e(TAG, "close() of connect " + mSocketType + " socket failed", e);
        }       }    }
/**
 * This thread runs during a connection with a remote device.
 * It handles all incoming and outgoing transmissions.
 */
private class ConnectedThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final InputStream mmInStream;
    private final OutputStream mmOutStream;

    public ConnectedThread(BluetoothSocket socket, String socketType) {
```

```
    Log.d(TAG, "create ConnectedThread: " + socketType);
    mmSocket = socket;
    InputStream tmpIn = null;
    OutputStream tmpOut = null;

    // Get the BluetoothSocket input and output streams
    try {
        tmpIn = socket.getInputStream();
        tmpOut = socket.getOutputStream();
    } catch (IOException e) {
        Log.e(TAG, "temp sockets not created", e);
    }
    mmInStream = tmpIn;
    mmOutStream = tmpOut;
}
public void run() {
    Log.i(TAG, "BEGIN mConnectedThread");
    byte[] buffer = new byte[1024];
    int bytes;

    // Keep listening to the InputStream while connected
    while (true) {
        try {
            // Read from the InputStream
            bytes = mmInStream.read(buffer);

            // Send the obtained bytes to the UI Activity
            mHandler.obtainMessage(BluetoothChat.MESSAGE_READ, bytes, -1, buffer)
                    .sendToTarget();
        } catch (IOException e) {
            Log.e(TAG, "disconnected", e);
            connectionLost();
            break;
        }           }       }
/**
 * Write to the connected OutStream.
 * @param buffer  The bytes to write
 */
public void write(byte[] buffer) {
    try {
        mmOutStream.write(buffer);

        // Share the sent message back to the UI Activity
        mHandler.obtainMessage(BluetoothChat.MESSAGE_WRITE, -1, -1, buffer)
                .sendToTarget();
    } catch (IOException e) {
        Log.e(TAG, "Exception during write", e);
    }
}

public void cancel() {
    try {
        mmSocket.close();
    } catch (IOException e) {
        Log.e(TAG, "close() of connect socket failed", e);
```

```
        }       }   }}
```

DEVICE LIST ACTIVITY

```
package com.example.android.BluetoothChat;

import java.util.Set;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;

/**
 * This Activity appears as a dialog. It lists any paired devices and
 * devices detected in the area after discovery. When a device is chosen
 * by the user, the MAC address of the device is sent back to the parent
 * Activity in the result Intent.
 */
public class DeviceListActivity extends Activity {
    // Debugging
    private static final String TAG = "DeviceListActivity";
    private static final boolean D = true;

    // Return Intent extra
    public static String EXTRA_DEVICE_ADDRESS = "device_address";

    // Member fields
    private BluetoothAdapter mBtAdapter;
    private ArrayAdapter<String> mPairedDevicesArrayAdapter;
    private ArrayAdapter<String> mNewDevicesArrayAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Setup the window
        requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
        setContentView(R.layout.device_list);

        // Set result CANCELED incase the user backs out
```

```
    setResult(Activity.RESULT_CANCELED)
    // Initialize the button to perform device discovery
    Button scanButton = (Button) findViewById(R.id.button_scan);
    scanButton.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            doDiscovery();
            v.setVisibility(View.GONE);
        }       });
    // Initialize array adapters. One for already paired devices and
    // one for newly discovered devices
    mPairedDevicesArrayAdapter = new ArrayAdapter<String>(this, R.layout.device_name);
    mNewDevicesArrayAdapter = new ArrayAdapter<String>(this, R.layout.device_name);

    // Find and set up the ListView for paired devices
    ListView pairedListView = (ListView) findViewById(R.id.paired_devices);
    pairedListView.setAdapter(mPairedDevicesArrayAdapter);
    pairedListView.setOnItemClickListener(mDeviceClickListener);

    // Find and set up the ListView for newly discovered devices
    ListView newDevicesListView = (ListView) findViewById(R.id.new_devices);
    newDevicesListView.setAdapter(mNewDevicesArrayAdapter);
    newDevicesListView.setOnItemClickListener(mDeviceClickListener);

    // Register for broadcasts when a device is discovered
    IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
    this.registerReceiver(mReceiver, filter);

    // Register for broadcasts when discovery has finished
    filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
    this.registerReceiver(mReceiver, filter);

    // Get the local Bluetooth adapter
    mBtAdapter = BluetoothAdapter.getDefaultAdapter();

    // Get a set of currently paired devices
    Set<BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();

    // If there are paired devices, add each one to the ArrayAdapter
    if (pairedDevices.size() > 0) {
        findViewById(R.id.title_paired_devices).setVisibility(View.VISIBLE);
        for (BluetoothDevice device : pairedDevices) {
            mPairedDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
        }
    } else {
        String noDevices = getResources().getText(R.string.none_paired).toString();
        mPairedDevicesArrayAdapter.add(noDevices);
    }
}

@Override
protected void onDestroy() {
    super.onDestroy()
    // Make sure we're not doing discovery anymore
    if (mBtAdapter != null) {
        mBtAdapter.cancelDiscovery();
```

```
    }
    // Unregister broadcast listeners
    this.unregisterReceiver(mReceiver);
}
/**
 * Start device discover with the BluetoothAdapter
 */
private void doDiscovery() {
    if (D) Log.d(TAG, "doDiscovery()");

    // Indicate scanning in the title
    setProgressBarIndeterminateVisibility(true);
    setTitle(R.string.scanning);

    // Turn on sub-title for new devices
    findViewById(R.id.title_new_devices).setVisibility(View.VISIBLE);

    // If we're already discovering, stop it
    if (mBtAdapter.isDiscovering()) {
        mBtAdapter.cancelDiscovery();
    }

    // Request discover from BluetoothAdapter
    mBtAdapter.startDiscovery();
}

// The on-click listener for all devices in the ListViews
private OnItemClickListener mDeviceClickListener = new OnItemClickListener() {
    public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3) {
        // Cancel discovery because it's costly and we're about to connect
        mBtAdapter.cancelDiscovery();

        // Get the device MAC address, which is the last 17 chars in the View
        String info = ((TextView) v).getText().toString();
        String address = info.substring(info.length() - 17);

        // Create the result Intent and include the MAC address
        Intent intent = new Intent();
        intent.putExtra(EXTRA_DEVICE_ADDRESS, address);

        // Set result and finish this Activity
        setResult(Activity.RESULT_OK, intent);
        finish();
    } };

// The BroadcastReceiver that listens for discovered devices and
// changes the title when discovery is finished
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        // When discovery finds a device
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            // Get the BluetoothDevice object from the Intent
```

```
    BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
    // If it's already paired, skip it, because it's been listed already
    if (device.getBondState() != BluetoothDevice.BOND_BONDED) {
        mNewDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
    }
// When discovery is finished, change the Activity title
} else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
    setProgressBarIndeterminateVisibility(false);
    setTitle(R.string.select_device);
    if (mNewDevicesArrayAdapter.getCount() == 0) {
        String noDevices = getResources().getText(R.string.none_found).toString();
        mNewDevicesArrayAdapter.add(noDevices);
    }          }       }    };}
```

- Comprobar el funcionamiento entre el teléfono celular y el Minirobot.

Se establece una comunicación óptima.

6. DIAGRAMAS Y FIGURAS.

Figura 6.2: Práctica N°9, pantalla de control.

7. TABULACIÓN Y RESULTADOS.

ARCHIVO	ESTADO
ADELANTE	CORECTO
ATRAS	CORECTO
DERECHA	CORECTO
IZQUIERDA	CORECTO
PARAR	CORECTO

Tabla 7.1: Funcionamiento del archivo ejecutable .APK.

8. CONCLUSIONES.

- Para la implementación se utilizó un módulo BT de clase 2 y su alcance máximo es de 10 metros en linea de vista.

- La programación en Eclipse se la realiza en base clases y sólo se crea una aplicación ejecutable.

- La comunicación se realizó con éxito, entre el módulo y el teléfono celular

9. BIBLIOGRAFÍA.

- http://sourceforge.net/projects/bluetoothled/files/

- http://es.wikipedia.org/wiki/Android.

- http://alejandro-esparza-tudela.suite101.net/que-es-android-en-celulares-a61379.

10. ANEXOS.

PRÁCTICA N° 10
CONTROL DE LAS INSTALACIONES DE UN DEPARTAMENTO A TRAVÉS DE UN TELÉFONO CELULAR CON BLUETOOTH

INTEGRANTES:

FECHA DE INICIO:

FECHA DE ENTREGA:

1. TITULO:

Control de las instalaciones de un departamento a través de un teléfono celular con bluetooth.

2. OBJETIVOS

2.1. OBETIVO GENERAL

Diseñar e implementar el control de las instalaciones de un departamento por bluetooth a través de J2ME.

2.2. OBJETIVOS ESPECÍFICOS

- Desarrollar una interfaz gráfica, como aplicación java, para un teléfono celular capaz de controlar tres áreas del departamento.
- Comprobar el funcionamiento de la comunicación inalámbrica entre el teléfono celular y las áreas del departamento.

3. MARCO TEÓRICO

BLUETOOTH

La tecnología Bluetooth es una especificación abierta para la comunicación inalámbrica (WIRELESS) de datos y voz. Está basada en un enlace de radio de bajo costo y corto alcance, implementado en un circuito integrado de 9 x 9 mm, proporcionando conexiones instantáneas (ad hoc) para entornos de comunicaciones

tanto móviles como estáticos. En definitiva, Bluetooth pretende ser una especificación global para la conectividad inalámbrica.

El principal objetivo de esta tecnología, es la posibilidad de reemplazar los muchos cables propietarios que conectan unos dispositivos con otros por medio de un enlace radio universal de corto alcance. Por ejemplo, la tecnología de radio Bluetooth implementada en el teléfono celular y en el ordenador portátil reemplazaría el molesto cable utilizado hoy en día para conectar ambos aparatos. Las impresoras, las agendas electrónicas, los PDA, los faxes, los teclados, los joysticks y prácticamente cualquier otro dispositivo digital son susceptibles de formar parte de un sistema Bluetooth.

Pero más allá de reemplazar, los incómodos cables, la tecnología Bluetooth ofrece un puente a las redes de datos existentes, una interfaz con el exterior y un mecanismo para formar en el momento, pequeños grupos de dispositivos conectados entre sí de forma privada fuera de cualquier estructura fija de red.

Integrado en un pequeño transmisor de radiofrecuencia que permite conectar entre sí todo tipo de dispositivos electrónicos (teléfonos, ordenadores, impresoras, faxes, etc) situados dentro de un radio limitado de 10 metros (ampliable a 100, aunque con mayor distorsión) sin necesidad de utilizar cables.

El transmisor está integrado en un pequeño microchip de 9x9 milímetros y opera en una frecuencia de banda global (2,4 GHz, utilizada en muchos países para usos médicos y científicos) que asegura la compatibilidad universal. Los dispositivos que incorporan Bluetooth se reconocen y se hablan de la misma forma que lo hace un ordenador con su impresora. El canal permanece abierto y no requiere la intervención directa y constante del usuario cada vez que se quiere enviar algo.

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

Las diferentes partes del sistema Bluetooth son:

- Una unidad de radio
- Una unidad de control del enlace

- Gestión del enlace
- Funciones software

4. LISTADO DE MATERIALES Y EQUIPOS.

- Entrenador Lógico.
- Minirobot.
- Teléfono celular.
- Cables de par trenzado.

5. PROCEDIMIENTO.

- Establecer las áreas del departamento a controlar.

Se controla tres áreas del del departamento que son:

1. Iluminación principal.
2. Control de la alarma de seguridad.
3. Sistema de ingreso automático.

- Realizar la programación

MIDLET BLUEHOME

```
import javax.bluetooth.ServiceRecord;
import javax.microedition.midlet.*;
 * @author JAVTER
public class BLUEHOME extends MIDlet implements Runnable {
    public void startApp() {
        new Portada(this);
        new Thread(this).start();
    }
    public void pauseApp() {
    }
    public void destroyApp(boolean unconditional) {
    }
    public void exitMidlet() {
        destroyApp(true);
        notifyDestroyed();
    }    public void run() {
```

```
        try {
            Thread.sleep(3000);
        } catch (InterruptedException ex) {
        }           Control bluetooth = new Control(this);
        bluetooth.makeConnections
     ("btspp://00195DEE405B:1;authenticate=false;encrypt=false;master=false",
      ServiceRecord.NOAUTHENTICATE_NOENCRYPT);

    }}
```

CLASE PORTADA

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.MIDlet;
public class Portada {
    private Display display;
    private MyCanvas canvas = new MyCanvas();
    private MIDlet midlet;
    public Portada(MIDlet midletIn) {
        midlet =  midletIn;
        display = Display.getDisplay(midlet);
        display.setCurrent(canvas);
    }
    public Display getDisplay() {
        return display;
    }
}
class MyCanvas extends Canvas {
    private Image image = null;
    public MyCanvas() {
        try {
            image = Image.createImage("/PORTADA.PNG");
        } catch (Exception error) {
       Alert alert = new Alert("Failure", "Can't open image file.", null, null);
            alert.setTimeout(Alert.FOREVER);
        }
    }
    protected void paint(Graphics graphics) {
        graphics.setColor(255,255,255);//RGB
        graphics.fillRect (0, 0, getWidth(), getHeight());
        if (image != null) {
            graphics.drawImage(image, getWidth() / 2, getHeight() / 2, Graphics.
                    HCENTER | Graphics.VCENTER);
        }
        setFullScreenMode(true);
    }
}
```

CLASE CONTROL

```
import javax.microedition.lcdui.*;
import java.util.Vector;
import javax.bluetooth.LocalDevice;
public class Control implements ConexionHandlerListener, CommandListener {
    private Alert alert;
    private Display display;
    private BLUEHOME mIDlet;
    private Command salir;
```

```
private List listaOpciones;
private Command atraz;
private final Vector handlers;
private volatile int numReceivedMessages = 0;
private volatile int numSentMessages = 0;
String receive = "";
private int maxConnections;
private int sendMessageId = 0;
private String sendcommand;
private String tipo="";
public Control(BLUEHOME mIDletIn) {
    init();
    mIDlet = mIDletIn;
    display = Display.getDisplay(mIDlet);
    display.setCurrent(listaOpciones);
    handlers = new Vector();
   String value =
        LocalDevice.getProperty(
                    "bluetooth.connected.devices.max");
    try
    {
        maxConnections = Integer.parseInt(value);
    }
    catch (NumberFormatException e)
    {
        maxConnections = 0;
    }
}
private void init() {
    salir = new Command("Salir", Command.EXIT, 2);
    atraz = new Command("Atrás", Command.BACK, 1);
    listaOpciones = new List("DOMO-HOME", List.IMPLICIT);
    listaOpciones.append("LUCES", null);
    listaOpciones.append("ALARMA", null);
    listaOpciones.append("SIST.INGRESO", null);
    listaOpciones.addCommand(salir);
    listaOpciones.setCommandListener(this);
}
public void presentarAlerta(String resultado) {
    if (resultado.startsWith("Error:")) {
        alert = new Alert("Alerta", resultado, null, AlertType.WARNING);
    } else if (resultado.startsWith("Ok:")) {
        alert = new Alert("Alerta", resultado, null, AlertType.INFO);
    }
    alert.setTimeout(Alert.FOREVER);
    alert.addCommand(new Command("Salir", Command.EXIT, 2));
    display.setCurrent(alert, listaOpciones);
}
public void commandAction(Command c, Displayable d) {
    if (c == listaOpciones.SELECT_COMMAND) {
        String label = listaOpciones.getString
              (listaOpciones.getSelectedIndex());
        //System.out.println(label);
        if (label.equals("LUCES")) {
            tipo="1";
```

```
                listaOpciones.deleteAll();
                listaOpciones.append("ON", null);
                listaOpciones.append("OFF", null);
                listaOpciones.addCommand(atraz);
            } else if (label.equals("ALARMA")) {
                tipo="2";
                listaOpciones.deleteAll();
                listaOpciones.append("ACTIVAR", null);
                listaOpciones.append("DESACTIVAR", null);
                listaOpciones.addCommand(atraz);
            } else if (label.equals("SIST.INGRESO")) {
                tipo="3";
                listaOpciones.deleteAll();
                listaOpciones.append("ABRIR", null);
                listaOpciones.addCommand(atraz);
            } else if (label.equals("ON")) {
                send("si"+tipo);
            } else if (label.equals("OFF")) {
                send("no"+tipo);
            }
            else if (label.equals("ABRIR")) {
                send("si"+tipo);
            }
            else if (label.equals("ACTIVAR")) {
                send("si"+tipo);
            }
            else if (label.equals("DESACTIVAR")) {
                send("NO"+tipo);
            }
        } else if (c == atraz) {
            listaOpciones.deleteAll();
            listaOpciones.removeCommand(atraz);
            listaOpciones.append("LUCES", null);
            listaOpciones.append("ALARMA", null);
            listaOpciones.append("SIST.INGRESO", null);
            listaOpciones.addCommand(salir);
            listaOpciones.setCommandListener(this);
        } else if (c == salir) {
            ((BLUEHOME) mIDlet).exitMidlet();
        }
    }
    public void handleOpen(ConexionHandler handler) {
        handlers.addElement(handler);
        // for the first open connection
        if (handlers.size() == 1) {
//            removeCommand(searchCommand);
//            removeCommand(sendCommand);
//            addCommand(sendCommand);
        }
        // Remove the 'Add connection' command
        // when the device already has open the
        // maximum number of connections it can
        // support.
        if (handlers.size() >= maxConnections) {
//            removeCommand(addConnectionCommand);
```

```
        }
        //texto2="Connection opened";
        String str = Integer.toString(handlers.size());
        //texto1=str;
        //fondoambiente=1;
        listaOpciones.append("Conectado", null);
    }
    public void handleOpenError(
            ConexionHandler handler,
            String errorMessage) {
   //midlet.noencontrado("El dispositivo no esta dentro del area de servicio");
        listaOpciones.append("Error de Conexion", null);
    }
    public void handleReceivedMessage(
            ConexionHandler handler,
            byte[] messageBytes) {
        numReceivedMessages++;
        String message = new String(messageBytes);
        receive = message;
        // texto2="# messages read: " + numReceivedMessages + " " +
        //     "sent: " + numSentMessages;
        if (message.substring(4, 5).equals("0")) {
            //   receive="off";
            //    foco.selFondo(0);
        }
        // Broadcast message to all clients
//        for (int i=0; i < handlers.size(); i++)
//        {
//            ConexionHandler h =
//                (ConexionHandler) handlers.elementAt(i);

//            Integer id = new Integer(sendMessageId++);
//            try
//            {
//                h.queueMessageForSending(id, messageBytes);
//            }
//            catch (IllegalArgumentException e)
//            {
//                String errorMessage =
//                    "IllegalArgumentException while trying to " +
//                    "send message: " + e.getMessage();
//                handleError(handler, errorMessage);
//            }
//        }
        listaOpciones.append("recibiendo datos", null);
    }
    public void handleQueuedMessageWasSent(
            ConexionHandler handler,
            Integer id) {
        numSentMessages++;
//texto2="# messages read: " +numReceivedMessages + " " +
        //"sent: " + numSentMessages;
    }
    public void handleClose(ConexionHandler handler) {
        removeHandler(handler);
```

```
        if (handlers.size() == 0) {
//          removeCommand(sendCommand);
//          addCommand(searchCommand);
        }
        // If the number of currently open connections
        // drops below the maximum number that this
        // device could have open, restore
        // 'Add connection' to the screen commands.
        if (handlers.size() < maxConnections) {
//          removeCommand(addConnectionCommand);
//          addCommand(addConnectionCommand);
        }
        //texto2="Connection closed";
        closeAll();
    }
    public void handleErrorClose(ConexionHandler handler,
            String errorMessage) {
        removeHandler(handler);
        if (handlers.size() == 0) {
//          removeCommand(sendCommand);
//          addCommand(searchCommand);
        }
     //   texto2 = "Error (close): '" + errorMessage + "'";
    }
    public void handleError(ConexionHandler handler,
            String errorMessage) {
        //texto2 = "Error: '" + errorMessage + "'";
    }
    public void makeConnections(String URL, int security) {
        ConexionHandler newHandler =
                new ConexionHandler(
                this,
                URL,
                security);
        newHandler.start(); // start reader & writer
    }
    private void removeHandler(ConexionHandler handler) {
        if (handlers.contains(handler)) {
            handlers.removeElement(handler);
            String str = Integer.toString(handlers.size());
          //  texto1 = str;
            if (handlers.size() == 0) {
            }
        }
    }
    private void send(String sendData) {
        Integer id = new Integer(sendMessageId++);
        for (int i = 0; i < handlers.size(); i++) {
            ConexionHandler handler =
                    (ConexionHandler) handlers.elementAt(i);
            try {
                handler.queueMessageForSending(
                        id,
                        sendData.getBytes());
            //    texto1 = "Queued a send message request";
```

```
            } catch (IllegalArgumentException e) {
                // Message length longer than
                // ServerConnectionHandler.MAX_MESSAGE_LENGTH
                String errorMessage =
                        "IllegalArgumentException while trying "
                        + "to send a message: " + e.getMessage();
                //handleError(handler, errorMessage);
            }
        }
    }
    void closeAll() {
        for (int i = 0; i < handlers.size(); i++) {
            ConexionHandler handler =
                    (ConexionHandler) handlers.elementAt(i);
            handler.close();
            removeHandler(handler);
        }
    }
}
```

CLASE CONEXIÓN HANDLER

```
import java.io.InputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.util.Hashtable;
import java.util.Enumeration;
import javax.bluetooth.ServiceRecord;
import javax.microedition.io.Connector;
import javax.microedition.io.StreamConnection;
import javax.microedition.io.StreamConnectionNotifier;
public class ConexionHandler
    implements Runnable
{
    private final static byte ZERO = (byte) '0';
    private final static int LENGTH_MAX_DIGITS = 5;

    // this is an arbitrarily chosen value:
    private final static int MAX_MESSAGE_LENGTH =
                        65536 - LENGTH_MAX_DIGITS;

    //private final ServiceRecord serviceRecord;
    private final int requiredSecurity;
    private final ConexionHandlerListener listener;
    private final Hashtable sendMessages = new Hashtable();

    private StreamConnection connection;
    private OutputStream out;
    private InputStream in;
    private volatile boolean aborting;
    private Writer writer;
    private String url;
    public ConexionHandler(
            ConexionHandlerListener listener,
            String URL,
            int requiredSecurity)
    {
```

```
        this.listener = listener;
        this.url=URL;
        //this.serviceRecord = serviceRecord;
        this.requiredSecurity = requiredSecurity;
        aborting = false;

        connection = null;
        out = null;
        in = null;
        listener = null;

        // the caller must call method 'start'
        // to start the reader and writer
    }
//    public ServiceRecord getServiceRecord()
//    {
//        return serviceRecord;
//    }
    public synchronized void start()
    {
        Thread thread = new Thread(this);
        thread.start();
    }
    public void close()
    {
        if (!aborting)
        {
            synchronized(this)
            {
                aborting = true;
            }

            synchronized(sendMessages)
            {
                sendMessages.notify();
            }

            if (out != null)
            {
                try
                {
                    out.close();
                    synchronized (this)
                    {
                        out = null;
                    }
                }
                catch(IOException e)
                {
                    // there is nothing we can do: ignore it
                }
            }

            if (in != null)
            {
```

```
            try
            {
                in.close();
                synchronized (this)
                {
                    in = null;
                }
            }
            catch(IOException e)
            {
                // there is nothing we can do: ignore it
            }
        }

        if (connection != null)
        {
            try
            {
                connection.close();
                synchronized (this)
                {
                    connection = null;
                }
            }
            catch (IOException e)
            {
                // there is nothing we can do: ignore it
            }
        }
    }
}
public void queueMessageForSending(Integer id, byte[] data)
{
    if (data.length > MAX_MESSAGE_LENGTH)
    {
        throw new IllegalArgumentException(
                    "Message too long: limit is " +
                    MAX_MESSAGE_LENGTH + " bytes");
    }

    synchronized(sendMessages)
    {
        sendMessages.put(id, data);
        sendMessages.notify();
    }
}
public void run()
{
    // the reader

    // 1. open the connection and streams, start the writer
    //String url = null;
    try
    {
        // 'must be master': false
```

```
            //url = serviceRecord.getConnectionURL(
            //                  requiredSecurity,
            //                  false);

            connection = (StreamConnection) Connector.open(url);
            in = connection.openInputStream();
            out = connection.openOutputStream();

//            LogScreen.log("Opened connection & streams to: '" +
//                    url + "'\n");

            // start the writer
            Writer writer = new Writer(this);
            Thread writeThread = new Thread(writer);
            writeThread.start();

//            LogScreen.log("Started a reader & writer for: '" +
//                    url + "'\n");

            // open succeeded, inform listener
            listener.handleOpen(this);
        }
        catch(IOException e)
        {
            // open failed, close any connections/streams, and
            // inform listener that the open failed

 //           LogScreen.log("Failed to open " +
 //                   "connection or streams for '" +
 //                    url + "' , Error: " +
 //                    e.getMessage());

            close();

            listener.handleOpenError(
                    this,
                    "IOException: '" + e.getMessage() + "'");

            return;
        }
        catch (SecurityException e)
        {
            // open failed, close any connections/streams, and
            // inform listener that the open failed

 //           LogScreen.log("Failed to open " +
 //                   "connection or streams for '" +
 //                    url + "' , Error: " +
 //                    e.getMessage());

            close();

            listener.handleOpenError(
                    this,
                    "SecurityException: '" + e.getMessage() + "'");
```

```
            return;
        }

        // 2. wait to receive and read messages
        while (!aborting)
        {
            int length = 0;
            try
            {
                byte[] lengthBuf = new byte[LENGTH_MAX_DIGITS];
                readFully(in, lengthBuf);
                length = readLength(lengthBuf);
                byte[] temp = new byte[length];
                readFully(in, temp);

                listener.handleReceivedMessage(this, temp);
            }
            catch (IOException e)
            {
                close();
                if (length == 0)
                {
                  listener.handleClose(this);
                }
                else
                {
                  // we were in the middle of reading...
                  listener.handleErrorClose(this, e.getMessage());
                }
            }
        }
    }
    private static void readFully(InputStream in, byte[] buffer)
        throws IOException
    {
        int bytesRead = 0;

        while (bytesRead < buffer.length)
        {
            int count = in.read(buffer,
                                bytesRead,
                                buffer.length - bytesRead);
            if (count == -1)
            {
                throw new IOException("Input stream closed");
            }
            bytesRead += count;
        }
    }
    private static int readLength(byte[] buffer)
    {
        int value = 0;

        for (int i = 0; i < LENGTH_MAX_DIGITS; ++i)
```

```
    {
        value *= 10;
        value += buffer[i] - ZERO;
    }
    return value;
}
private void sendMessage(OutputStream out, byte[] data)
    throws IOException
{
    int i;
    byte[] buf=new byte[2+data.length];
    for (i=0;i<data.length;i++){
        buf[i]=data[i];
    }
    buf[i]=13;
    buf[i+1]=10;
    out.write(buf);
    out.flush();
}
private static void writeLength(int value, byte[] buffer)
{
    for (int i = LENGTH_MAX_DIGITS -1; i >= 0; --i)
    {
        buffer[i] = (byte) (ZERO + value % 10);
        value = value / 10;
    }
}
private class Writer
    implements Runnable
{
    private final ConexionHandler handler;
    Writer(ConexionHandler handler)
    {
        this.handler = handler;
    }
    public void run()
    {
        while (!aborting)
        {
            synchronized(sendMessages)
            {
                Enumeration e = sendMessages.keys();
                if (e.hasMoreElements())
                {
                    // send any pending messages
                    Integer id = (Integer) e.nextElement();
                    byte[] sendData =
                          (byte[]) sendMessages.get(id);
                    try
                    {
                        sendMessage(out, sendData);

                        // remove sent message from queue
                        sendMessages.remove(id);
```

```
                    // inform listener that it was sent
                    listener.handleQueuedMessageWasSent(
                            handler,
                            id);
                }
                catch (IOException ex)
                {
                    close(); // stop the networking thread

                    // inform that we got an error close
                    listener.handleErrorClose(handler,
                                    ex.getMessage());
                }
            }

            if (sendMessages.isEmpty())
            {
                try
                {
                    sendMessages.wait();
                }
                catch (InterruptedException ex)
                {
                    // this can't happen in MIDP: ignore it
                }
            }
        }
    }
  }
}
```

CLASE CONEXIÓN HANDLER LISTENER

```
public interface ConexionHandlerListener
{
  // The handler's open succeeded. It can now be used for sending
  // and receiving messages.
  public void handleOpen(ConexionHandler handler);
  // The hadler's open failed. It has closed any connections or
  // streams that were open. The handler should not be used anymore,
  // and should be discarded.
  public void handleOpenError(ConexionHandler handler,
                  String errorMessage);
  public void handleQueuedMessageWasSent(ConexionHandler handler, Integer id);
  // The handler got an inbound message.
  public void handleReceivedMessage(
            ConexionHandler handler,
            byte[] messageBytes);
  // A message that had been previously queued for sending
  // (identified by id) by the handler, has been sent successfully.
  // The handler has closed its connections and streams.
  // The handler should not be used anymore, and should be discarded.
  // Only handlers which have previously called 'handleOpen' may
  // call 'handleClose', and only just once.
  public void handleClose(ConexionHandler handler);
  // An error related to the handler occurred. The handler
```

```
    // has closed the connection, and the handler should no
    // longer be used.
    public void handleErrorClose(ConexionHandler handler,
                                 String errorMessage);

}
```

- Comprobar el funcionamiento entre el teléfono celular y el Minirobot.

Se establece una comunicación óptima.

6. DIAGRAMAS Y FIGURAS.

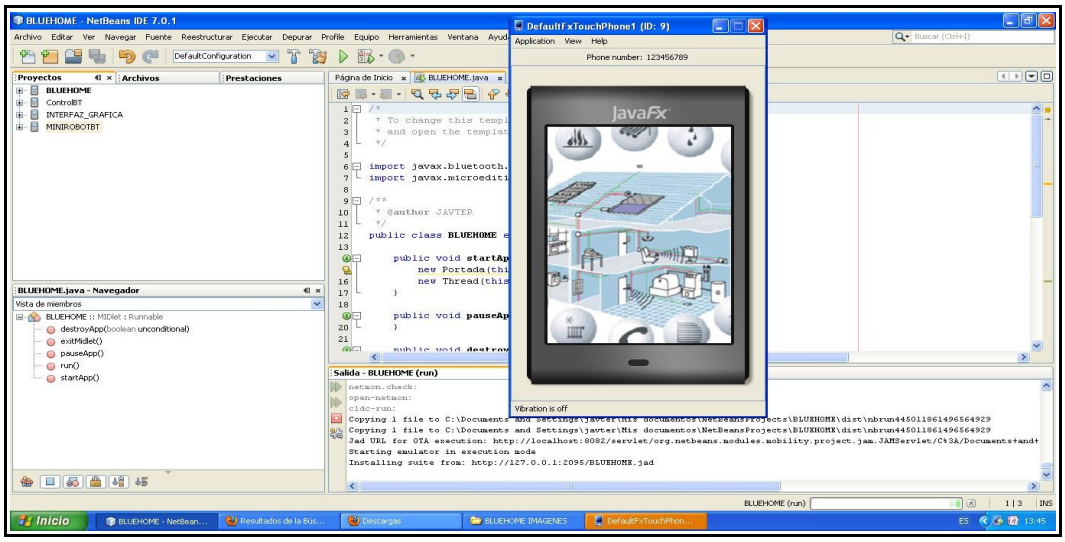


Figura 6.1: Práctica N°10, pantalla de inicio.

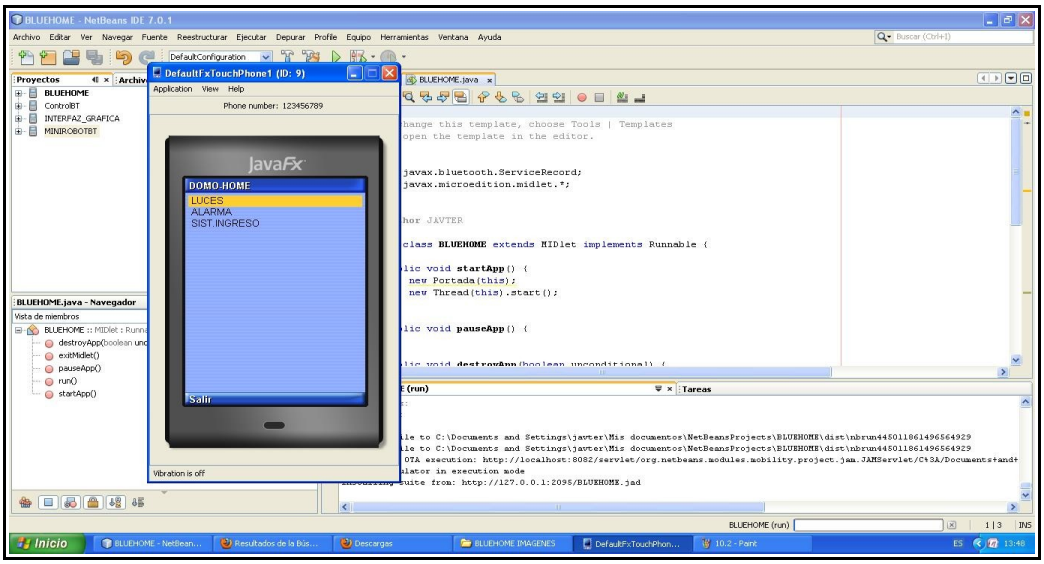


Figura 6.2: Práctica N°10, pantalla de control.

7. TABULACIÓN Y RESULTADOS.

ARCHIVO	ESTADO
LUCES	CORECTO
ALARMA	CORECTO
SIST.INGRESO	CORECTO

Tabla 7.1: Funcionamiento del archivo ejecutable .JAR.

8. CONCLUSIONES.

- Para la implementación se utilizó un módulo BT de clase 2 y su alcance máximo es de 10 metros en linea de vista.
- La programación en Netbeans se la realiza en base clases y solo se crea una aplicación ejecutable.
- La comunicación se realizó con éxito, entre el módulo y el teléfono celular

9. BIBLIOGRAFÍA.

- http://sourceforge.net/projects/bluetoothled/files/
- http://www.google.com/search?

client=ubuntu&channel=fs&q=comunicacion+bt+j2me&ie=utf-8&oe=utf-8#hl=es&client=ubuntu&hs=lcr&channel=fs&sa=X&ei=RQRAT_uxLMavg-wes2YSbCA&sqi=2&ved=0CBgQBSgA&q=comandos+de+comunicaci%C3%B3n+en+netbeans&spell=1&bav=on.2,or.r_gc.r_pw.,cf.osb&fp=141a0420e47bc3e0&biw=1368&bih=606

10. ANEXOS.

4.2 ANÁLISIS DEL DESARROLLO DE LAS PRÁCTICAS

La validación del Entrenador Lógico se realizó mediante la implementación de las 10 prácticas planteadas en las Guías con el siguiente análisis:

- Para el correcto funcionamiento de los PICs utilizados en las prácticas se necesitó un pulsador como MCLR, y desactivar los comparadores análogos.

- En las prácticas que se utilizaron pulsadores, se programaron con una subrutina de anti-rebote ya que al momento de la transmisión de datos se genera varias pulsaciones lo que provoca que los sistemas de transmisión y recepción funcionen en un ciclo de repetición durante aproximadamente ¼ de segundo generando respuestas incorrectas.

- Para la transmisión y recepción en los módulos de radiofrecuencia se realizó la programación con una subrutina de transmisión serial con mas de un caracter, a modo de clave, utilizando la sentencia "LOOKUP", dicha subrutina previene que, mediante interferencias generadas por dispositivos inalámbricos de cualquier frecuencia, activen o desactiven los sistemas de las prácticas planteadas en las Guías.

- Para los dispositivos móviles que soportan J2ME se utilizó siempre las clases "import javax.microedition.lcdui.* " y "import javax.microedition.midlet.*" que crean la pantalla principal de la interfaz gráfica y un archivo ejecutable que es de extensión ".jar".

- Para la creación de la interfaz gráfica en los dispositivos móviles Android se utilizó la misma programación java pero en el programa "Eclipse". Para J2ME y ANDROID es indispensable las clases "CONEXIÓN HANDLER " que es la que realiza un llamado a los métodos para el procesamiento de la información; y "CONEXIÓN HANDLER LISTENER " que es la que está a la espera de recibir los datos proporcionados por el módulo bluetooth para el emparejamiento de los dispositivos.

  Todas las pruebas realizadas en el Entrenador Lógico fueron avaladas por el tutor Ing. Mauricio Alminati durante el desarrollo de las pruebas del equipo.

CAPÍTULO V

ANÁLISIS FINANCIERO

5.1. ANÁLISIS F.O.D.A.

5.1.1. FORTALEZAS.

- Implemento electrónico de laboratorio nuevo en la Universidad Israel.
- Facilita el aprendizaje de comunicaciones inalámbricas a traves de J2ME y ANDROID.
- Agiliza el desarrollo de las prácticas de Diseño Electrónico.

5.1.2. OPORTUNIDADES.

- No existe alguna empresa en el Ecuador que fabrique Entrenadores para comunicaciones inalámbricas a traves de J2ME y ANDROID, ni guías desarrolladas, por lo que es una gran oportunidad el generar este producto en un mercado virgen.
- Existe una gran demanda por los estudiantes de la facultad de Electrónica de la Universidad Israel.

5.1.3. DEBILIDADES.

- Que la demanda supere la producción, ya que es un implemento importante a la hora de montar un proyecto de comunicaciones inalámbricas.
- Que no exista una guía profesional para la implementación de nuevos proyectos.

5.1.4. AMENAZAS.

- Que por la gran demanda, una empresa de mayor capacidad económica construya el mismo prototipo con un precio menor.
- No vender dos prototipos por semestre.
- Que los estudiantes no se interesen por los seminarios a impartirse.

5.2. ANÁLISIS DE MERCADO

El Análisis Financiero se realizó mediante etapas en las que se analiza:

- La competencia del producto.
- Clase económica de la población se va atacar según los resultados del mercado.

- Los objetivos para insertar el producto en el mercado.
- El costo del producto, el P.V.P. y el FODA del producto.

5.2.1 COMPETENCIA

Se realizó un sondeo de mercado, luego del cual se determinó que no existe competencia en Ecuador ya que ninguna empresa fabrica equipos de entrenamiento para comunicaciones inalámbricas. Para realizar un sondeo de la competencia se basó en dos tipos de análisis:

Análisis Externo

Se parte de un análisis del entorno, donde no existe demasiada inversión en el campo de entrenadores para comunicaciones inalámbricas, aunque existen mediadores en la red que facilitan el producto pero en diferentes equipos y en otros países.

Análisis Interno

Durante el análisis se determinó que no hay empresas en Ecuador dedicadas a la fabricación de equipos entrenadores que ayuden a desarrollar los conocimientos en comunicaciones inalámbricas, lo que abre un mercado para el producto.

5.2.2 MERCADO A ATACAR

Mediante los resultados adquiridos por el análisis se obtuvo que el mercado será dirigido a universidades en sus facultades de electrónica y comunicaciones.

5.2.3 OBJETIVOS DE MERCADO

Se tiene los siguientes objetivos para el Entrenador Lógico:

1. Dar a conocer en el mercado nacional a nivel de universidades el Entrenador Lógico.

2. Abaratar costos para que el Entrenador Lógico sea accesible a las universidades y para estudiantes aficionados a la electrónica, especialmente en lo que se refiere a comunicaciones inalámbricas.

5.2.4 COSTOS DEL PRODUCTO

Para establecer el P.V.P. Se realiza un análisis de los costos de cada elemento electrónico, el costo de la mano de obra del Entrenador Lógico y del Minirobot.

La tabla 5.1 muestra la lista de los componentes electrónicos y el valor monetario de de los mismos, utilizados en el Entrenador Lógico.

LISTA DE COMPONENTES ELECTRÓNICOS			
CANT	**DESCRIPCIÓN**	**V.UNI**	**V.TOT**
29	RESISTENCIAS DE 330 OHM	0.02	0.58
21	RESISTENCIAS DE 4,7 K OHM	0.02	0.42
1	RESISTENCIA DE 10 OHM	0.03	0.03
9	CONDESADORES 1000uF25V	0.22	1.98
1	7805	0.4	0.4
1	7812	0.4	0.4
1	1117	0.85	0.85
1	FUSIBLE 2.5A	0.11	0.11
1	PORTA FUSIBLE	0.4	0.4
1	TRANSFORMADOR 9VAC	2	2
3	DISIPADOR TO220	0.45	1.35
3	BORNERAS 3P	0.4	1.2
4	BORNERAS 2P	0.22	0.88
11	DIODOS LED	0.07	0.77
1	BUZER 12 VDC	0.58	0.58
1	SWITC CUADRADO	0.4	0.4
1	PARLANTE PEQUEÑO DE 8 OHM	0.67	0.67
1	POTENCIOMETRO DIGITAL	1	1
1	MICROMOTOR 30:1	9	9
1	DIP-SWITCH 8 POSICIONES	0.49	0.49
1	MATRIZ DE LED 8x8	2.54	2.54
12	TRANSISTORES 2N3904	0.07	0.84
4	TRANSISTORES 2N3906	0.07	0.28
1	PANTALLA LCD 4x20	18.75	18.75
1	TECLADO MATRICIAL 4x4	1	1
4	DISPLAY DE ÁNODO COMÚN	0.89	3.56
1	POTENCIOMETRO DE 10KOHM	0.22	0.22
1	ZÓCALO UNIVERSAL	1	1
1	ZÓCALO 16 PINES	0.09	0.09
1	ZÓCALO 18 PINES	0.11	0.11
20	REGLETA HEMBRA	0.67	13.4
3	RELE 12 VDC	0.5	1.5
3	DIODOS RECTIFICADORES	0.07	0.21
5	PULSADORES PEQUEÑOS	0.11	0.55
1	MÓDULO BLUETOOTH WK	40.18	40.18
1	MÓDULO DE TX 434A	6.16	6.16
1	MÓDULO DE RX 434A	6.16	6.16
1	C.I. 7447	0.94	0.94
1	16F628A	2.9	2.9
1	BAQUELITA FIBRA DE VIDRIO	20	20
4	METRO DE ESTAÑO	0.22	0.88
2	FUNDA ACIDO FERRICO	0.5	1
	SUBTOTAL		145.78
	IVA 12%		17.49
	TOTAL		163.27

Tabla 5.1: Valor monetario de los componentes electrónicos utilizados en el Entrenador Lógico.

LISTA DE COMPONENTES ELECTRÓNICOS M-R			
CANT	**DESCRIPCIÓN**	**V.UNI**	**V.TOT**
1	7805	0.4	0.40
1	7812	0.4	0.40
1	1117	0.85	0.85
3	DISIPADOR TO220	0.45	1.35
1	ZÓCALO UNIVERSAL	1	1.00
6	REGLETA HEMBRA	0.67	4.02
8	DIODOS RECTIFICADORES	0.07	0.56
1	SENSOR ULTRASONIDO	20	20.00
1	MÓDULO BLUETOOTH WK	40.18	40.18
1	MÓDULO DE TX 434A	6.16	6.16
1	MÓDULO DE RX 434A	6.16	6.16
1	KIT LLANTAS	6.79	6.79
1	TWIN MOTOR GEAR	10.2	10.20
1	RUEDA LOCA	8.48	8.48
1	BATERIA 6V	4.6	4.60
1	PUENTE H	1.5	1.50
1	BAQUELITA FIBRA DE VIDRIO	10	10.00
4	METRO DE ESTAÑO	0.22	0.88
1	FUNDA ACIDO FERRICO	0.5	0.50
	SUBTOTAL		124.03
	IVA 12%		14.88
	TOTAL		138.91

Tabla 5.2: Valor monetario de los componentes electrónicos utilizados en el Mini-Robot.

DESCRIPCIÓN	DETALLE	VALOR
COMPONENTES ELECTRÓNICOS	ENTRENADOR LÓGICO	163.27
	MINI ROBOT	138.91
	TOTAL	**302.18**
ESTRUCTURA METÁLICA	ENTRENADOR LÓGICO	10
	MINI ROBOT	4
	TOTAL	**14**
COSTOS INDIRECTOS		20
	TOTAL	**20**
TOTAL		**336.18**

Tabla 5.3: Costo de diseño.

La tabla 5.3 muestra el valor monetario de los componentes físicos que se utilizaron para la elaboración del Entrenador Lógico

La tabla 5.4 muestra los los valores tomados en cuenta para el costo total o costo de Mano-Factura del Entrenador Lógico.

DETALLE	VALOR
COMPONENTES	302.18
ESTRUCTURA	14
COSTOS INDIRECTOS	20
MANO DE OBRA DIRECTA	40
COSTO TOTAL DEL PRODUCTO	**376.18**

Tabla 5.4: Costo de Mano-Factura del Entrenador Lógico.

5.3. ANÁLISIS COSTO BENEFICIO.

Mediante este análisis se determina la conveniencia de la implementación del proyecto mediante la enumeración y valoración posterior en términos monetarios de todos los costos y beneficios derivados directa e indirectamente del proyecto.

La tabla 5.5 muestra un beneficio importante y de mayor remuneración económica, la misma está proyectada dentro de un semestre de estudios en la Universidad Israel.

CANT	DESCRIPCIÓN	# ALUMNOS	VALOR	TOTAL
3	SEMINARIO POR SEMESTRE	4	60	$720.00
	TOTAL BENEFICIOS			$720.00

Tabla 5.5: Valor monetario del beneficios del proyecto.

El costo total de los beneficios en un semestre de estudios en la Universidad Israel determinada por la tabla 5.5 resultó $ 720,00 USD.

La Ecuación para determinar si el proyecto es o no rentable esta dada por :

B/C=VALOR MONETARIO DE LOS BENEFICIOS **/**COSTO TOTAL DEL PROYECTO

Aplicando la ecuación dada se obtiene el siguiente resultado:

B/C= 720,00 **/** 376,18

B/C= 1,914

La relación costo beneficio es mayor a uno por lo que el proyecto a realizar es rentable y además por cada entrenador invertido se obtiene una ganancia de 342,82 dolares USD.

Este análisis se lo realizó para el primer semestre para lo cual el proyecto genera beneficios económicos descritos en la tabla 5.5. Si la universidad adquiere un entrenador adicional se tendrá el doble de ganancia.

Ahora para el segundo semestre la ganancia es la totalidad del valor del beneficio económico descrito en la tabla 5.5 ya que no se adquiere otro Entrenador Lógico, entonces La Universidad Israel tendrá una ganancia de USD $ 720,00 por semestre a partir del segundo semestre.

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

- La guía de prácticas está realizada en base a J2ME que cuenta con componentes que la diferencian de otras versiones, como el uso de una máquina virtual denominada KVM, (Kilo Virtual Machine), debido a que requiere sólo pocos Kilo bytes de memoria para funcionar en lugar de JVM clásica.

- La programación de la interfaz gráfica para los dispositivos móviles java se realizó mediante el IDE NetBeans porque es un entorno de desarrollo integrado, es un producto libre y gratuito sin restricciones de uso, de código abierto escrito completamente en java usando la plataforma NetBeans y soporta el desarrollo de todos los tipos de aplicación Java requeridos para las prácticas en el Entrenador Lógico.

- Los teléfonos de última generación Android poseen memoria y procesador limitados, por lo que utilizan Dalvik que es una máquina virtual especializada diseñada específicamente para Android con la cual se puede ejecutar los programas de las prácticas para el Entrenador Lógico.

- El diseño del Entrenador Lógico se realizó en base a un proceso de análisis y síntesis que permitió la recopilación de los antecedentes bibliográficos necesarios para la implementación de un equipo con los componentes óptimos para la elaboración de proyectos basados en RF, BT, JAVA y ANDROID, tomando en cuenta parámetros eléctricos, electrónicos así como también parámetros de funcionalidad, costo y tamaño.

- Con el presente proyecto se beneficiará el estudiante, la Facultad de

Ingeniería Electrónica y la Uisrael ya que este Entrenador Lógico trabaja en base a programación J2ME y Android, sistemas operativos de ultima tecnología en comunicación en dispositivos móviles.

- Luego de realizar el Análisis Costo-Beneficio se determinó que la relación es mayor a uno por lo que el proyecto a implementar en la facultad de Ingeniería Electrónica de la UIsrael genera un superávit del 100% a partir del segundo semestre de implementado.

6.2. RECOMENDACIONES

- Realizar el mantenimiento del Entrenador Lógico cada 3 meses para prevenir daños irreversibles en el mismo ya que el equipo cuenta con módulos de comunicación inalámbrica sensibles.

- No utilizar el Entrenador Lógico en prácticas ajenas a las descritas por las Guías ya que el mismo podría generar errores en la transmisión y recepción de datos.

- Utilizar el Entrenador Lógico alejado de instrumentos de gran interferencia como cables de alta tensión, transmisores o receptores de radiodifusión, ya que éstos podrían ser causantes de datos erróneos en las prácticas realizadas.

- Se recomienda a los estudiantes, antes de utilizar el Entrenador Lógico, se familiaricen con el uso del sistema leyendo el manual y tomando en cuenta las respectivas normas de seguridad.

REFERENCIAS BIBLIOGRAFICAS

- http://www.cad.com.mx/historia_del_lenguaje_java.htm
- http://www.iec.csic.es/criptonomicon/java/quesjava.html
- http://www.lcc.uma.es/~galvez/ftp/libros/J2ME.pdf
- http://grasia.fdi.ucm.es/j2me/images/j2meLayers.jpg
- http://grasia.fdi.ucm.es/j2me/_J2METech/index.html
- http://es.wikipedia.org/wiki/NetBeans
- http://www.celularis.com/software/historia-android.php
- http://es.wikipedia.org/wiki/Android#Caracter.C3.ADsticas
- http://es.wikipedia.org/wiki/Archivo:System-architecture.jpg
- http://es.wikipedia.org/wiki/Radiofrecuencia
- http://es.wikipedia.org/wiki/Bluetooth
- http://es.wikipedia.org/wiki/bluethooth
- http://www.electronicafacil.net/tutoriales/Protocolos-Bluetooth.php
- http://www.univalle.edu.co/~telecomunicaciones/trabajos_de_grado/informes/tg_OscarRodriguez_RicardoMaya.pdf
- http://www.buenastareas.com/ensayos/Modulos-De-Transmision-y-Recepcion-Tipos-y/1714974.html
- http://www.bilbaoelectronics.com/radio-modulos-rf.html
- http://www.bilbaoelectronics.com/radio-transmisor-tlp434.jpg
- http://www.bilbaoelectronics.com/radio-modulos-rf.html
- http://www.bilbaoelectronics.com/radio-receptor-rlp434.jpg
- http://www.tecbolivia.com/index2.php
- http://www.tecbolivia.com/components/com_virtuemart/shop_image/product/M__dulo_Bluetoot_4dd290d121f28.jpg
- http://es.wikipedia.org/wiki/Microcontrolador_PIC#Arquitectura_central
- http://es.wikipedia.org/wiki/AVR
- Boylestad Nashelsky

INDICE DE FIGURAS

INDICE DE TABLAS

ANEXOS