UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

CONTROL DE UN SISTEMA DE ALARMA DE UN VEHÍCULO POR CELULAR

INFORME DE SUFICIENCIA

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PRESENTADO POR:
NEIL VICTORIANO MORENO SANDOVAL

PROMOCIÓN

1994-II

LIMA-PERÚ

2012

CONTROL DE UN SISTEMA DE ALARMA DE UN VEHÍCULO POR CELULAR

A mis padres, Esposa e Hija por
su apoyo permanente para poder
completar el presente informe

SUMARIO

El presente informe se basa en el desarrollo de un sistema de activación a distancia de una alarma de un vehículo, utilizando para ello la comunicación telefónica entre dos celulares. Para este propósito, se utilizará un celular emisor que sera llevado por el usuario (propietario del vehículo), desde el cual se podra gobernar ciertas acciones a distancia, las cuales serán ejecutadas en el vehículo. El otro celular es el receptor, el cual estará instalado en el vehículo junto con un circuito decodificador y de control, el celular receptor recibe los códigos enviados por el celular emisor y activa las acciones que se ejecutaran en el vehículo.

Para facilitar el manejo de este sistema por el usuario, en nuestro informe se desarrollará un programa en Java, el cual sera implantado en el celular emisor, de tal forma que el usuario cuente, en la pantalla del celular emisor, con una tabla de acciones que se pueden manejar con nuestro sistema.

Para realizar el envió de ordenes del celular emisor al receptor, utilizaremos el sistema de codificación por tonos DTMF (Dual Tone Multi Frequency), por lo cual en nuestro informe se hace un estudio sobre este sistema DTMF que se utiliza en los teléfonos para realizar una llamada telefonica.

Finalmente se desarrollará un programa para un Microcontrolador PIC16F84A, quien se encargará de recibir el código binario del decodificador para luego ejecutar las tareas que se le envian desde el celular emisor.

ÍNDICE

INTRODUCCIÓN

La elaboración del presente informe surge de la necesidad de poder encontrar una alternativa de solución eficaz frente a la problemática de robos de autos que vienen presentadose en nuestra ciudad. Tal es así, que el propietario de un vehículo en caso de haber sido victima del robo de este, tendrá la opción de poder recuperarlo utilizando nuestro sistema de alarma controlado a distancia.

El sistema de alarma controlado a distancia, que se plantea en este informe, pretende superar a cualquier otro sistema de alarma ya existente, ya que utilizamos equipos y software modernos, como el uso de Celulares, el uso de un Microcontrolador y lenguajes de programación avanzados como el Java. Todo esto lleva como resultado la fabricación de un Sistema de alarma mas sofisticado, presentando mayor seguridad para el usuario en el momento de activarlo y además de ser más económico que otros sistemas de alarmas similares existentes en el mercado.

Para poder cumplir con el proposito planteado, nuestro informe consta de cuatro capítulos, conclusiones y anexos.

En el primer capítulo, se describe y evalúa el problema, se establecen las limitaciones del trabajo y se detalla el objetivo del mismo, objetivo que guía su ejecución y que está presente desde la estructura hasta las conclusiones de este informe.

En el segundo capítulo marco teórico conceptual, se refiere a los conceptos teóricos con los que se basa el presente informe, desarrollando principalmente lo que es la programación de celulares con Java, programación de Microcontroladores y el sistema de codificación por tonos DTMF (Dual Tone Multi Frequency).

En el tercer capítulo, se presenta la propuesta de solución al problema planteado, así como la metodología a utilizar. El desarrollo de la propuesta, consiste en la realización de nuestro sistema de alarma, el cual podra ser activado a larga distancia desde un celular.

En el cuarto capítulo, se realiza la presentación de los resultados, en la que se muestra la recomendación expuesta en el presente informe para asegurar el correcto funcionamiento de nuestro sistema de alarma.

Finalmente, se presentan las conclusiones finales y recomendaciones del informe.

CAPÍTULO I
PLANTEAMIENTO DEL PROBLEMA

1.1 Descripción del problema

En nuestra ciudad la problemática que representa el robo de autos es alarmante, se calcula que aproximadamente unos 15 vehículos son robados a diario en la ciudad de Lima, trayendo en muchos casos consecuencias fatales para la sociedad, ya que a veces no solo se produce la pérdida del bien material, sino que también termina con la muerte del propietario del vehículo.

La marca de vehículos más buscada por los delincuentes es la Toyota, tal como se muestra en la Figura 1.1, debido a su gran demanda por su alta calidad y lo apreciado de sus repuestos. La mayoría de estos vehículos son canibalizados o desmantelados para ser vendidos como autopartes o repuestos de vehículos en el mercado negro, siendo más fácil y rentable para los delincuentes que la venta del vehículo robado.

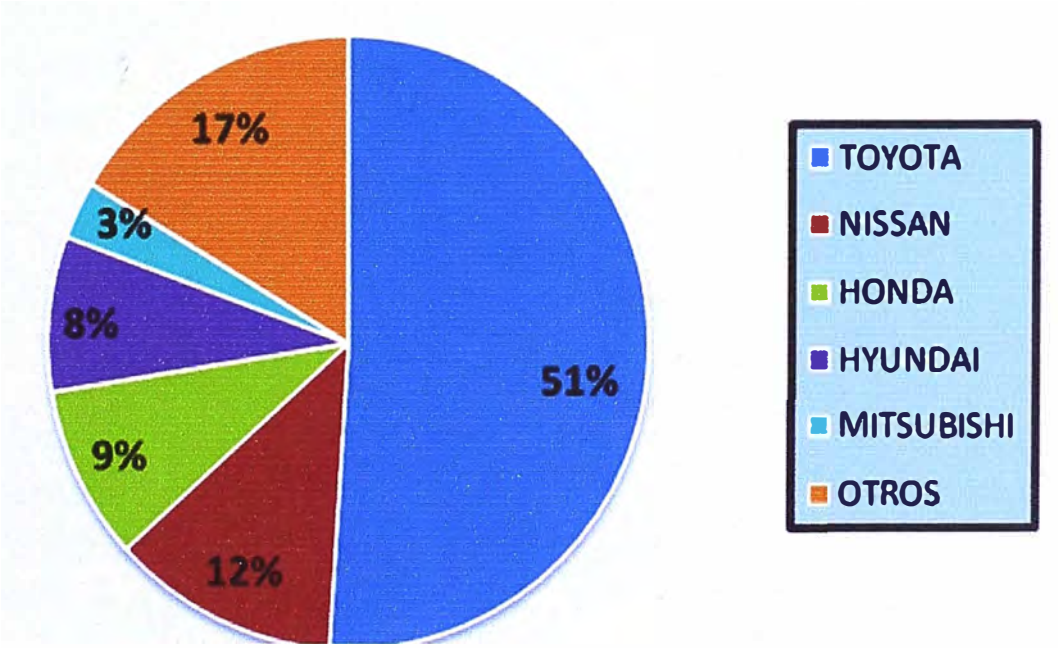


Figura 1.1 Marca de vehículos más robados en Lima.

Sin embargo, un porcentaje de vehículos robados son vendidos y llevados a provincias, donde existe menos control de la flota vehicular y pueden circular libremente. De igual modo, una parte de vehículos robados salen al exterior, a países como Bolivia y Ecuador, donde son comercializados y por último, también algunos de estos vehículos

sustraídos son utilizados para cometer asaltos, secuestros y robos al paso, en distintos lugares de la capital.

Entre las principales modalidades que usan los delincuentes para robar un vehículo, se tienen la del encañonamiento, la interceptación y el vehículo estacionado, tal como se muestra en la figura 1.2.

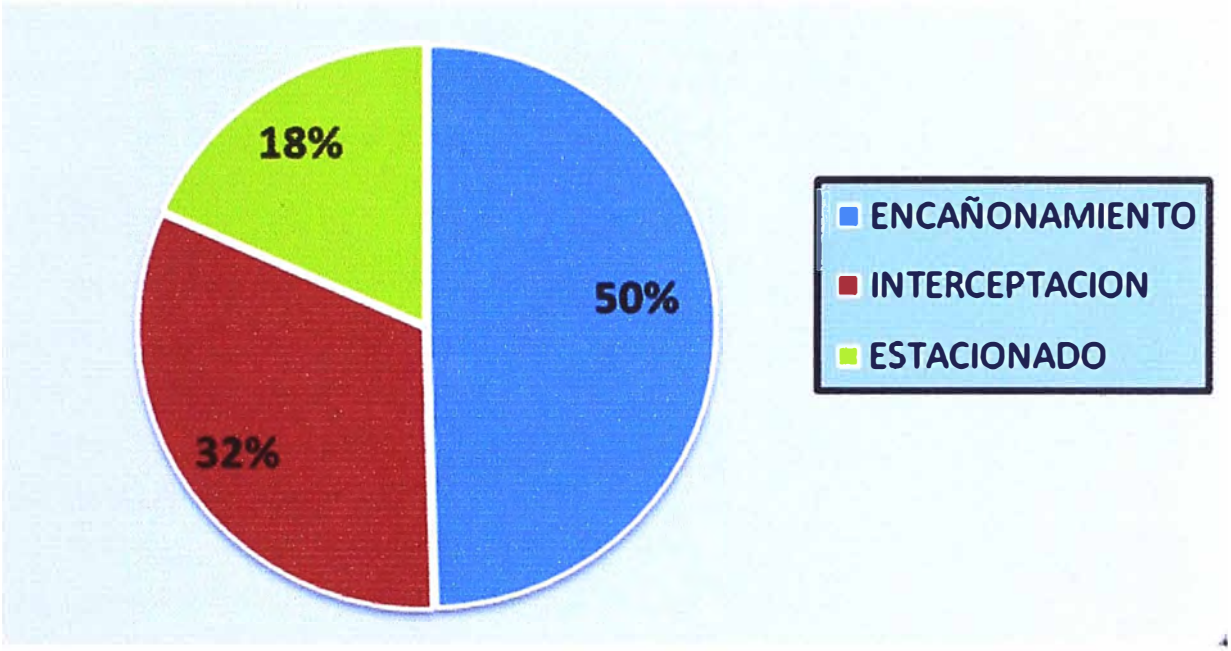


Figura 1.2 Modalidades que usan los delincuentes para robar vehículos.

El Encañonamiento; que representa el 50% de los robos de vehículo cometidos en el año 2011. Esta modalidad consiste en el seguimiento de las víctimas hasta que estas llegan a su destino y detienen el vehículo para estacionarse, este instante es aprovechado por los delincuentes que descienden de su vehículo con armas de fuego reduciendo y hasta golpeando a su víctima para llevarse su vehículo si ésta ofrece resistencia.

La Interceptación; representa el 32% del total de robos cometidos durante el año pasado. Esta modalidad consiste en hacer el seguimiento a una víctima mientras conduce su vehículo por las calles del distrito y esperar el momento apropiado (generalmente en una calle desolada o con poca iluminación) para cerrar el vehículo del agraviado obligando a detenerlo para encañonarlo y despojarlo de su vehículo.

Estacionado; esta modalidad de hurto de vehículos representa el 18%, del total de robos de vehículos en la ciudad, y consiste en el hurto de un vehículo estando estacionado sin custodia. Esta ocasión es aprovechada por los delincuentes para fracturar la cerradura de las puertas del auto utilizando una herramienta llamada "peine" o "llave maestra" para poder violentar la puerta y proceder al hurto del vehículo.

Frente a esta problemática los propietarios de los vehículos se ven obligados a instalar sistemas de alarma en sus automoviles, pero siendo en su mayoría sistemas

ineficientes, ya que los delincuentes se las ingenian para desactivarlas y lograr que estas no lleguen a activarse. Existen también en nuestro medio otro tipo de alarma más eficiente y seguro, que utilizan un localizador GPS, pero amedita que el usuario pague una mensualidad elevada, por lo tanto este tipo de alarma no es muy accesible para la mayoría de automovilistas, es por este motivo la razon de este proyecto, donde el dueño del vehículo tendrá la opción de poder comandar el apagado de su auto a distancia, aun cuando el vehículo ya halla sido robado, también se podrá activar una sirena para que los ladrones dejen el auto y el dueño así pueda recuperarlo, y además el costo de esta alarma será económico.

1.2 Objetivos

Los objetivos del presente trabajo estan divididos en dos objetivos generales y tres objetivos específicos, los cuales señalamos a continuación.

1.2.1 Objetivos Generales:

Disminuir el número de robos de vehículos en la ciudad, diseñando para esto un sistema de alarma para vehículos más eficiente.

Contar con un sistema de alarma que sea fácil de operar, que preste mayor seguridad al propietario al momento de activarla y que sea económica.

1.2.2 Objetivos Especificos:

Elaborar una aplicación en Java para celulares.

Decodificar los tonos DTMF que se producen en una comunicación telefonica.

Programar un microcontrolador para que controle los accionamientos del sistema.

Poder controlar a distancia desde un celular el bloqueo y desbloqueo del funcionamiento de un motor, la activación y desactivación de una sirena de un automóvil.

Figura 1.3 Control de Alarma a distancia por Celular

1.3 Limitaciones

El funcionamiento del sistema de alarma desarrollado en este informe, esta supeditado a la cobertura de telefonía celular que se tenga, tal es asi que si el vehículo se encontrase en alguna zona donde no se cuente con señal de comunicación por celular, la alarma no podria activarse, por lo tanto no funcionaría nuestro sistema de alarma en general.

El proyecto que se detalla en este informe solo hace mención al diseño del circuito de control del sistema de alarma, pero no se detalla la circuiteria del sistema de encendido que tiene el automóvil, ni tampoco el circuito de la sirena, que ya debe de estar instalado en el automóvil para poder ser conectada a nuestro circuito de control.

En el presente informe se desarrolla el sistema de alarma para el vehículo, pero no se detalla la instalación del sistema en el circuito electrico del automóvil, ya que este informe no comprende el estudio de la circuiteria de encendido que tienen los vehículos y además se considera que esta instalación no representa mayor dificultad para su ejecución, pudiendola realizar cualquier persona capacitada en la especialidad de electricidad automotriz.

CAPÍTULO II
MARCO TEÓRICO CONCEPTUAL

2.1 Teléfonos celulares

La telefonía móvil usa ondas de radio para poder ejecutar las operaciones desde el móvil a la base, ya sea llamar, mandar un mensaje de texto, etc., y esto es producto de lo que sucedió hace algunas décadas.

La comunicación inalámbrica tiene sus raíces en la invención de la radio por Nikola Tesla en los años 1880, aunque formalmente presentado en 1894 por un joven italiano llamado Guglielmo Marconi.

El teléfono móvil se remonta a los inicios de la Segunda Guerra Mundial, donde ya se veía que era necesaria la comunicación a distancia, es por eso que la compañía Motorola creó un equipo llamado Handie Talkie H12-16, que es un equipo que permite el contacto con las tropas vía ondas de radio cuya banda de frecuencias en ese tiempo no superaban los 60MHz.

Este fue el inicio de una de las tecnologías que más avances tiene, aunque continúa en la búsqueda de novedades y mejoras.

Durante ese periodo y 1985 se comenzaron a perfeccionar y amoldar las características de este nuevo sistema revolucionario ya que permitía comunicarse a distancia. Fue así que en los años 1980 se llegó a crear un equipo que ocupaba recursos similares a los Handie Talkie pero que iba destinado a personas que por lo general eran grandes empresarios y debían estar comunicados, es ahí donde se crea el teléfono móvil y marca un hito en la historia de los componentes inalámbricos ya que con este equipo podría hablar a cualquier hora y en cualquier lugar.

La evolución de la telefonía móvil se ha ido agrupando en generaciones, tal es así que se inicia con la primera generación (1G) y llega hasta la cuarta generación (4G).

2.1.1 Primera Generación (1G)

La primera generación de teléfonos celulares comenzó en 1979 y se trataba de conexiones estrictamente de voz y analógicas. Estas conexiones no tenían seguridad y generaban muchos conflictos en las comunicaciones. La tecnología que ha permitido esta comunicación se llamó AMPS (Advanced Mobile Phone System) y todavía sigue siendo utilizada en lugares rurales y ciudades alejadas de América.

Los equipos 1G pueden parecer algo aparatosos para los estándares actuales pero fueron un gran avance para su época, ya que podían ser trasladados y utilizados por una única persona.

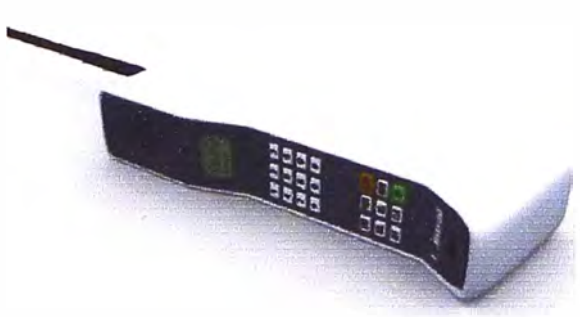


Figura 2.1 Teléfono Celular 1G

2.1.2 Segunda Generación (2G)

Hacia 1990 la tecnología evolucionó a lo que se denominó 2G (Segunda Generación), esta etapa se caracterizó por ser digital y utilizar algoritmos de compresión y seguridad más utilizada actualmente en las comunicaciones móviles del mundo. En esta etapa, se encuentran predominantemente 3 tipos de tecnologías compitiendo en el mercado: Acceso múltiple por división de código (CDMA), Acceso múltiple por división de tiempo (TDMA) y Global System for Mobile communications. GSM es la tecnología que más ha evolucionado en esta generación y por ello, actualmente. Posee más del 70% del mercado mundial. Técnicamente es un derivado de la tecnología TDMA.

El sistema 2G trajo consigo nuevas aplicaciones de datos sobre la red celular: fax, modem, SMS; aunque rápidamente su capacidad de ancho de banda quedó limitada. Por esta limitación de la segunda generación (9,6 Kbps) y, al darse cuenta que la próxima generación (la 3G) tardaría unos cuantos años más en venir, los fabricantes crearon un intermedio llamado 2,5G que permite velocidades de 64 Kbps o superiores.

Figura 2.2 Teléfono Celular 2G

2.1.3 Generación de Transición (2.5G)

Dado que la tecnología de 2G fue incrementada a 2.5G, en la cual se incluyen nuevos servicios como EMS y MMS:

EMS es el servicio de mensajería mejorado, permite la inclusión de melodías e iconos dentro del mensaje basándose en los sms; un EMS equivale a 3 o 4 sms.

MMS (Sistema de Mensajería Multimedia) Este tipo de mensajes se envían mediante GPRS y permite la inserción de imágenes, sonidos, videos y texto. Un MMS se envía en forma de diapositiva, en la cual cada plantilla solo puede contener un archivo de cada tipo aceptado, es decir, solo puede contener una imagen, un sonido y un texto en cada plantilla, si se desea agregar más de estos tendría que agregarse otra plantilla. Cabe mencionar que no es posible enviar un vídeo de más de 15 segundos de duración.

Para poder prestar estos nuevos servicios se hizo necesaria una mayor velocidad de transferencia de datos, que se hizo realidad con las tecnologías GPRS y EDGE.

GPRS (General Packet Radio Service) permite velocidades de datos desde 56 kbps hasta 114 kbps.

EDGE (Enhaced Data rates for GSM Evolution) permite velocidades de datos que llegan hasta los 384 Kbps

Figura 2.3 Teléfono Celular 2.5G

2.1.4 Tercera Generación (3G)

La tercera generación nace de la necesidad de aumentar la capacidad de transmisión de datos para poder ofrecer servicios como la conexión a Internet desde el móvil, la videoconferencia, la televisión y la descarga de archivos. En este momento el desarrollo tecnológico ya posibilita un sistema totalmente nuevo: UMTS (Universal Mobile Telecomunicaciones System). UMTS utiliza la tecnología CDMA, lo cual le hace alcanzar velocidades realmente elevadas, de 144 Kbps hasta 7.2 Mbps, según las condiciones del terreno.

Figura 2.4 Teléfono Celular 3G

2.1.5 Cuarta Generación (4G)

En la actualidad en algunos países del mundo ya está instalada la cuarta generación o 4G, la cual ofrece al usuario de telefonía móvil un mayor ancho de banda y velocidades de acceso mayores a 100 Mbps que permite, entre muchas otras cosas, la recepción de televisión en Alta Definición y la conexión a internet a muy altas velocidades.

Figura 2.5 Teléfono Celular 4G

Debido a este gran avance tecnológico que se tiene en los equipos y servicios celulares, es que la telefonía celular se ha convertido en algo muy indispensable para la humanidad, tal es así que actualmente se estima que el número de usuarios de celulares en todo el mundo alcanzó los 6,000 millones, según el Informe de Datos del Tráfico y Desarrollo del Mercado Móvil de la compañía Ericsson y se espera que en un futuro muy próximo, el número de celulares en el mundo será mayor al número de habitantes.

2.2 Programación con Java

En la década de los 90, la empresa Sun Microsystems, desarrollo un nuevo lenguaje con una característica que lo hacía único era Multiplataforma. Así nació Java, un lenguaje de programación que permite ejecutar las aplicaciones que desarrollamos en varias plataformas, sistemas operativos y equipos sin necesidad de recompilar o realizar cambios en el código fuente. Por ello, las aplicaciones Java pueden ejecutarse actualmente en varias plataformas distintas, entre DOS, Windows, Macintosh, Linux,

Solaris y, como veremos más adelantes, también en equipos Plam, Pocket PC y celulares de diferentes marcas y sistemas operativos.

Figura 2.6 Logo de Java

2.2.1 Características de Java

Dentro de las principales caracteristicas de java tenemos:

- Java es un lenguaje orientado a objetos: Esto es lo que facilita abordar la resolución de cualquier tipo de problema.

Es un lenguaje sencillo, aunque sin duda potente.

La ejecución del código Java es segura y fiable: Los programas no acceden directamente a la memoria del ordenador, siendo imposible que un programa escrito en Java pueda acceder a los recursos del ordenador sin que esta operación le sea permitida de forma explícita. De este modo, los datos del usuario quedan a salvo de la existencia de virus escritos en Java. La ejecución segura y controlada del código Java es una característica única, que no puede encontrarse en ninguna otra tecnología.

- Es totalmente multiplataforma: Es un lenguaje sencillo, por lo que el entorno necesario para su ejecución es de pequeño tamaño y puede adaptarse incluso al interior de un navegador.

2.2.2 Programación Orientada a Objetos

JAVA es un lenguaje totalmente orientado a objetos. Esto significa que toda la funcionalidad que queramos desarrollar deberá estar enmarcada en clases y objetos.

En la POO (Programación Orientada a Objetos), no trabajamos con funciones, procedimientos y variables por como las entendemos en programación imperativa. Aquí se definen clases de objetos. Una clase es un conjunto de atributos, variables u otros objetos, sumado a métodos que se pueden ejecutar sobre él. Por ejemplo, en un sistema de administración de escuelas, una clase puede ser Alumno, que contenga todos los datos del legajo y materias cursadas, junto con operaciones (métodos) tales como "Dar un examen", "Pagar una Cuota", "Actualizar Legajo" , así como puede existir otra clase Profesor, que contenga los cursos y horarios que esté dictando, entre otra información y métodos, tales como "Cobrar Sueldo", "Dictar clase", "Avisar Ausencia por Enfermedad".

Cuando programamos, creamos objetos que son de alguna clase ya declarada de antemano, por ello podremos acceder a todos sus métodos para comunicarnos con cada

objeto. Es importante no pensar en los objetos como simple variable, sino como estructura con las cuales podremos interactuar teniendo encapsulada su forma de ejecutar las acciones.

La POO tiene más ventaja además del encapsulamiento tales como mejor reutilización del código, mayor claridad en entender conceptualmente un programa y algunas otras características más complejas como ser la herencia y el polimorfismo.

Diremos rápidamente que la herencia permite que una clase "herede" de otra clase su comportamiento y atributos y pueda extenderlos o agregarles métodos nuevos.

Por ejemplo, podemos tener una clase Persona que posea nombre, apellido y teléfono y extender dicha clase a otras dos, Alumnos y Profesor, quienes heredaran las características de persona y agregaran nuevas, como ser materias aprobadas o materia dictadas.

En la POO tendremos que pensar nuestra aplicaciones como ciertos objetos de alguna clase interactuando y enviándose mensajes entre si para lograr el objetivo.

2.2.3 Máquina Virtual de Java

Las aplicaciones JAVA poseen una particularidad respecto a otros lenguajes: nuestra aplicación no interactúan directamente con la computadora donde se ejecute ni con el sistema operativo; interactuará con un sistema llamado Maquina Virtual o JVM (Java virtual Machine). Dado que JAVA es multiplataforma, es necesario que exista una aplicación que tenga la tarea de traducir nuestro código a cada plataforma compatible con JAVA; esa es la tarea de la JVM.

Cuando compilamos nuestro programa en Java, no obtenemos un programa ejecutable como con otros lenguajes, sino que obtendremos un código precompilado que podrá ser ejecutado en cualquier Maquina Virtual Java. De esta forma, es necesario que esta JVM esté instalada en el sistema operativo donde queremos ejecutar nuestra aplicación. Esta máquina virtual viene instalada ya por defecto en varios sistemas operativos, con ella podremos ejecutar cualquier aplicación Java, aunque haya sido pensada y desarrollada en otro sistema.

En el caso de la programación para celulares, debido a las capacidades limitadas de almacenamiento, memoria, procesamiento y pantalla que tienen estos equipos, no se puede integrar una máquina virtual Java completa, por lo cual se eliminan algunas características de la JVM y se llega a crear una nueva máquina virtual Java compacta y portable, la cual se ejecuta en unos cuantos Kilobytes de memoria, de alli su nombre de Kilobyte Virtual Machine (KVM).

2.2.4 Versiones de Java

Sun define tres plataformas bien diferenciadas, de acuerdo con el tipo de aplicación y

destino que se requiera, las cuales son:

- J2ME (Java 2 Micro Edition), orientada a entornos de limitados recursos, como teléfonos móviles, televisores, automóviles, etc.
- J2SE (Java 2 Standard Edition), para entornos de gama media y estaciones de trabajo. Aquí se sitúa al usuario medio en un PC de escritorio.
- J2EE (Java 2 Enterprise Edition), orientada a entornos distribuidos empresariales o de Internet.

2.2.5 El J2ME

Sun desarrollo J2ME para trabajar sobre equipos con ciertas limitaciones: en procesamiento, pantalla, energía basada en batería y de alguna forma poseen servicios de red para comunicarnos con el exterior.

La versión de Java J2ME, que si bien es lenguaje Java nativo, posee ciertas limitaciones en cuanto a las clases disponibles, tipos de datos y funciones que se pueden utilizar. Estas limitaciones se deben solamente a que los equipos celulares poseen poco poder de procesamiento y de memoria. Mas allá de esto, las características de J2ME son las mismas que JAVA.

J2ME fue pensado para trabajar sobre equipos con ciertas limitaciones: en procesamiento, pantalla, energía basada en batería y de alguna forma poseen servicios de red para comunicarnos con el exterior.

Una aplicación desarrollada en J2ME podrá ser ejecutada en cualquier equipo celular o Palm que tenga una Maquina Virtual Java instalada y, en esta categoría, entran la mayoría de los dispositivos móviles que se ofrecen en el mercado actualmente. De esta forma, nuestras aplicaciones o juegos podrán ejecutarse de igual forma en cualquier equipo sin realizar cambios en el código ni tener que recompilar.

Java para poder ejecutar una aplicación en cualquier teléfono celular teniendo en cuenta las diferencias técnicas entre cada uno de ellos, le entrega la responsabilidad a cada fabricante. Los teléfonos o equipos que soporten J2ME deberán cumplir con el estándar desarrollado por SUN y ejecutar las funciones como mejor puedan. De esta forma cada fabricante adapta las funcionalidades que se ofrecen en J2ME de la mejor forma posible en cada uno de sus modelos.

2.2.6 Configuraciones del J2ME

J2ME no solo fue pensado para equipos celulares, sino para otros dispositivos que poseen características similares, como televisores, sistemas de navegación en automóviles, palms y otros equipos. Estos equipos poseen algunas ventajas respecto a los teléfonos celulares, sin embargo aun asi no cumplen con los requisitos para poder ejecutar la versión madre de Java, la J2SE.

Por ello, existen dos configuraciones para poder trabajar en J2ME:

- La configuración CLDC (Connected Limited Device Configuration), es una configuración pensada para equipos con bajas capacidades técnicas, como los teléfonos celulares. Poseen una Máquina Virtual muy optimizada y algunas de las funciones y capacidades de Java no están presentes. La librería de funciones disponibles es limitada, teniendo en cuenta que la KVM ocupa entre 40 y 80 KB de memoria.
- La configuración CDC (Connected Device Configuration), esta configuración esta hecha para equipos de mayor capacidad, de 32 bits y más de 2 MB de memoria. Al tener mayor capacidad se posee acceso completo a las funciones de Java Standard Edition, mejores funciones de seguridad y de su máquina virtual CVM (Compact Virtual Machine).

2.2.7 Perfiles en J2ME

El perfil es un conjunto de APIs (Application Program Interface) que dotan a una configuración de funcionalidad específica, orientado a un ámbito de aplicación determinado. Los perfiles identifican un grupo de dispositivos por la funcionalidad que proporcionan y el tipo de aplicaciones que se ejecutarán en ellos.

En J2ME existen unos perfiles que se usarán sobre la configuración CDC y otros sobre la CLDC.

Para la configuración CDC existen los siguientes perfiles:

- Foundation Profile.
- Personal Profile.
- RMI Profile.

Y para la configuración CLDC:

- PDA Profile.
- Mobile Information Device Profile (MIDP).

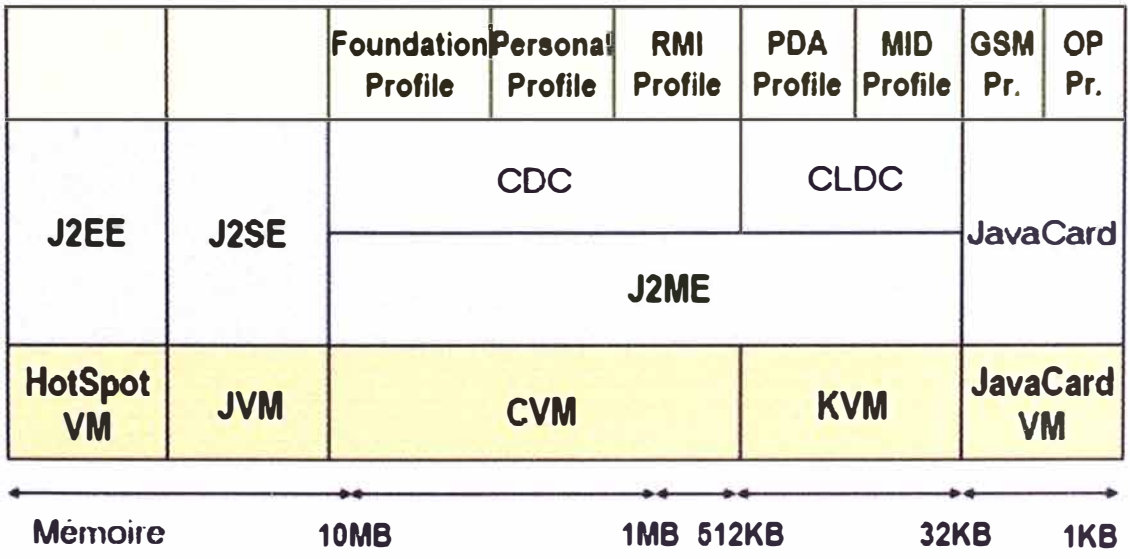


Figura 2.7 Configuraciones del J2ME

2.2.8 El MIDP

El MIDP fue el primer perfil definido para la plataforma CLDC, en la actualidad existen dos versiones disponibles para MIDP la 1.0 y la 2.0. Los primeros equipos con soporte de Java, solo tenian disponibles la versión 1.0 y los que actualmente se ofrecen en el mercado generalmente tienen disponibles ambas versiones.

Las clases en las APIs de Java se organizan en grupos disjuntos llamados paquetes, cada paquete contiene un conjunto de interfaces, clases y excepciones relacionadas, tal es así que los paquetes que están incluidos en el MIDP son los siguientes:

- Javax.microedition.midlet

Esta clase nos permite crear una clase heredada que haga las funciones de Midlet. Sera quien se ejecute en primer instancia al invocar a la aplicación desde el celular. Al crear una clase de este tipo nos obligara a definir tres métodos: startApp(), destroyApp() y pauseApp() que serán invocados al entrar en cada uno de los estados.

- Javax.microedition.lcdui

Este paquete contiene todas las clases necesarias para realizar nuestras interfaz de entrega y salida con el usuario. Tiene una clase Display que será la encargada del manejo de la pantalla, en bajo o alto nivel, que veremos en próximo capítulo. También tiene clases para manejar comandos y opciones y para leer teclas presionadas.

- Javax.microedition.rms

Es el paquete que contendrá las clases necesarias para poder escribir y leer registros almacenados en la memoria del equipo celular para un futuro uso de la aplicación.

- Javax.microedition.io

Contiene la clase connection que será la encargada de administrar las conexiones entrante y saliente hacia internet u otro medio de comunicación disponibles.

Además el CLDC cuenta con clases heredadas (derivadas) de J2SE, estas clases heredadas se encuntran en los siguientes paquetes:

- Java.io

Contiene clases heredadas de entrada/salida.

- Java.lang

Contiene algunas clases heredadas de sistema.

- Java.util

Contiene clases heredadas que tienen que ver con algunas utilidades de la plataforma J2SE.

Las aplicaciones creadas para celulares utilizando el perfil MIDP reciben el nombre de MIDlet.

2.2.9 El MIDlet

Un MIDlet es una aplicación Java realizada con el perfil MIDP sobre la configuración CLDC, tal es así que toda aplicación para celular siempre tendrá un objeto de tipo Midlet que será el encargado de proveernos la funcionalidad necesaria en un equipo móvil.

Todo MIDlet durante su ejecución pasa por 3 estados diferentes: Activo, Pausado y Destruido.

- Activo: la aplicación se encuentra ejecutándose normalmente.
- Pausado: la aplicación ha sido detenida temporalmente por algún evento, como puede ser la recepción de una llamada o todavía no ha pasado a estado Activo al ser creada.
- Destruido: la aplicación se ha cerrado y se ha liberado la memoria ocupada.

Un MIDlet puede cambiar de un estado a otro mediante una llamada a los métodos startApp (), pauseApp () y destroyApp (). El gestor de aplicaciones cambia el estado de los MIDlets haciendo una llamada a cualquiera de los métodos anteriores o también el MIDlet puede cambiar de estado por sí mismo.

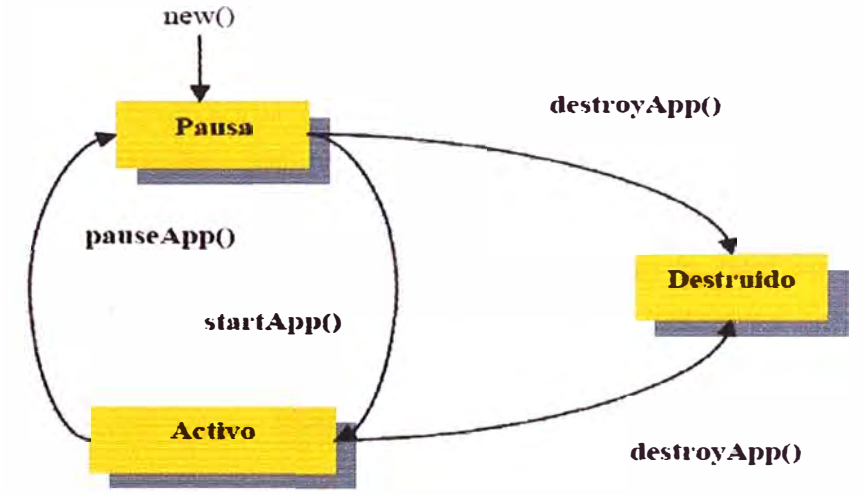


Figura 2.8 Estados de un MIDlet

2.2.10 El Netbeans

Para lograr un mejor rendimiento como programadores se han desarrollado entornos integrados de desarrollos (IDE), que en una sola interfaz nos permite editar el código fuente, administrar los proyectos, compilar, ejecutar, realizar debug y otras herramienta sin cambiar de entorno de trabajo.

En este proyecto se va a utilizar el Netbeans IDE, ya que es de facil uso y además contiene todos los elementos necesarios directamente integrados en un mismo entorno sin necesidad de estar añadiendo otras herramientas.

Netbeans es un IDE creado por Sun para programar en Java, da la posibilidad de crear aplicaciones para todas sus plataformas entre ellas J2ME, y se puede descargar desde la misma pagina de Sun en forma gratuita.

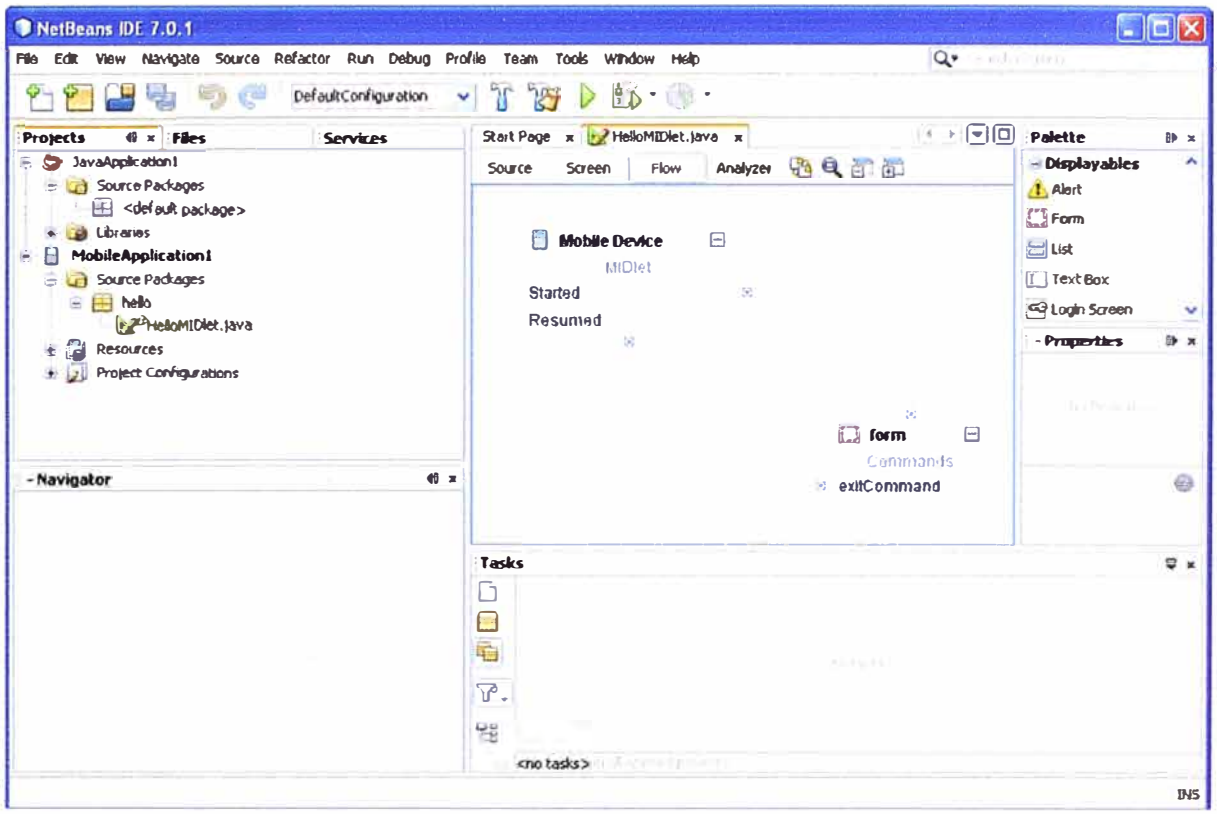


Figura 2.9 El Netbeans

Netbeans también nos permite realizar nuestro progama en forma gráfica, mediante el agregado de elementos directamente sobre la pantalla del terminal y el diseño se realiza mediante diagramas de flujo, con lo cual se logra realizar una programación sencilla y rapida. Además el Netbeans incluye un emulador integrado el cual utilizaremos en nuestro proyecto.

Figura 2.10 Emulador del Netbeans

2.3 El Microcontrolador

Un microcontrolador es un circuito integrado que contiene toda la estructura de una microcomputadora, o sea, un CPU (Unidad de Central de Proceso), Memoria RAM, Memoria ROM, circuitos de entrada/salida (I/O), y otros módulos con aplicaciones especiales, formando asi un sistema completo para cumplir con una aplicación especifica dentro del mundo real.

Para que el sistema pueda realizar su labor deberá ejecutar paso a paso un programa que consiste en una secuencia de números binarios o instrucciones, almacenadas, en su memoria interna de programa.

A mediados de los años 70, el panorama de la electrónica y especialmente el de la electrónica digital cambió radicalmente con la aparición en el mercado del microcontrolador. Esto introdujo un concepto novedoso que en la actualidad se conserva y refuerza cada vez más, el de la lógica programada, que consiste en que en el microcontrolador es posible modificarle su programa cuantas veces se requiera, aún en forma remota y el circuito o aparato en el cual esta instalado trabajará de una forma diferente sin midificar una sola conexión. De esta forma, el limite de los diseños solo lo establece la imaginación de los programadores y la capacidad de los dispositivos que lancen al mercado los fabricantes especializados en estas tecnologías.

En los últimos años han tenido una gran acogida los microcontroladores PIC (Peripheral Interface Controller), fabricados por Microchip Technology Inc., gracias a sus buenas caracteristicas, bajo precio, reducido consumo, pequeño tamaño, gran calidad, fiabilidad y abundancia de información, los convierten en muy fácil, cómodo y rápido de utilizar. Es por esto que en nuestro proyecto utilizamos el microcontrolador PIC16F84A, el cual se encuentra encapsulado en un chip de 18 pines, tal como se muestra en la fig.2.11

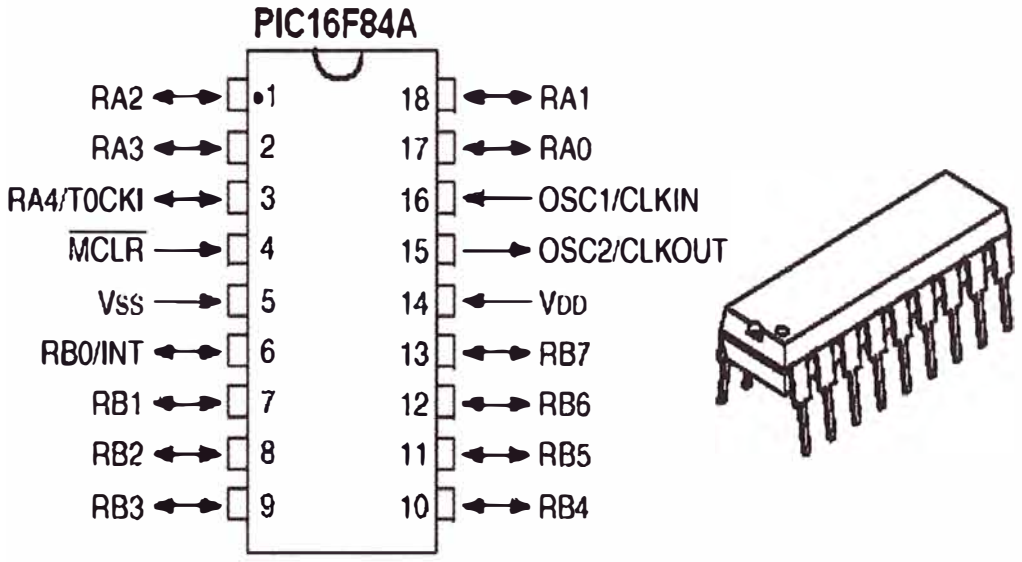


Figura 2.11 El PIC16F84A

2.3.1 Alimentación

El microcontrolador PIC16F84A se alimenta con 5 voltios, los cuales son aplicados entre los pines VDD y VSS que son, respectivamente, la alimentación y la masa del chip.

El consumo de corriente para el funcionamiento del microcontrolador depende de la frecuencia de trabajo y de las cargas que soporten sus salidas, siendo del orden de unos pocos miliamperios.

2.3.2 Puertos de Entrada/Salida

El microcontrolador se comunica con el mundo exterior a través de los puertos, los cuales están constituidos por líneas digitales de entradas/salida que trabajan entre 0 y 5 V. Los puertos se pueden configurar como entradas para recibir datos o como salidas para gobernar dispositivos externos, el PIC16F84A tiene dos puertos:

- El Puerto A con 5 líneas, pines RA0 a RA4.
- El Puerto B con 8 líneas, pines RB0 a RB7.

Cada línea puede ser configurada como entrada o como salida, independientemente unas de otras, según se programe.

Las líneas son capaces de entregar niveles TTL cuando la tensión de alimentación aplicada en VDD es de 5V. La máxima capacidad de corriente de cada una de ellas es de aproximadamente 20 mA.

2.3.3 Oscilador

Todo microcontrolador requiere de un circuito que le indique la velocidad de trabajo, es el llamado oscilador o reloj. Este genera una onda cuadrada de alta frecuencia que se utiliza como señal para sincronizar todas las operaciones del sistema. Este circuito es muy simple pero de vital importancia para el buen funcionamiento del sistema. Generalmente todos los componentes del reloj se encuentran integrados en el propio microcontrolador y tan solo se requieren unos pocos componentes externos, como un cristal de cuarzo o una red RC, para definir la frecuencia de trabajo.

En el PIC16F84 los pines OSC1/CLKIN y OSC2/CLKOUT son las líneas utilizadas para este fin. Permite cinco tipos de osciladores para definir la frecuencia de funcionamiento:

- XT. Cristal de cuarzo.
- RC. Oscilador con resistencia y condensador.
- HS. Cristal de alta velocidad.
- LP. Cristal para baja frecuencia y bajo consumo de potencia.
- Externa. Cuando se aplica una señal de reloj externa.

2.3.4 Reset

El llamado reset en un microcontrolador provoca la reinicialización de su

funcionamiento, un "comienzo a funcionar desde cero". En este estado, la mayoría de los dispositivos internos del microcontrolador toman un estado predeterminado.

En los microcontroladores se requiere un pin de reset para reiniciar el funcionamiento del sistema cuando sea necesario. El pin de reset en los PIC se denomina MCLR *(Master Clear)* y produce un reset cuando se le aplica un nivel lógico bajo.

El PIC16F84 también permite el llamado power-On reset *(POR)*, que proporciona un reset al microcontrolador en el momento de conectar la fuente de alimentación. El PIC dispone de un temporizador denominado reset PWRT *(Power-up Timer)*, que proporciona una retardo de 72 ms desde el momento de la conexión a la alimentación; un reset se mantiene durante este tiempo, garantizado que VCC alcance un nivel aceptable de tensión para un arranque correcto del sistema. Para utilizar este tipo de reset, hay que conectar el pin MCLR al positivo de la alimentación. Además, hay que programarlo así durante el proceso de grabación. Con esto se evita utilizar las tradicionales redes RC externas de otros microcontroladores.

2.3.5 Arquitectura Interna

En la figura 2.12 se representa el diagrama de bloques de la arquitectura interna del PIC16F84A, donde se destacan los siguientes componentes:

- Memoria de programa tipo ROM Flash de 1 k x 14 bits.
- Memoria de datos divididas en 2 áreas:
- Área RAM constituida por 22 registros de propósito específicos (SFR) y 68 de propósito general.
- Área EEPROM de datos formada por 64 registros de 8 bits
- ALU de 8 bits y registro de trabajo W, del que normalmente recibe un operado que puede ser cualquier registro, memoria, puerto de entrada/salida o el propio código de instrucción.
- Dos puertos para la comunicación con el mundo exterior: PORTA y PORTB de 8 bits cada uno.
- Contador de programa de 13 bits, lo que permite direccionar 1k de la memoria de programa del PIC.

2.3.6 Organización de la Memoria

Dentro del PIC16F84A se distinguen tres bloques de memoria:

- Memoria de programa. En sus 1024 posiciones contiene el programa con las instrucciones que gobiernan la aplicación. Es del tipo no volátil, es decir, el programa se mantiene aunque desaparezca la alimentación.
- Memoria de datos RAM. Se destina a guardar las variables y datos. Es volátil, es decir, los datos almacenados se borran cuando desaparece la alimentación.

- Memoria EEPROM de datos. Es una pequeña área de memoria de datos de lectura y escritura no volátil, gracias a la cual, un corte del suministro de la alimentación no ocasiona la perdida de la información, que estará disponible al reinicializarse el programa.

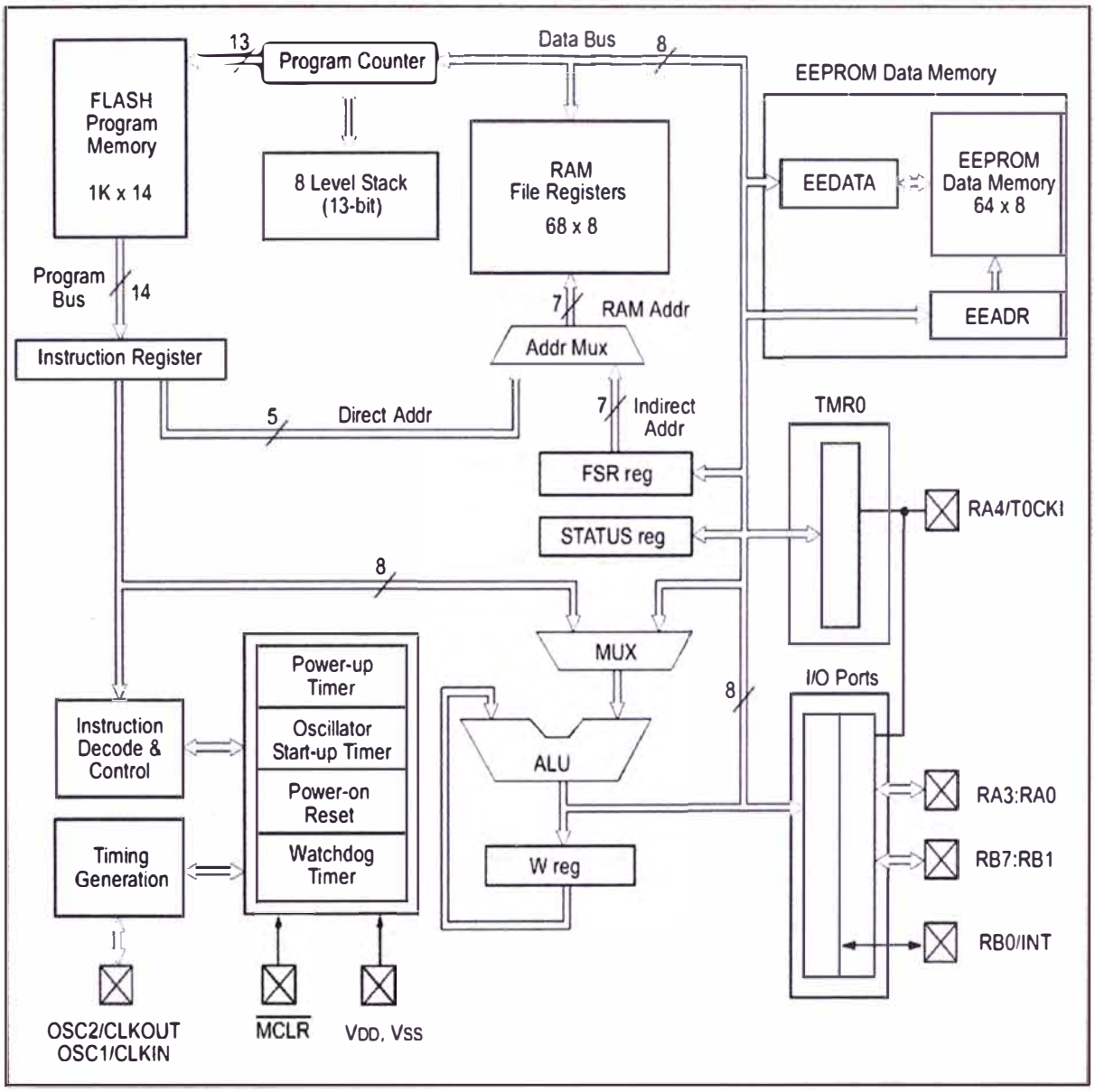


Figura 2.12 Arquitectura Interna del PIC16F84A

2.3.7 Repertorio de Instrucciones

El repertorio del PIC16F84 está compuesto por 35 instrucciones, agrupadas según la operación que cumplen y que acontinuación se detallan:

Nemónico	Descripción
Instrucciones de carga	
Clrf f	00 → f
Clrw	00 → w
Movf f,d	f → (destino)
Movwf f	W → f

Instrucciones de bit		
	bcf f,b	Pone a 0 el bit "b" del reg "f"
	bsf f,b	Pone a 1 el bit "b" del reg "f"
Instrucciones aritmeticas		
	addlw k	W +k → W
	addwf f,d	W + f → (destino)
	decf f,d	f - 1 → (destino)
	incf f,d	f + 1 → (destino)
	sublw k	k - W → W
	subwf f,d	f - W → (destino)
Instrucciones logicas		
	andlw k	W AND k → W
	andwf f,d	W AND f → (destino)
	comf f,d	/f → (destino)
	iorlw k	W OR k → W
	iorwf f,d	W OR f → (destino)
	rlf f,d	Rota f a la izquierda → (destino)
	rrf f,d	Rota f a la derecha → (destino)
	swapf f,d	Intercambia los nibbles de f → (destino)
	xorlw k	W XOR k → W
	xorwf f,d	W XOR f → (destino)
Instrucciones de salto		
	btfsc f,b	Salta si el bit "b" de "f" es 0
	btfss f,b	Salta si el bit "b" de "f" es 1
	decfsz f,d	f - 1 → destino y salta si es 0
	incfsz f,d	f + 1 → destino y salta si es 0
	goto k	Salta a la dirección "k"
Instrucciones de manejo de Subrutinas		
	call k	Llamada a subrutina
	retfie	Retorno de una interrupción
	retlw	Retorno con un literal en W
	return	Retorno de una subrutina
Instrucciones Especiales		
	clrwdt	Borra Timer del Watchdog
	nop	No operación
	sleep	Entra en modo de bajo consumo

En las instrucciones el valor del destino depende del valor que se le de a "d", tal es asi que si d=0, el destino sera el registro de trabajo "W" y si d=1, el destino será el registro "f".

2.3.8 El MPLAB

El MPLAB IDE es un software de "Entorno de Desarrollo Integrado", que se ejecuta bajo Windows. Con este entorno se puede desarrollar aplicaciones para los microcontroladores PIC.

El MPLAB incluye todas las utilidades necesarias para la realización de proyectos con microcontroladores PIC, permite editar el archivo fuente del proyecto, además de ensamblarlo y simularlo en pantalla para comprobar cómo evolucionan tanto la memoria de datos RAM, como la memoria de programa ROM, los registros del SFR, etc, según progresa la ejecución del programa.

El MPLAB incluye:

- Un editor de texto
- Un ensamblador llamado MPASM.
- Un simulador llamado MPLAB SIM.
- Un organizador de proyectos

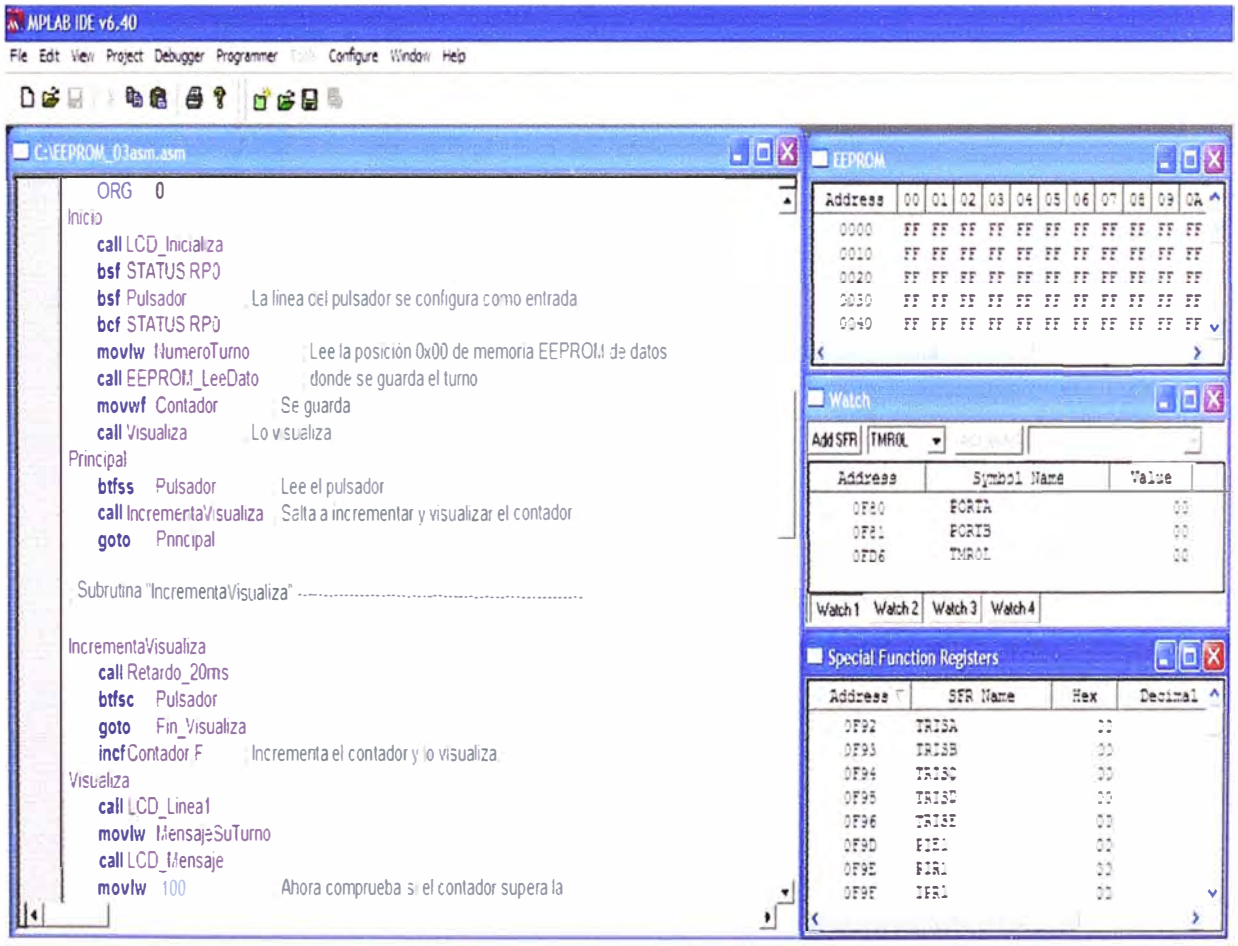


Figura 2.13 El MPLAB

El proceso de ensamblado produce un fichero ejecutable con extensión (*.hex) que será el que posteriormente se grabara en la memoria de programa del PIC mediante el grabado. Ese fichero contiene unicamente numeros hexadecimales, que es la forma de representar los ceros y unos binarios de la información que se grabará posteriormente en la memoria de programa del microcontrolador.

2.4 El Decodificador de tonos DTMF

En nuestro proyecto se utiliza el sistema de marcación por tonos para poder enviar la señal desde el celular emisor al celular receptor, por lo tanto en el celular receptor se necesitara un circuito que realice la decodificación de la señal DTMF.

2.4.1 Frecuencias del DTMF

El sistema de marcación por tonos, también llamado sistema multifrecuencial o DTMF (Dual-Tone Multi-Frequency), consiste en que cuando el usuario pulsa en el teclado de su teléfono la tecla correspondiente al dígito que quiere marcar, se envían dos tonos, de distinta frecuencia uno por columna y otro por fila en la que esté la tecla, que la central descodifica a través de filtros especiales, detectando instantáneamente que dígito se marcó. En la Tabla 2.1 se muestran las frecuencias utilizadas en el DTMF, correspondientes a cada tecla del teléfono celular.

FILA/COLUMNA	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

Tabla 2.1 Distribución de Frecuencias en el DTMF

2.4.2 El MT8870

Existe un solo integrado que realiza todo el proceso de decodificación de tonos DTMF, el circuito integrado utilizado en nuestro proyecto encargado de esta tarea es el MT8870, que con solo unos cuantos elementos externos conformará nuestro circuito decodificador.

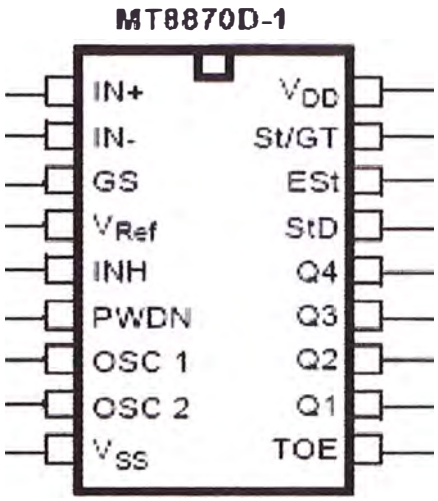


Figura 2.14 El MT8870

El voltaje de alimentación del MT8870 es de 5 voltios con una tolerancia de ± 5%, que se deben conectar entre los pines V_{DD} y V_{SS}.

El MT8870 para poder decodificar la señal DTMF necesita de una señal de reloj, para lo cual posee un circuito oscilador interno que necesita que se le conecte externamente entre los pines OSC1 y OSC2 un cristal de 3.579545 Mhz.

Este integrado tiene una entrada diferencial entre los pines IN+ e IN-, además tiene un voltaje de referencia que sale por la patilla Vref.

El resultado de la señal decodificada es un código digital, que saldrá por los pines Q1, Q2, Q3 y Q4, estas salidas pueden ser bloqueadas poniendo a "0" el pin de habilitación TOE.

Por el pin StD se tendrá un nivel alto mientras se tenga un tono DTMF presente en la entrada del circuito integrado, cuando desaparece el tono en la entrada, el nivel de este pin se pondrá en bajo.

En los otros pines del circuito integrado se conectaran las resistencias y condensadores necesarios para que trabaje el decodificador, el circuito completo del decodificador de tonos se detalla en el siguiente capítulo.

CAPÍTULO III
METODOLOGÍA PARA LA SOLUCIÓN DEL PROBLEMA

3.1 Alternativa de solución

Una alternativa de solución para disminuir los casos de robo de automoviles, es contar con un sistema de alarma moderno, que sea de facil uso, barato, seguro en cuanto a su activación y que además pueda ser activado a distancia, logrando con esto una mayor seguridad para el propietario del vehículo en cuanto al riesgo de poder sufrir daños personales causados por los ladrones.

3.2 Solución del problema

Para la solución del problema se plantea la construcción del sistema de alarma que utilice la tecnología de los celulares, ya que será controlada mediante la comunicación de dos celulares, logrando con esto el control de la alarma a distancia.

Nuestro sistema de alarma cuenta basicamente con 3 elementos: un celular emisor, un celular receptor, una unidad decodificadora y de control.

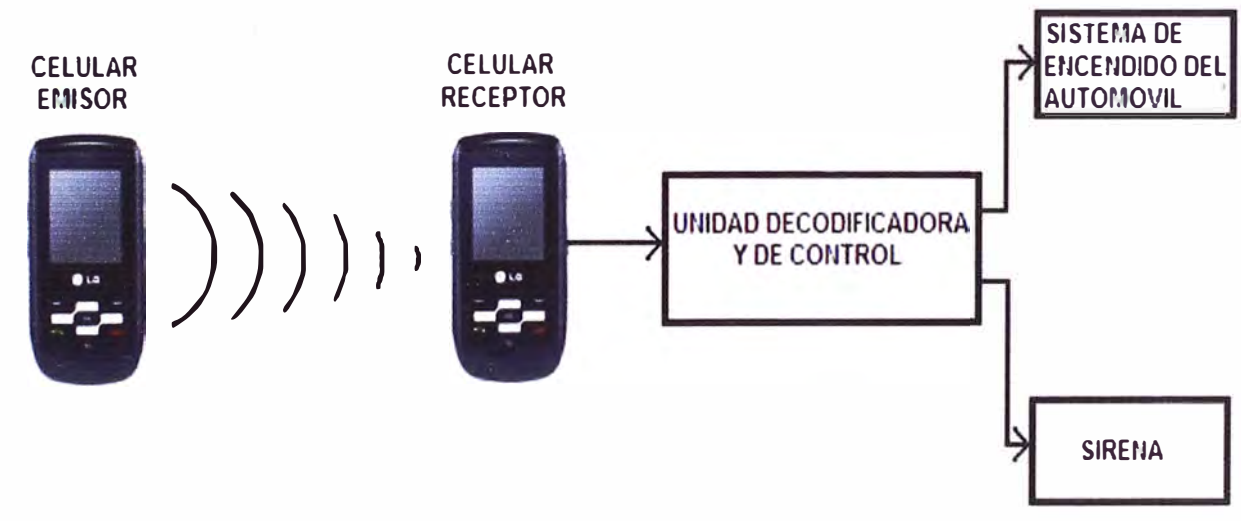


Figura 3.1 Sistema de Seguridad del Automovil

3.2.1 Celular Emisor

Desde el celular emisor se podrá controlar nuestro sistema de alarma a distancia el cual estará en posesión del propietario del vehículo, para ejercer el control este celular mostrara 3 pantallas, la primera pantalla corresponde a un logo, la segunda pantalla nos pedira ingresar una clave, y la tercera pantalla nos mostrara una lista de opciones sobre el control que se puede ejecutar.

El logo solo aparece en la pantalla del celular por unos cuantos segundos y nos sirve para identificar que la aplicación que se esta ejecutando en el celular corresponde a la alarma del automóvil.

Para acceder a las opciones de control con el que cuenta nuestro sistema, se deberá ingresar una clave, si la clave que se ingresa no es la correcta entonces el celular seguira mostrando la pantalla de ingrese su clave, esto seguira asi hasta que se ingrese la clave correcta. Esto de ingresar una clave secreta se fijo asi como un medio de seguridad, para que cualquier persona no pueda operar nuestro sistema de alarma y solo el propietario del vehículo sea el que controle el sistema.

Las opciones con las que cuenta nuestro sistema de alarma son 4, las cuales se detallan a continuación:

- Tecla 2: Sirena on
- Tecla 8: Sirena off
- Tecla 4: Motor on
- Tecla 6: Motor off

En la pantalla del celular apareceran las opciones que se tienen conjuntamente con la tecla que se debe precionar para activar dicha opción. Tal es así que al precionar la tecla 2 del celular se podra activar la sirena que se encuentra en el vehículo, al precionar la tecla 8 la sirena se apagará, al presionar la tecla 4 el motor del automóvil quedará habilitado para poder ser encendido por el conductor, pero si presionamos la tecla 6 el motor del carro quedará bloqueado, con lo cual no se podra encender el vehículo.

Para que nuestro celular emisor pueda hacer las funciones arriba descritas, es necesario instalarle una aplicación de un programa previamente elaborado utilizando el lenguaje de programación JAVA, este programa ha sido elaborado utilizando el Netbeans IDE, que nos permite poder programar en alto nivel, utilizando bloques de programas previamente elaborados y construyendo un diagrama de flujo, tal como se muestra en la figura 3.2.

Del diagrama de flujo elaborado, el Netbeans nos permite obtener en forma automática el programa en forma literal, el cual se muestra a continuación:

Programa en Java

```
package hello;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import org.netbeans.microedition.lcdui.SimpleTableModel;
import org.netbeans.microedition.lcdui.SplashScreen;
```

```
import org.netbeans.microedition.lcdui.TableItem;
public class HelloMIDlet extends MIDlet implements CommandListener {
private boolean midletPaused = false;
private Command exitCommand1;
private Command screenCommand;
private Command okCommand2;
private Command exitCommand;
private Command okCommand;
private Command okCommand1;
private Form form;
private TableItem tableItem;
private SplashScreen splashScreen;
private Form form1;
private TextField textField;
private Alert alert;
private Font font1;
private Image image1;
private SimpleTableModel simpleTableModel;
public HelloMIDlet() {
}
private void initialize () {
}
public void startMIDlet () {
switchDisplayable (null, getSplashScreen ());
}
public void resumeMIDlet () {
}
public void switchDisplayable (Alert alert, Displayable nextDisplayable) {
Display display = getDisplay ();
if (alert == null) {
display.setCurrent (nextDisplayable);
    } else {
display.setCurrent (alert, nextDisplayable);
    }
}
public void commandAction (Command command, Displayable displayable) {
```

```
if (displayable == alert) {
if (command == okCommand2) {
textField.delete(0,4);
switchDisplayable (null, getSplashScreen ());
    }
    } else if (displayable == form) {
if (command == exitCommand) {
exitMIDlet ();
    } else if (command == okCommand) {
try{
pauseApp();
notifyPaused();
platformRequest("tel:985811565");//Colocar el número al que se desea llamar
resumeRequest();
        }catch(Exception e){}
    }
    } else if (displayable == form1) {
if (command == exitCommand1) {
exitMIDlet ();
    } else if (command == okCommand1) {
method ();
    }
    } else if (displayable == splashScreen) {
if (command == SplashScreen.DISMISS_COMMAND) {
switchDisplayable (null, getForm1 ());
    }
    }
}
public Command getExitCommand () {
if (exitCommand == null) {
exitCommand = new Command ("SALIR", Command.EXIT, 0);
    }
return exitCommand;
}
public Form getForm () {
if (form == null) {
```

```
form = new Form ("OPCIONES", new Item[] { getTableItem () });
form.addCommand (getOkCommand ());
form.addCommand (getExitCommand ());
form.setCommandListener (this);
    }
return form;
}
public SplashScreen getSplashScreen () {
if (splashScreen == null) {
splashScreen = new SplashScreen (getDisplay ());
splashScreen.setTitle ("ALARMA AUTOMOVIL");
splashScreen.setCommandListener (this);
splashScreen.setImage (getImage1 ());
splashScreen.setText ("");
splashScreen.setTimeout (4000);
    }
return splashScreen;
}
public Font getFont1 () {
if (font1 == null) {
font1 = Font.getFont (Font.FONT_INPUT_TEXT);
    }
return font1;
    }
public Image getImage1 () {
if (image1 == null) {
try {
image1 = Image.createImage ("/AUTO12.JPG");
    } catch (java.io.IOException e) {
e.printStackTrace ();
    }
    }
return image1;
}
public TableItem getTableItem () {
if (tableItem == null) {
```

```
tableItem = new TableItem (getDisplay (), "");
tableItem.setModel (getSimpleTableModel ());
    }
return tableItem;
}
public SimpleTableModel getSimpleTableModel () {
if (simpleTableModel == null) {
simpleTableModel = new SimpleTableModel (new java.lang.String[][] {
new java.lang.String[] { "2", "SIRENA ON" },
new java.lang.String[] { "8", "SIRENA OFF" },
new java.lang.String[] { "4", "MOTOR ON" },
new java.lang.String[] { "6", "MOTOR OFF" }}, new java.lang.String[] {"TECLA", "ACCION"
});
    }
return simpleTableModel;
}
public Command getOkCommand () {
if (okCommand == null) {
okCommand = new Command ("ACTIVAR", Command.OK, 0);
    }
return okCommand;
}
public void method () {
String valor;
int clave;
valor=textField.getString();
clave=Integer.parseInt(valor);
if (clave==2012) {
 switchDisplayable (null, getForm ());
    } else {
 switchDisplayable (null, getAlert ());
    }
}
public Command getOkCommand1 () {
if (okCommand1 == null) {
 okCommand1 = new Command ("Ok", Command.OK, 0);
```

```
    }
return okCommand1;
}
public Form getForm1 () {
if (form1 == null) {
form1 = new Form ("form1", new Item[] { getTextField () });
form1.addCommand (getOkCommand1 ());
form1.addCommand (getExitCommand1 ());
form1.setCommandListener (this);
    }
return form1;
}
public TextField getTextField () {
if (textField == null) {
 textField = new TextField ("INGRESE CLAVE", null, 4, TextField.NUMERIC
TextField.PASSWORD);
textField.setPreferredSize (-1, -1);
textField.setInitialInputMode ("UCB_BASIC_LATIN");
    }
return textField;
}
public Command getScreenCommand () {
if (screenCommand == null) {
screenCommand = new Command ("Screen", Command.SCREEN, 0);
    }
return screenCommand;
}
public Alert getAlert () {
if (alert == null) {
alert = new Alert ("", "CLAVE INCORRECTA \nINTENTE NUEVAMENTE", null,
AlertType.ERROR);
alert.addCommand (getOkCommand2 ());
alert.setCommandListener (this);
alert.setTimeout (2500);
    }
return alert;
```

```
}
public Command getOkCommand2 () {
if (okCommand2 == null) {
 okCommand2 = new Command ("INICIO", Command.OK, 0);
    }
return okCommand2;
}
public Command getExitCommand1 () {
if (exitCommand1 == null) {
exitCommand1 = new Command ("Exit", Command.EXIT, 0);
    }
return exitCommand1;
}
public Display getDisplay() {
return Display.getDisplay(this);
    }
public void exitMIDlet() {
switchDisplayable(null, null);
destroyApp(true);
notifyDestroyed();
 }
public void startApp() {
if (midletPaused) {
resumeMIDlet();
} else {
initialize();
startMIDlet();
    }
midletPaused = false;
}
public void pauseApp() {
midletPaused = true;
}
public void destroyApp(boolean unconditional) {
    }
}
```

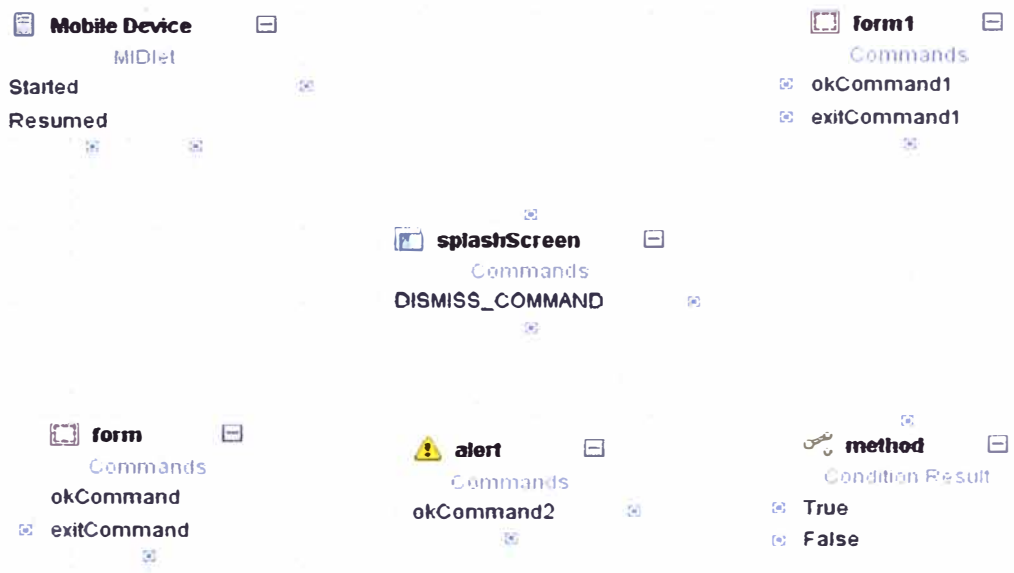


Figura 3.2 Programación en Bloques

3.2.2 Celular Receptor

El celular receptor es el encargado de recibir la señal que le enviá el celular emisor. Cuando se presiona una tecla en el celular emisor este enviá un tono que corresponde a la tecla precionada, tal es así que a cada tecla del celular emisor le corresponderá un tono de frecuencia determinado.

Cuando el celular receptor recibe el tono enviado por el celular emisor, este lo reproduce por su auricular en forma audible, pudiendo asi nosotros escuchar el tono que es recepcionado.

Para poder llevar la señal del tono recibido, del celular receptor a nuestro circuito decodificador utilizamos la salida para audifonos que tiene el celular receptor, al cual le conectamos un conector de auriculares con sus respectivos cables y así tendremos la señal al exterior del celular y lista para ser llevada al circuito decodificador de tonos, tal como se muestra en la Figura 3.3.

Figura 3.3 Conector de Auricular del Celular Receptor

El celular receptor estará instalado en el automóvil en un lugar no visible por lo tanto necesitará de una fuente de alimentación que lo mantenga cargado todo el tiempo, para ello se ha instalado un cargador de celular para auto, al cual se le ha soldado unos cables para poder instalarlo a los 12V que tiene el circuito de alimentación que proviene de la bateria del automóvil, tal como se muestra en la Figura 3.4.

Figura 3.4 Cargador del Celular Receptor

El celular receptor deberá responder en forma automática cuando reciba la llamada del celular emisor, para esto se tendrá que configurar al celular receptor para que conteste automaticamente cuando reciba una llamada, esto se hace entrando a las opciones que tiene el celular y activar la opción de respuesta automática, tal como muestra la Figura 3.5

Figura 3.5 Opción de Respuesta Automática en un Celular

El celular que utilicemos en nuestro proyecto como receptor, necesariamente deberá contar con la opción de respuesta automática y aunque la mayoria de celulares tienen esta opción, se ha elaborado la Tabla 3.1 donde se muestran algunos modelos de celulares de las marcas más comerciales que cuentan con la opción de respuesta automática.

NOKIA	SAMSUNG
1208	E1086
1600	E2120
1800	E2121
2630	F250
2760	S3350
5000	S5230
5030	S5570
5130	S5830
6267	GT-C3222
X2-01	GT-C3520
X3-02	GTS5260

Tabla 3.1 Modelos de Celulares con Respuesta Automática

3.2.3 Circuito Decodificador de Tonos

Este circuito es el encargado de realizar la decodificación de la señal que proviene del celular receptor, la señal que tenemos a la entrada de este circuito es un Tono de frecuencia audible, a la salida del circuito decodificador tendremos un código binario cuyo valor dependerá del tono que se tenga al ingreso.

El alma de este circuito decodificador es el circuito integrado MT8870, el cual añadiendole solo unos cuantos elementos externos realiza todo el trabajo de conversión, logrando asi a su salida tener los códigos binarios que llevaremos a nuestro circuito de control.

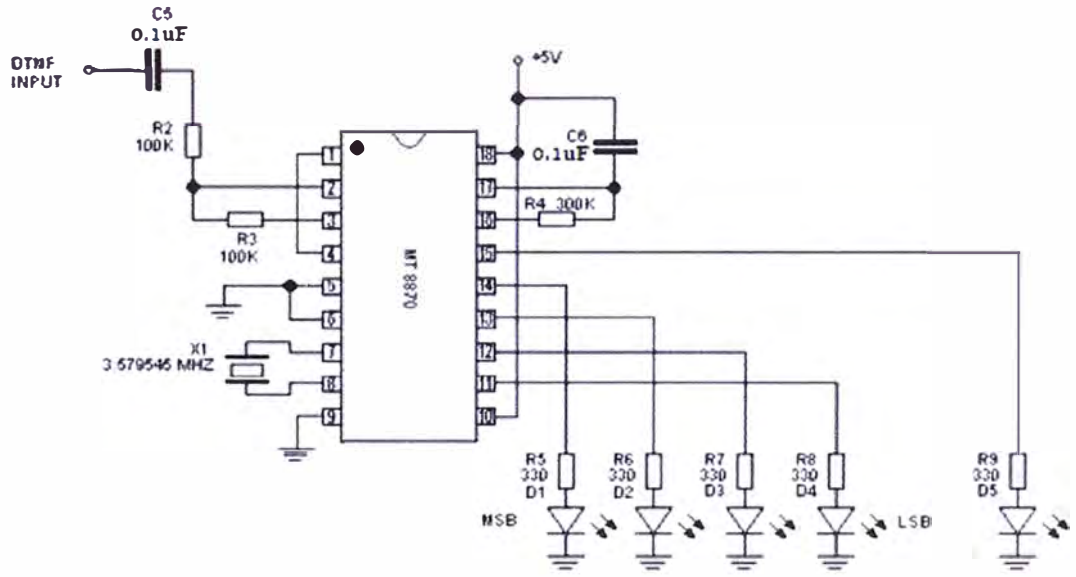


Figura 3.6 Circuito decodificador de tonos DTMF

En el circuito de la figura 3.6 se han conectado 4 leds a la salida del decodificador solo a manera de prueba del decodificador ya que en ellos se podra apreciar la salida del circuito integrado. En el circuito la señal de tonos DTMF, provenientes del celular receptor, ingresan por la patilla 2 del MT8870, a travez de C5 y R2 que sirven como filtro de entrada, luego el integrado decodificador se encargara de detectar el tono DTMF, que proviene del celular receptor, y producirá un pulso por el pin 15 que lo podremos apreciar en el led D5, este pulso se genera cuando se presiona una tecla en el celular emisor y su duración de encendido de este led será durante todo el tiempo que se tenga presionada la tecla; además este circuito integrado por las patillas 11, 12, 13 y 14 nos dará el codigo binario correspondiente al tono que se ingresa al circuito, este código binario lo podremos apreciar en el juego de leds D1, D2, D3 y D4 que se encuentran conectados a la salida del circuito.

A la salida del circuito decodificador se tendrá el código binario correspondiente al número de tecla que se ha presionado en el celular emisor, tal es así que si se presiona la tecla que corresponde al número 1, a la salida del circuito en los leds se tendrá el número binario 0001, con lo cual solo encendara el led D4 y los otros estaran apagados.

3.2.4 Circuito de Control

El circuito de control esta constituido por el microcontrolador PIC16F84A, tal como se muestra en la Figura 3.7, el PIC recibe el código binario procedente de la salida del circuito decodificador y dependiendo del valor del código que se reciba el PIC realizará una determinada acción.

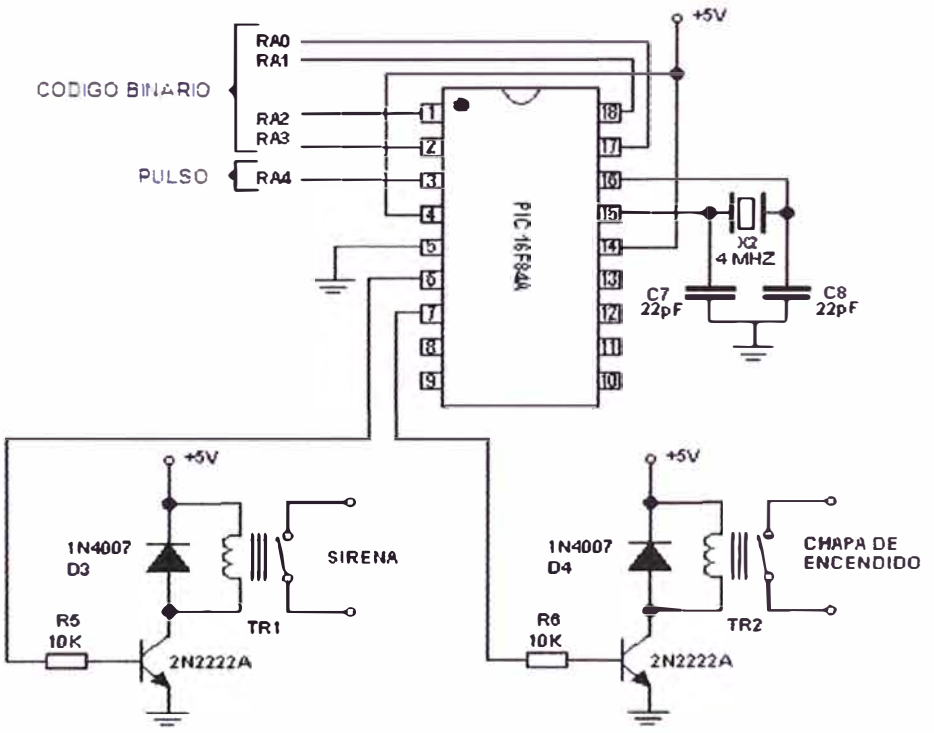


Figura 3.7 Circuito de Control

El circuito de control, a través de los relay que tiene a su salida, podra ejecutar las siguietes acciones:

- Activar la sirena del automóvil
- Desactivar la sirena del automóvil
- Bloquear el sistema de encendido del automóvil
- Desbloquear el sistema de encendido del automóvil

Cada una de las acciones que maneja el circuito de control seran activadas por las teclas que se presionen en el celular emisor, tal es asi que a cada acción le correspondera un determinado número de tecla.

El PIC recibe el código binario, procedente del circuito decodificador, por las entradas RA0, RA1, RA2 y RA3, luego procesa el valor recibido mediante un programa que previamente es cargado en el PIC y según el código que se tenga el PIC ejecutará una determinada acción. El PIC también recibe un pulso en su entrada RA4, el cual le servirá para detectar que una tecla en el celular emisor se ha presionado y apartir de ese momento empesará a procesar el código que tenga en su entrada.

Para que el Microcontrolador realice su tarea, cuenta con un programa, y para realizar este programa partimos del diagrama de bloques que se muestra en la Fig. 3.8, donde se puede apreciar la secuencia lógica que tendra nuestro programa.

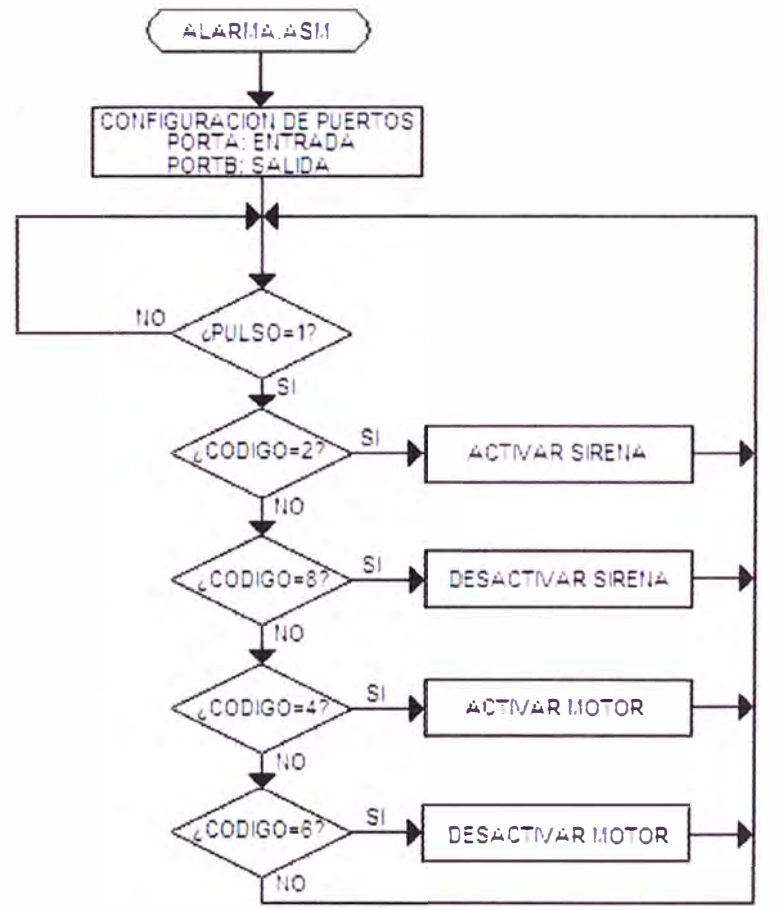


Figura 3.8 Diagrama de Flujo del Programa del Microcontrolador

El programa del Microcontrolador se ha realizado en el lenguaje Ensamblador y es el que se detalla a continuación:

Programa del Microcontrolador

```
__CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC
LIST      P=16F84A
INCLUDE  <P16F84A.INC>
#DEFINE PULSO PORTA,4
CBLOCK 0X0C
CODIGO
CONTA
CONTB
ENDC
        ORG 0
        bsf     STATUS,RP0    ; Acceso al Banco 1.
        clrf    TRISB         ; Las líneas del Puerto B se configuran como salida.
        movlw b'00011111'     ; Las 5 líneas del Puerto A se configuran como entrada.
        movwf TRISA
        bcf     STATUS,RP0    ; Acceso al Banco 0.
        clrf    PORTB         ; Limpia el Puerto B.
        bsf     PORTB,1       ; Activa la salida RA1
TEST
        btfss   PULSO         ; Testea la entrada RA4.
        goto    TEST          ; Sigue testeando.
        call    RETARDO_20ms ; Espera que se estabilice el pulso.
        btfss   PULSO         ; Vueve a testear RA4
        goto    TEST          ; Regresa a testear RA4
        movf    PORTA,W       ; Lee el Puerto A
        andlw   b'00001111'   ; Limpia los 4 bits mas significativos el Puerto A
        movwf CODIGO          ; El resultado se guarda en el registro CODIGO
COMPARACIÓN
        movf    CODIGO,W      ; El CODIGO se guarda en W.
        sublw   d'2'          ; El CODIGO se compara con 2.
        btfsc   STATUS,Z
        goto    SIRENA_ON     ; Si el CODIGO es 2 se salta a activar la Sirena.
        movf    CODIGO,W      ; El CODIGO se guarda en W.
        sublw   d'8'          ; El CODIGO se compara con 8.
```

```
        btfsc   STATUS,Z
        goto    SIRENA_OFF      ; Si el CODIGO es 8 se salta a desactivar la Sirena.
        movf    CODIGO,W        ; El CODIGO se guarda en W.
        sublw   d'4'            ; El CODIGO se compara con 4.
        btfsc   STATUS,Z
        goto    MOTOR_ON        ; Si el CODIGO es 4 se salta a activar Motor.
        movf    CODIGO,W        ; El CODIGO se guarda en W.
        sublw   d'6'            ; El CODIGO se compara con 6.
        btfsc   STATUS,Z
        goto    MOTOR_OFF       ; Si el CODIGO es 6 se salta a desactivar el Motor.
        goto    TEST            ; Se regresa a testear.
SIRENA_ON
        bsf     PORTB,0         ; Se activa la Sirena.
        goto    TEST            ; Se regresa a testear.
SIRENA_OFF
        bcf     PORTB,0         ; Se desactiva la Sirena.
        goto    TEST            ; Se regresa a testear.
MOTOR_ON
        bsf     PORTB,1         ; Se activa el motor.
        goto    TEST            ; Se regresa a testear.
MOTOR_OFF
        bcf     PORTB,1         ; Se desactiva el motor.
        goto    TEST            ; Se regresa a testear.
RETARDO_20ms
        movlw   d'20'
        movwf   CONTB
BucleExterno
        movlw   d'249'
        movwf   CONTA
BucleInterno
        decfsz  CONTA, F
        goto    BucleInterno
        decfsz  CONTB, F
        goto    BucleExterno
        return
    END
```

La interconexión entre los circuitos decodificador de tonos y el controlador se muestra en la Fig. 3.9, en la cual se muestra además el circuito regulador de tensión que se encargará de alimentar a nuestro circuito. Este regulador de tensión lo conforma principalmente el circuito integrado LM7805, el cual se encargará de regular la tensión de 12V proveniente de la bateria del automóvil y reducirla a 5V que es la tensión que requiere nuestro circuito electrónico para funcionar.

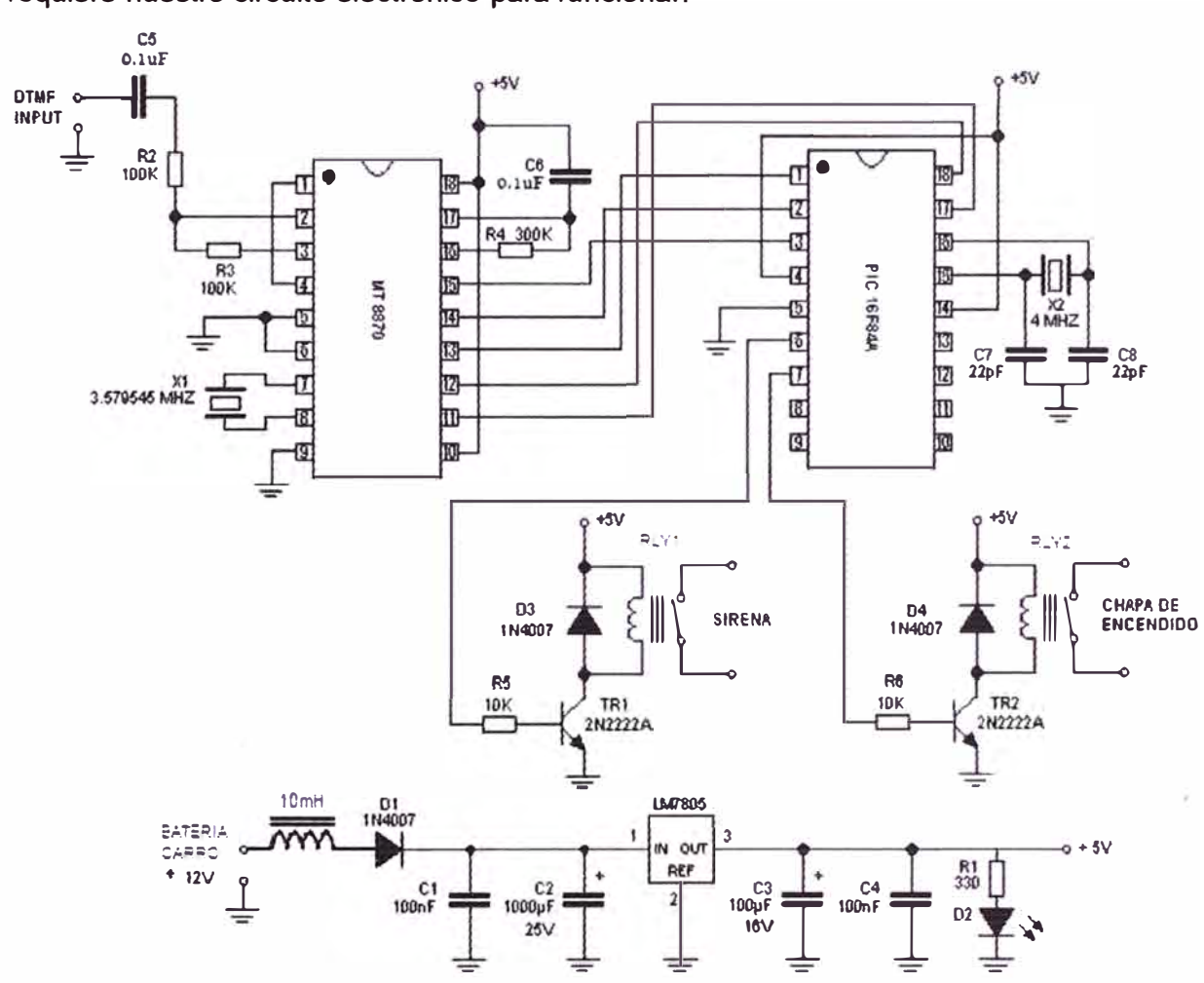


Figura 3.9 Circuito Decodificador y de Control

CAPÍTULO IV
PRESENTACIÓN DE RESULTADOS

4.1 Resultados

En el presente informe se elaboró el diseño de un sistema de alarma para automóviles controlado a distancia, mediante el uso de un teléfono celular, presentandose los siguientes resultados:

Al utilizar un teléfono celular para comandar la alarma, nos permitirá poder utilizar todos los recursos que este posee, como por ejemplo utilizar pantallas graficas, tabla de opciones, ingreso de clave, uso del teclado del celular, etc. Todo esto trae como resultado un manejo facil y seguro de nuestro sistema de alarma.

Para iniciar el control de nuestro sistema de alarma, se deberá buscar y activar la aplicación en Java que se encuentra instalado en el celular emisor. Esta aplicación es un simple archivo, que se encuentra guardado en la memoria del celular, el cual se deberá seleccionar y ejecutar.

Una vez seleccionada la aplicación en Java, el programa empezará a ejecutarse, mostrando en la pantalla del celular un grafico de un automóvil con un titulo que indica "Alarma Automóvil", tal como se muestra en la Fig. 4.1, con este grafico y el mensaje que muestra el celular emisor, el usuario podra saber que el programa de control de la alarma se ha iniciado.

Figura 4.1 Logo del Sistema de Alarma

Después de unos cuantos segundos desaparece automáticamente el grafico y la pantalla cambia a una nueva, en la cual se nos pide que ingresemos una clave secreta para poder continuar con el manejo del sistema, tal como se muestra en la Fig 4.2, esto se hace para que solo el usuario pueda acceder al control del sistema de alarma, dando asi una mayor seguridad al manejo del sistema.

Figura 4.2 Ingreso de Clave Secreta

En caso que se ingrese un código erroneo, aparcera un mensaje indicandonos que el código ingresado es incorrecto y nos volvera a pedir que ingresemos la clave correcta, tal como se muestra en la Figura 4.3.

Figura 4.3 Mensaje de Clave Incorrecta

Una vez que se ha ingresado la clave correcta, en la pantalla del celular aparecerá un cuadro, el cual nos indica las opciones que se tienen para poder activarlas con el celular, además nos indica que tecla del celular se debe presionar para activar una determinada acción, esto se muestra en la Figura 4.4.

Figura 4.4 Cuadro de Opciones del Sistema de Alarma

En la misma pantalla del cuadro de opciones, se encuentra el comando de "Activar", el cual deberá seleccionarse para que el celular emisor realice automáticamente la llamada al celular receptor que se encuentra escondido en el automóvil y cuando este conteste la llamada en forma automática se lograra establecer una comunicación entre ambos celulares, y el sistema quedara listo para accionar cualquier opción que se active desde el celular emisor.

Como los celulares emisor y receptor se comunican mediante una llamada común y corriente, es posible poder escuchar, si es que se quiere, desde el celular emisor todo lo que acontece en el vehículo robado, tal es asi que se puede escuchar el ruido del motor del vehículo, con lo cual se puede saber si el carro esta encendido o apagado; asi mismo es posible tambien poder escuchar las conversaciones que se den dentro del automóvil.

Mediante el celular receptor, el circuito decodificador y de control que se encuentra instalado en el automóvil, se ha logrado poder recibir y ejecutar ordenes desde el celular emisor que se encuentra a distancia, por esto es que se puede apagar el vehículo a distancia en caso de haberse producido un robo, lo que da al propietario una opción de poder recuperar su vehículo en forma eficaz y segura.

Aunque este sistema de alarma para automóvil tiene sus limitaciones frente al sistema de alarma por GPS, ya que con el GPS aparte de poder gobernar el bloqueo de encendido del motor, también es posible determinar la ubicación del automóvil robado, pero la ventaja de nuestro sistema esta en que el costo para su construcción es bajo, tal como se muestra en la tabla 4.1, en donde se calcula el costo total de nuestro sistema en

S/250, en cambio en el sistema con GPS solo el costo del equipo para GPS cuesta S/500, aparte se debe pagar una mensaluidad de S/70, según la empresa "Satellital Patrol", por otra parte en nuestro sistema no se necesita pagar mensualidad, solo se deberá hacer una recarga al celular receptor cada 3 meses de S/5, para poder mantener la línea activa. Por lo tanto estamos obteniendo un sistema de alarma que su beneficio frente al GPS es que representa un costo más bajo.

CANTIDAD	COMPONENTE O EQUIPO	P. UNIT. S/.	TOTAL S/.
2	Celulares Nokia 100	99.00	198.00
1	C.I. PIC16F84A	9.00	9.00
1	C.I. MT8870	4.00	4.00
1	C.I. 7805	1.50	1.50
2	Transistores 2N2222A	0.50	1.00
4	Condensadores cerámicos 0,1uF/50V	0.20	0.80
2	Condensadores cerámicos 22pF/50V	0.20	0.40
2	Condensador electrolítico 1000uF/25V y 100uF/16V	0.50	1.00
1	Bobina de choque de 10mH 500mA	5.00	5.00
1	Diodo led rojo	0.30	0.30
3	Diodo rectificador 1N4007	0.30	0.90
1	Resistencia 330Ω	0.03	0.03
2	Resistencias 10KΩ	0.03	0.06
2	Resistencias 100KΩ	0.03	0.06
1	Resistencias 300KΩ	0.03	0.03
2	Relay 5Vdc / 24Vdc 1A.	2.50	5.00
2	Cristales 4 MHz. y de 3.579545 MHz	0.80	1.60
1	Tarjeta de impreso	5.00	5.00
1	Caja de plastico de 10x10x3 cm.	6.00	6.00
1	Otros	10.00	10.00
COSTO TOTAL			**249.68**

Tabla 4.1 Costo Total del Sistema de Alarma para Automóvil

CONCLUSIONES Y RECOMENDACIONES

En el presente informe se ha planteado una alternativa de solución a un problema real, como es el robo de automóviles en nuestra ciudad, tratando de que si bien no se elimine por completo el problema, pero al menos se redusca el número de robos de autos, además que también se resdusca en gran manera el saldo de muertos que deja esta problemática, cuando el propietario del vehículo opone resistencia al ser victima del robo de su vehículo.

En el desarrollo de nuestro sistema de alarma, se han utilizado en gran parte los conocimientos de la Electrónica digital en conjunto con software de programación de celulares y de microcontroladores, con lo cual se tiene un sistema moderno, eficaz y seguro.

Al haber utilizado en nuestro sistema componentes comerciales y baratos, el costo de nuestro sistema es bastante bajo, pudiendo ser asi accesible para cualquier automovilista de nuestro medio.

Si en nuestro sistema de alarma se utilizaran celulares con video llamada, sería posible ver en tiempo real, desde el celular emisor, el video del interior del vehículo, logrando así poder grabar o tomar fotos de los delincuentes autores del robo, lo que permitiría su posterior reconocimiento y captura.

Para el buen funcionamiento de nuestro sistema de alarma, es necesario que se logre establecer la comunicación telefónica entre el celular emisor y el celular receptor, si no se puediese dar esta comunicación nuestro sistema de alarma no funcionaría, por lo tanto es indispensable que para que trabaje nuestro sistema de alarma se tiene que tener cobertura y credito en los celulares.

Es recomendable instalar el programa de Java en más de un celular, ya que si se da el caso que nos roben el auto y el celular que tenemos en uso, podriamos activar nuestro sistema de alarma desde el otro celular que tenga el programa Java de nuestro sistema de seguridad.

ANEXO A

DATASHEET DEL MICROCONTROLADOR PIC16F84A

MICROCHIP

PIC16F84A

18-pin *Enhanced* FLASH/EEPROM 8-Bit Microcontroller

High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single-cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- 1024 words of program memory
- 68 bytes of Data RAM
- 64 bytes of Data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 Special Function Hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
  - External RB0/INT pin
  - TMR0 timer overflow
  - PORTB<7:4> interrupt-on-change
  - Data EEPROM write complete

Peripheral Features:

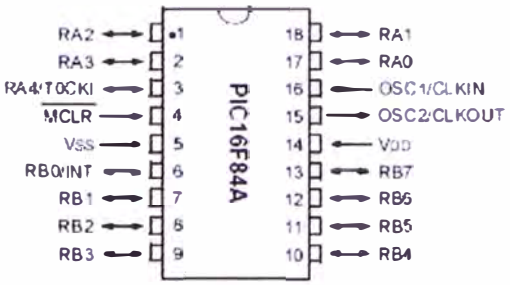
- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Special Microcontroller Features:

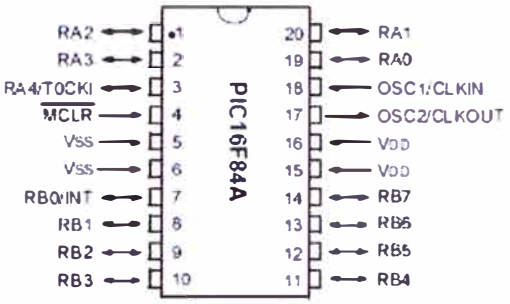
- 10,000 erase/write cycles *Enhanced* FLASH Program memory typical
- 10,000,000 typical erase/write cycles EEPROM Data memory typical
- EEPROM Data Retention > 40 years
- In-Circuit Serial Programming™ (ICSP™) - via two pins
- Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Code protection
- Power saving SLEEP mode
- Selectable oscillator options

Pin Diagrams

PDIP, SOIC



SSOP



CMOS Enhanced FLASH/EEPROM Technology:

- Low power, high speed technology
- Fully static design
- Wide operating voltage range:
  - Commercial: 2.0V to 5.5V
  - Industrial: 2.0V to 5.5V
- Low power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 15 µA typical @ 2V, 32 kHz
  - < 0.5 µA typical standby current @ 2V

PIC16F84A

TABLE 1-1: PIC16F84A PINOUT DESCRIPTION

Pin Name	PDIP No.	SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	18	I	ST/CMOS(3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	19	O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKOUT, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
$\overline{MCLR}$	4	4	4	I/P	ST	Master Clear (Reset) input/programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0	17	17	19	I/O	TTL	
RA1	18	18	20	I/O	TTL	
RA2	1	1	1	I/O	TTL	
RA3	2	2	2	I/O	TTL	
RA4/T0CKI	3	3	3	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	7	I/O	TTL/ST(1)	RB0/INT can also be selected as an external interrupt pin.
RB1	7	7	8	I/O	TTL	
RB2	8	8	9	I/O	TTL	
RB3	9	9	10	I/O	TTL	
RB4	10	10	11	I/O	TTL	Interrupt-on-change pin.
RB5	11	11	12	I/O	TTL	Interrupt-on-change pin.
RB6	12	12	13	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming clock.
RB7	13	13	14	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming data.
VSS	5	5	5,6	P	—	Ground reference for logic and I/O pins.
VDD	14	14	15,16	P	—	Positive supply for logic and I/O pins.

Legend: I= input O = Output I/O = Input/Output P = Power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

PIC16F84A

2.2 Data Memory Organization

The data memory is partitioned into two areas. The first is the Special Function Registers (SFR) area, while the second is the General Purpose Registers (GPR) area. The SFRs control the operation of the device.

Portions of data memory are banked. This is for both the SFR area and the GPR area. The GPR area is banked to allow greater than 116 bytes of general purpose RAM. The banked areas of the SFR are for the registers that control the peripheral functions. Banking requires the use of control bits for bank selection. These control bits are located in the STATUS Register. Figure 2-2 shows the data memory map organization.

Instructions MOVWF and MOVF can move values from the W register to any location in the register file ("F"), and vice-versa.

The entire data memory can be accessed either directly using the absolute address of each register file or indirectly through the File Select Register (FSR) (Section 2.5). Indirect addressing uses the present value of the RP0 bit for access into the banked areas of data memory.

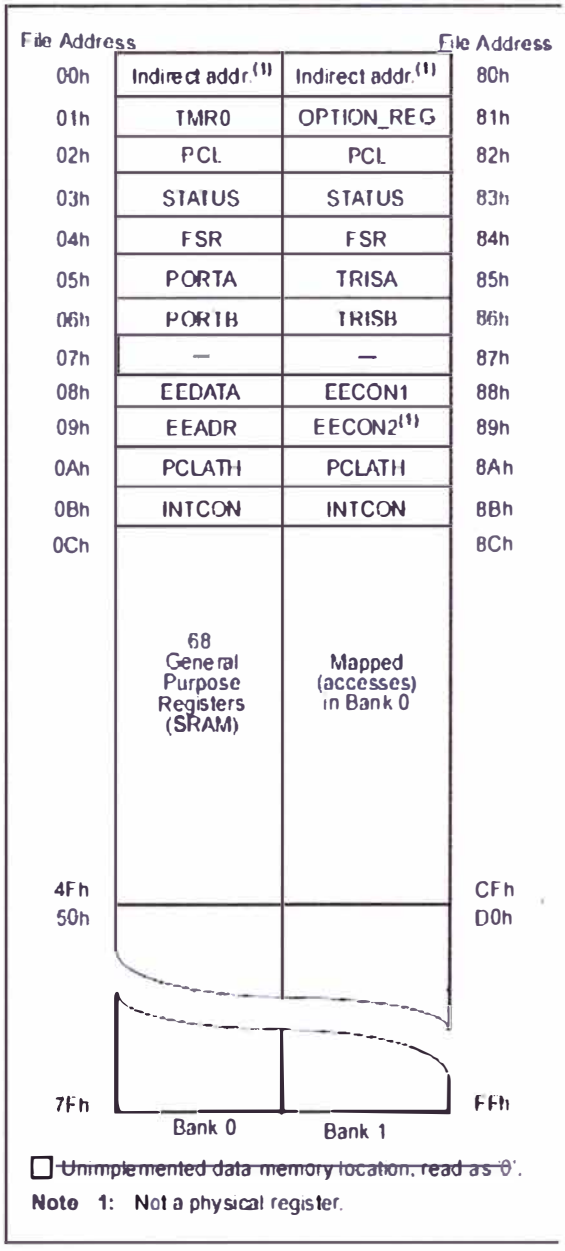
Data memory is partitioned into two banks which contain the general purpose registers and the special function registers. Bank 0 is selected by clearing the RP0 bit (STATUS<5>). Setting the RP0 bit selects Bank 1. Each Bank extends up to 7Fh (128 bytes). The first twelve locations of each Bank are reserved for the Special Function Registers. The remainder are General Purpose Registers, implemented as static RAM.

2.2.1 GENERAL PURPOSE REGISTER FILE

Each General Purpose Register (GPR) is 8-bits wide and is accessed either directly or indirectly through the FSR (Section 2.5).

The GPR addresses in Bank 1 are mapped to addresses in Bank 0. As an example, addressing location 0Ch or 8Ch will access the same GPR.

FIGURE 2-2: REGISTER FILE MAP - PIC16F84A



2.3 Special Function Registers

The Special Function Registers (Figure 2-2 and Table 2-1) are used by the CPU and Peripheral functions to control the device operation. These registers are static RAM.

The special function registers can be classified into two sets, core and peripheral. Those associated with the core functions are described in this section. Those related to the operation of the peripheral features are described in the section for that specific feature.

TABLE 2-1: SPECIAL FUNCTION REGISTER FILE SUMMARY

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on RESET	Details on page
Bank 0											
00h	INDF	Uses contents of FSR to address Data Memory (not a physical register)								---- ----	11
01h	TMR0	8-bit Real-Time Clock/Counter								xxxx xxxx	20
02h	PCL	Low Order 8 bits of the Program Counter (PC)								0000 0000	11
03h	STATUS(2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
04h	FSR	Indirect Data Memory Address Pointer 0								xxxx xxxx	11
05h	PORTA(4)	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	16
06h	PORTB(5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	18
07h	—	Unimplemented location, read as '0'								—	—
08h	EEDATA	EEPROM Data Register								xxxx xxxx	13,14
09h	EEADR	EEPROM Address Register								xxxx xxxx	13,14
0Ah	PCLATH	—	—	—	Write Buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10
Bank 1											
80h	INDF	Uses Contents of FSR to address Data Memory (not a physical register)								---- ----	11
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	9
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	11
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	11
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	16
86h	TRISB	PORTB Data Direction Register								1111 1111	18
87h	—	Unimplemented location, read as '0'								—	—
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	13
89h	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	14
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0', q = value depends on condition

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> are never transferred to PCLATH.

2: The $\overline{TO}$ and $\overline{PD}$ status bits in the STATUS register are not affected by a $\overline{MCLR}$ Reset.

3: Other (non power-up) RESETS include: external RESET through MCLR and the Watchdog Timer Reset.

4: On any device RESET, these pins are configured as input.

5: This is the value that will be in the port output latch.

ANEXO B

DATASHEET DEL DECODIFICADOR DE TONOS MT8870D

ISO²-CMOS MT8870D/MT8870D-1
Integrated DTMF Receiver

ISSUE 3 May 1995

Features
- Complete DTMF Receiver
- Low power consumption
- Internal gain setting amplifier
- Adjustable guard time
- Central office quality
- Power-down mode
- Inhibit mode
- Backward compatible with MT8870C/MT8870C-1

Applications
- Receiver system for British Telecom (BT) or CEPT Spec (MT8870D-1)
- Paging systems
- Repeater systems/mobile radio
- Credit card systems
- Remote control
- Personal computers
- Telephone answering machine

Ordering Information

MT8870DE/DE-1	18 Pin Plastic DIP
MT8870DC/DC-1	18 Pin Ceramic DIP
MT8870DS/DS-1	18 Pin SOIC
MT8870DN/DN-1	20 Pin SSOP
MT8870DT/DT-1	20 Pin TSSOP

-40 °C to +85 °C

Description

The MT8870D/MT8870D-1 is a complete DTMF receiver integrating both the bandsplit filter and digital decoder functions. The filter section uses switched capacitor techniques for high and low group filters; the decoder uses digital counting techniques to detect and decode all 16 DTMF tone-pairs into a 4-bit code. External component count is minimized by on chip provision of a differential input amplifier, clock oscillator and latched three-state bus interface.

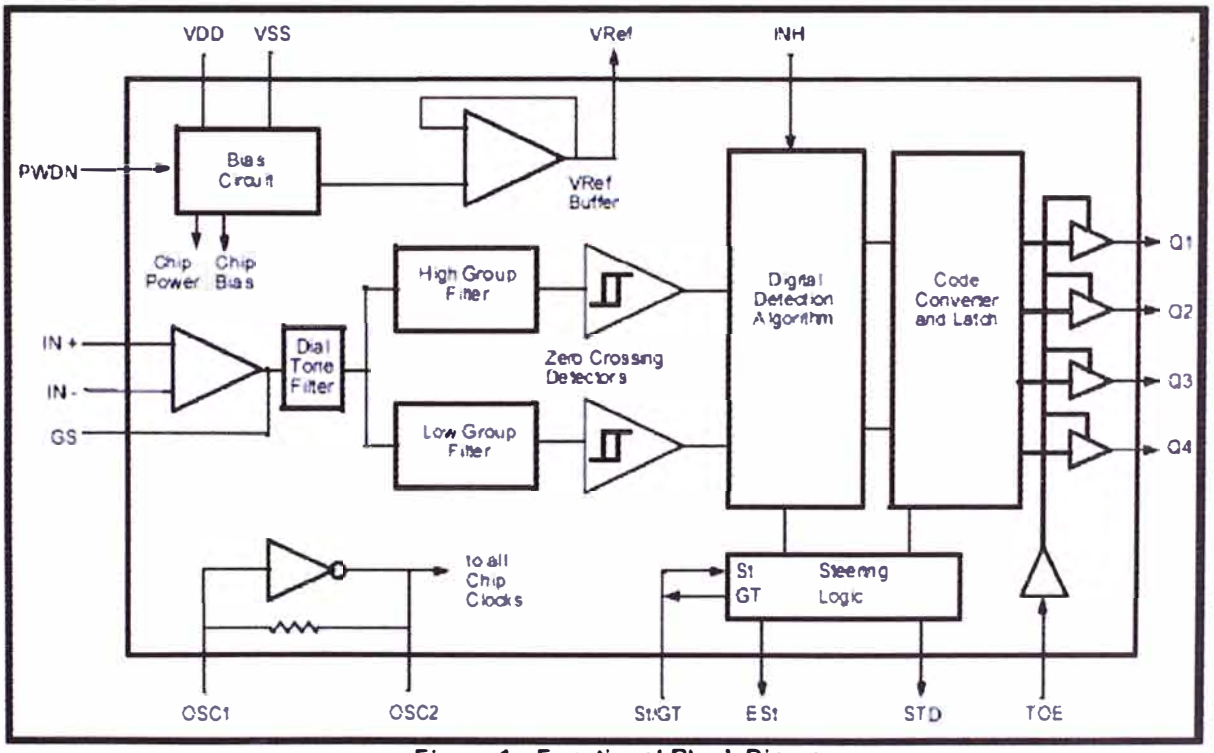


Figure 1 - Functional Block Diagram

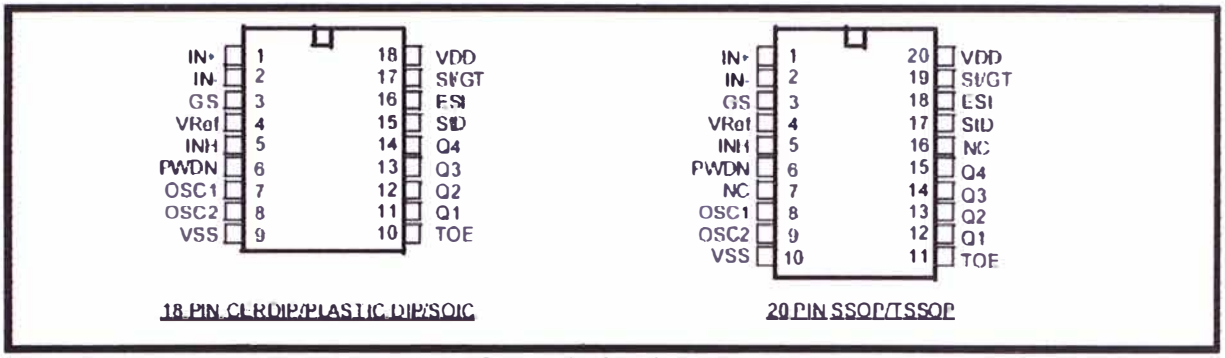


Figure 2 - Pin Connections

Pin Description

Pin #		Name	Description
18	20		
1	1	IN+	**Non-Inverting Op-Amp (Input).**
2	2	IN-	**Inverting Op-Amp (Input).**
3	3	GS	**Gain Select.** Gives access to output of front end differential amplifier for connection of feedback resistor.
4	4	V_{Ref}	**Reference Voltage (Output).** Nominally $V_{DD}/2$ is used to bias inputs at mid-rail (see Fig. 6 and Fig. 10).
5	5	INH	**Inhibit (Input).** Logic high inhibits the detection of tones representing characters A, B, C and D. This pin input is internally pulled down.
6	6	PWDN	**Power Down (Input).** Active high. Powers down the device and inhibits the oscillator. This pin input is internally pulled down.
7	8	OSC1	**Clock (Input).**
8	9	OSC2	**Clock (Output).** A 3.579545 MHz crystal connected between pins OSC1 and OSC2 completes the internal oscillator circuit.
9	10	V_{SS}	**Ground (Input).** 0V typical.
10	11	TOE	**Three State Output Enable (Input).** Logic high enables the outputs Q1-Q4. This pin is pulled up internally.
11-14	12-15	Q1-Q4	**Three State Data (Output).** When enabled by TOE, provide the code corresponding to the last valid tone-pair received (see Table 1). When TOE is logic low, the data outputs are high impedance.
15	17	StD	**Delayed Steering (Output).** Presents a logic high when a received tone-pair has been registered and the output latch updated; returns to logic low when the voltage on St/GT falls below V_{TSt}.
16	18	ESt	**Early Steering (Output).** Presents a logic high once the digital algorithm has detected a valid tone pair (signal condition). Any momentary loss of signal condition will cause ESt to return to a logic low.
17	19	St/GT	**Steering Input/Guard time (Output) Bidirectional.** A voltage greater than V_{TSt} detected at St causes the device to register the detected tone pair and update the output latch. A voltage less than V_{TSt} frees the device to accept a new tone pair. The GT output acts to reset the external steering time-constant; its state is a function of ESt and the voltage on St.
18	20	V_{DD}	**Positive power supply (Input).** +5V typical.
	7, 16	NC	No Connection.

ANEXO C

DATASHEET DEL REGULADOR DE VOLTAJE LM7805

FAIRCHILD
SEMICONDUCTOR®

www.fairchildsemi.com

MC78XX/LM78XX/MC78XXA

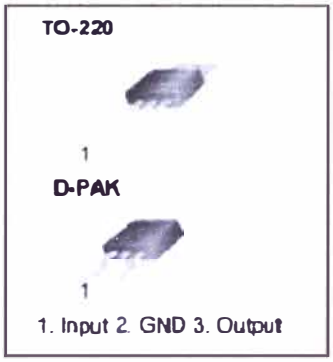
3-Terminal 1A Positive Voltage Regulator

Features

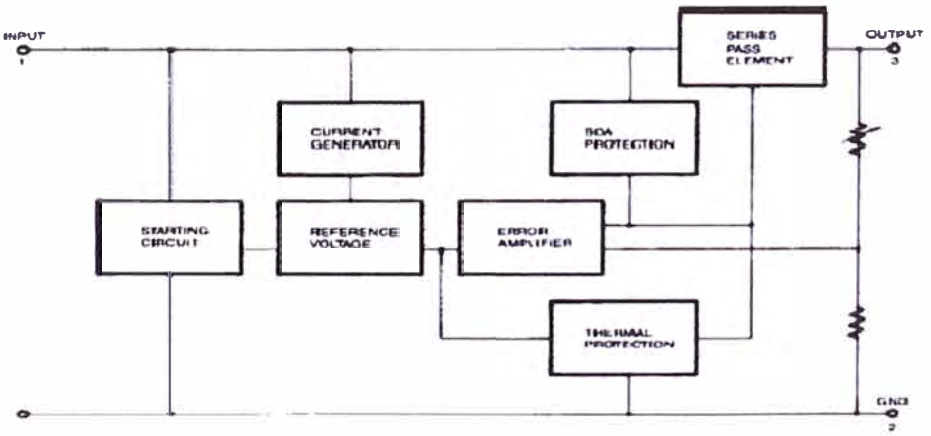
- Output Current up to 1A
- Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V
- Thermal Overload Protection
- Short Circuit Protection
- Output Transistor Safe Operating Area Protection

Description

The MC78XX/LM78XX/MC78XXA series of three terminal positive regulators are available in the TO-220/D-PAK package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut down and safe operating area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.



Internal Block Digram



Rev. 1.0.1

Absolute Maximum Ratings

Parameter	Symbol	Value	Unit
Input Voltage (for V_O = 5V to 18V) (for V_O = 24V)	V_I V_I	35 40	V V
Thermal Resistance Junction-Cases (TO-220)	$R_{\theta JC}$	5	°C/W
Thermal Resistance Junction-Air (TO-220)	$R_{\theta JA}$	65	°C/W
Operating Temperature Range	T_{OPR}	0 ~ +125	°C
Storage Temperature Range	T_{STG}	-65 ~ +150	°C

Electrical Characteristics (MC7805/LM7805)

(Refer to test circuit ,0°C < T_J < 125°C, I_O = 500mA, V_I = 10V, C_I= 0.33μF, C_O= 0.1μF, unless otherwise specified)

Parameter	Symbol	Conditions		MC7805/LM7805			Unit
				Min.	Typ.	Max.	
Output Voltage	V_O	T_J =+25°C		4.8	5.0	5.2	V
		5.0mA ≤ Io ≤ 1.0A, P_O ≤ 15W V_I = 7V to 20V		4.75	5.0	5.25	
Line Regulation (Note1)	Regline	T_J=+25°C	V_O = 7V to 25V	-	4.0	100	mV
			V_I = 8V to 12V	-	1.6	50	
Load Regulation (Note1)	Regload	T_J=+25°C	I_O = 5.0mA to1.5A	-	9	100	mV
			I_O =250mA to 750mA	-	4	50	
Quiescent Current	I_Q	T_J =+25°C		-	5.0	8.0	mA
Quiescent Current Change	ΔI_Q	I_O = 5mA to 1.0A		-	0.03	0.5	mA
		V_I= 7V to 25V		-	0.3	1.3	
Output Voltage Drift	$\Delta V_O/\Delta T$	I_O= 5mA		-	-0.8	-	mV/°C
Output Noise Voltage	V_N	f = 10Hz to 100KHz, T_A=+25°C		-	42	-	μV/Vo
Ripple Rejection	RR	f = 120Hz V_O = 8V to 18V		62	73	-	dB
Dropout Voltage	V_{Drop}	I_O = 1A, T_J =+25°C		-	2	-	V
Output Resistance	r_O	f = 1KHz		-	15	-	mΩ
Short Circuit Current	I_{SC}	V_I = 35V, T_A =+25°C		-	230	-	mA
Peak Current	I_{PK}	T_J =+25°C		-	2.2	-	A

Note:

1. Load and line regulation are specified at constant junction temperature. Changes in V_O due to heating effects must be taken into account separately. Pulse testing with low duty is used.

BIBLIOGRAFIA

[1] Sergio Gálvez Rojas y Lucas Ortega Diaz,"Java a Tope J2ME", Universidad de Malaga España-2003.

[2] Deitel Paul J. y Harvey M. Deitel, "Java como programar", Pearson Educación, Mexico 2008.

[3] Jonathank Knudsen and Sing Li, "Beginning J2ME", Apress, United States of America 2005.

[4] Maximiliano R. Firtman, "Programación para celulares con Java", MP Ediciones, Buenos Aires, Argentina 2004.

[5] Robert L. Boylestad y Louis Nashelsky "Electrónica: Teoría de circuitos y dispositivos electrónicos" Pearson Educación, Mexico 2003.

[6] Enrique Palacios, Fernando Remiro, Lucas J. Lopez, "Microcontrolador PIC16F84, De-Desarrollo de Proyectos", Alfaomega Grupo Editor, México-2004.

[7] Jose M. Angulo Usategui, Ignacio Angulo Martínez, "Microcontroladores PIC, Diseño práctico de Aplicaciones", McGraw-Hill/Interamericana de España-2001.

[8] Página web catálogo de los semiconductores: www.alldatasheet.com

[9] Página web de la compañía microchip, fabricantes de los microcontroladores PIC: www.microchip.com

[10] Página web oficial de Java: www.java.sun.com

[11] Página web de documentos y artículos para los desarrolladores de programas Java: www.microjava.com

[12] Página web de herramientas, documentos e información técnica sobre los teléfonos celulares de Nokia: www.forum.nokia.com

[13] Página web apuntes del circuito integrado MT8870: www.zarlin.com/zarlink/msan108-appnote.pdf