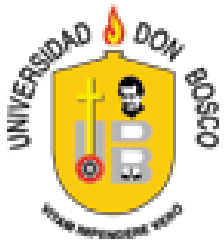
UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERÍA

ESCUELA DE ELECTRÓNICA



DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE BLOQUEO Y DESBLOQUEO DE DISPOSITIVOS, UTILIZANDO TECNOLOGÍA GSM APLICADO A UN AUTOMÓVIL

TRABAJO DE GRADUACIÓN

PRESENTADO POR

Marvin Alexander Martínez Rodríguez	MR-000051
Carlos José Zelaya Rodríguez	ZR-000371
Néstor Iván Zelaya Rodríguez	ZR-000286

PARA OPTAR AL GRADO DE

Ingeniero en Electrónica

Asesor:

Ing. Juan Carlos Castro

Febrero de 2006

Soyapango – El Salvador – Centro América

UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE ELECTRÓNICA

AUTORIDADES:

RECTOR
ING. FEDERICO HUGUET RIVERA

VICERRECTOR ACADÉMICO
PBRO. VÍCTOR BERMÚDEZ, sdb

SECRETARIO GENERAL
LIC. MARIO RAFAEL OLMOS

DECANO DE LA FACULTAD DE INGENIERÍA
ING. GODOFREDO GIRÓN

DIRECTOR DE ESCUELA DE ELECTRÓNICA
ING. OSCAR DURÁN VIZCARRA

ASESOR DEL TRABAJO DE GRADUACIÓN
ING. JUAN CARLOS CASTRO

UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE ELECTRÓNICA

JURADO EVALUADOR DEL TRABAJO DE GRADUACIÓN

Ing. Wenceslao Rivas
JURADO

Ing. Angel Soriano
JURADO

Ing. Juan Carlos Cruz Dada
JURADO

Ing. Juan Carlos Castro
ASESOR

DEDICATORIAS

Este logro es dedicado primeramente a **DIOS** por todo lo que tengo, por todo lo que soy, por todo lo que hecho y seguiré haciendo, por brindarme a lo largo de mi vida a las personas correctas que me han sabido guiar. Unos de ellos son mis **PADRES**. Sé que más que un triunfo mío, es un triunfo de ustedes, y esta meta cumplida es el reflejo, por lo que tanto se sacrificaron y les estoy agradecido. Padre se que desde el cielo estás orgulloso de mi.

A mis **HERMANOS** que estuvieron siempre para apoyarme, acompañarme y ayudarme. A mis **FAMILIARES**, por brindarme sus consejos. A mis **AMIG@S** por darme su confianza, apoyarme y hacer que levante los ánimos en momentos de preocupación. Y a mi **NOVIA**, en quien encuentro un apoyo, refugio y fortaleza para seguir adelante. A cada uno de ustedes, le dedico mi logro debido a que han formado parte de mi vida y me han ayudado, ya sea directa o indirecta, en la obtención del mismo.

MARVIN ALEXANDER MARTÍNEZ

DEDICATORIAS

A DIOS TODO PODEROSO por que sin el nada es posible e iluminar cada paso de mi vida, **A MIS PADRES** Dina Rodríguez de Zelaya y Rodolfo Zelaya por confiar en mi y apoyarme en cada etapa de mi vida, por su cariño y comprensión, les agradezco infinitamente por sus consejos y guiarme en el buen camino, los quiero con todo mi corazón. **A MIS HERMANOS**, Maria Elena Zelaya, Maria Eugenia Zelaya y Néstor Zelaya, por haberme apoyado y ayudado cada vez que los necesitaba. **A MIS FAMILIARES**, por apoyarme y mostrarme su cariño e interés, **A MIS COMPAÑEROS DE TESIS Y ASESOR**, quienes estuvieron conmigo en todos los momentos de esta etapa de mi vida, les agradezco mucho. **A MIS AMIGOS**: Sería una larga lista, pero todos los que estuvieron en las buenas y malas para apoyarme y salir adelante día a día, les agradezco su amistad y comprensión en todo momento.

CARLOS ZELAYA

DEDICATORIAS

Primeramente a **DIOS** por permitirme llegar a esta etapa de mi vida; **A MI PADRES** por haberme apoyado en todo momento y haber tenido confianza en mi de lograr las metas que me propuse, este logro no es solo mío sino también de ellos; **A MIS HERMANOS** que siempre supieron saber que decir en los momentos difíciles y lograr levantarme el ánimo cuando lo necesité, **A MIS AMIGOS Y AMIGAS** con los que he compartido tantas experiencias, especialmente aquellos a los que cuando busque siempre estaban allí para darme su apoyo, que me supieron **C**omprender en **L**os momentos difíciles y que me tuvieron **L**a **P**aciencia para escucharme cuando lo necesitaba, **A MI ASESOR** por habernos encaminado a mi y a mis compañeros de tesis en el desarrollo completo del proyecto, sin su ayuda sin duda este logro no seria posible. En general agradezco a cada persona que de una u otra forma estuvo involucrada con nosotros para que pudiéramos conquistar nuestra meta.

NESTOR IVAN ZELAYA

INDICE

INTRODUCCIÓN

Durante los últimos años, las telecomunicaciones en nuestro país se han ido desarrollando grandemente, y la demanda de los servicios ha ido creciendo simultáneamente, es por eso que el desarrollo de nuevas tecnologías y la innovación de otras ya existentes ayudan a mejorar el servicio que se provee a los usuarios de dichas redes. La tecnología GSM ha sido en gran parte, base para el desarrollo de nuevos servicios ya que además de ser un estándar abierto para el desarrollo de aplicaciones ofrece todas las funcionalidades de dichas redes lo cual ha beneficiado no solo a las compañías sino también a los usuarios que hacen uso de estos servicios.

En el presente documento, se explica el desarrollo de un prototipo de una nueva aplicación basada en tecnología GSM, en donde, se describe totalmente su desarrollo, implementación y su funcionalidad, el documento consiste básicamente de un marco teórico que antecede el desarrollo de la aplicación, explicando como se desarrollan dichas aplicaciones, los requerimientos de los sistemas que se deben utilizar para poder llevar a cabo el desarrollo con éxito de ellas. Posteriormente se presenta el desarrollo específico de la aplicación, se muestran los pasos desde la programación de dispositivos, diseño de circuitos hasta la implementación de éstos, en conjunto para obtener la funcionalidad requerida. Seguido al desarrollo de la aplicación se presenta el manual de usuario para comprender las funciones que desempeña el sistema y poder configurarlas desde un dispositivo móvil. Finalmente se presenta a manera de justificación un capítulo que contiene un informe estadístico del problema de robo de vehículos en los últimos años en el país, lo cual da la pauta para el desarrollo de dicha aplicación.

Debido a la gran demanda de los usuarios de la red de telefonía celular, se hace inevitable el desarrollo de nuevas aplicaciones que puedan contribuir a satisfacer sus necesidades, trayendo provecho tanto a los usuarios finales como a las compañías que las proveen, y así poder llegar a satisfacer dichas necesidades que cada vez con el paso del tiempo se van haciendo mas grandes.

OBJETIVOS

OBJETIVO GENERAL

Desarrollar como trabajo de graduación un sistema basado en tecnología GSM para controlar el bloqueo y desbloqueo de dispositivos aplicados a un automóvil, aprovechando de esta manera, todas las ventajas que esta tecnología ofrece.

OBJETIVOS ESPECIFICOS

- Implementar un sistema que pueda bloquear y desbloquear el funcionamiento del automóvil.
- Que el sistema sea capaz de notificar al usuario el estado de los dispositivos instalados en el automóvil.
- Desarrollar aplicaciones en lenguaje Java para dispositivos móviles.

ALCANCES

- Se lograría que por medio del móvil se pueda bloquear y desbloquear la funcionalidad del auto dentro del rango de cobertura de la red celular.
- Se tendrá conocimiento del estado en que se encuentra el sistema (si el sistema esta activado o desactivado) por medio de una bandera o menú en el terminal móvil.
- El sistema debido a que se implementara con tecnología GSM es bastante versátil y abierto por lo que puede ser aplicado en diversas áreas a los cuales es necesario la activación y desactivación de dispositivos como también el conocer el estado de algunas variables dentro de un proceso.
- En cuestión de privacidad, el usuario tendrá un código único con el cual se identificara en el sistema, esto hará posible que el usuario haga uso del sistema desde cualquier móvil de manera segura.
- Dado que los automóviles ya poseen sensores en diversos puntos aprovecharemos éstos para nuestra aplicación. Utilizaremos como variables los sensores de las puertas y el voltaje de doce voltios conmutado que presenta el sistema eléctrico. Y si en caso el automóvil posea un sistema antirrobo se emplearan los sensores que éste provee.
- El propósito del proyecto no es el implementar un servicio, por lo cual no se toman en cuenta criterios relacionados a convenios que deben hacerse entre operadoras de telefonía celular y usuarios.

LIMITACIONES

- El sistema está restringido al alcance de la cobertura que tenga la red celular en el que el móvil esté registrado.
- Se considera que existen formas en las que la transmisión y recepción de las señales no pueden llevarse a cabo, lo que conlleva al mal funcionamiento del sistema, como por ejemplo: desconectar la alimentación al sistema, desconectar el módem o la antena de éste, tener apagado el celular, que la red celular no este disponible y otros.
- Como el sistema lo que realiza es la activación y desactivación de dispositivos en el automóvil, siendo uno de ellos el que sea capaz de lograr el bloqueo y desbloqueo del funcionamiento del automóvil, esto no imposibilita que de cierta forma mecánica el automóvil siga en movimiento, algo que seria indeseado en alguna emergencia.
- Se podría obtener la posición geográfica en la que se encuentra el vehículo, pero las operadoras de red celular mantienen esta información fuera del alcance de los usuarios.
- El usuario podrá tener acceso desde cualquier móvil hacia el sistema siempre y cuando pertenezca a la misma red celular y que posea el software de control instalado en el mismo.

Capítulo 1: GSM

1. GSM

1.1 Descripción del Capítulo

El desarrollo y el éxito de GSM ha sido un ejemplo excepcional de la empresa internacional en acción. A continuación encontrará una breve historia de GSM, describiendo cómo surgió y cómo la asociación nació, ha crecido y se ha desarrollado. Esta historia es a partir de 1987 hasta la fecha. Posteriormente, se mostrará la arquitectura de la misma, explicando cada unos de sus principales etapas.

1.2 Historia de GSM [1]

1.2.1 En el principio (1980-82)

Mientras que el negocio llegaba a ser cada vez más internacional, el filo de la industria de las comunicaciones se centró en soluciones celulares exclusivamente locales, y ninguno era remotamente compatible con cualquiera de los otros existentes: NMT[1] 450 en los países nórdicos, TACS[2] en el Reino Unido, C-Netz en República Federal de Alemania, Radiocom 2000 en Francia y RTMI/RTMS en Italia. Estos sistemas permitían la comunicación localmente pero no permitían el poder realizar llamadas hacia otras operadoras de países vecinos debido a las diferentes tecnologías que utilizaban. Sin embargo, estaba claro que habría una demanda de extensión para una tecnología que facilitó comunicaciones móviles flexibles y confiables. Pero ésta, en sí mismo era una bomba de tiempo potencialmente mortal que amenazó la durabilidad de las redes celulares de primera generación; el problema era la capacidad o la carencia de ella. Pronto llegó a ser obvio que para los años 90 las redes análogas dispares se derrumbarían bajo la presión de la demanda.

[1] Nordic Mobile Telephone
[2] Telephone Advanced Cellular System

1.2.2 *Groupe Speciale Mobile*[1] *(1982-83)*

También llegó a estar claro a los vigilantes de la industria, que las soluciones localizadas al desarrollo de comunicaciones móviles no tuvieron sentido económico a largo plazo. Dado a los costos desalentadores que hacían frente a operadores y a fabricantes, era esencial poder explotar las economías a escala inherentes en la penetración de mercado global. El beneficio del mercado interior no justificaría simplemente programas sostenidos de la inversión.

En un amplio marco como lo es la industria de las comunicaciones, el CEPT[2] merece un lugar muy especial en la historia. En 1982, el CEPT abarcaron las administraciones de las operadoras de telefonía móvil de veintiséis países europeos. Digno, pero no intrínsecamente emocionante. Sin embargo, la respuesta decisiva de esta organización a un estudio francogermano que detallaba los problemas que hacían frente a la industria móvil de la comunicación no era nada visionario.

Vale la pena recordar que la mayoría de los miembros del CEPT eran monopolios de estados utilizados para considerar el interés nacional como su objetivo primordial. Sin embargo, en este caso su primer paso era reconocer que el futuro económico de la nueva industria confió en niveles hasta ahora sin precedentes de la cooperación Pan-Europea. Con este fin, el CEPT estableció el *Groupe Speciale Mobile*. Su objetivo era desarrollar la especificación para una red de comunicaciones móvil Pan-Europeo capaz de soportar a los posibles millones de suscriptores, para dar un giro a las comunicaciones móviles en los años venideros.

Nadie tenía duda que al establecer tal estándar sería necesario lidiar con los problemas: técnicos, económicos y logísticos. Sin embargo, estaba igualmente claro que las recompensas podrían ser satisfactorias. Pero sobretodo había una voluntad genuina para combinar recursos para alcanzar éxito a largo plazo.

[1] Conocido como GSM
[2] Conference European Post and Telecommunications

1.2.3 Visión Digital (1984-85)

Todo estaba muy bien, el mundo comercial que reconocía la exigencia de una solución Pan-Europea a las comunicaciones móviles, pero tal iniciativa también iba a requerir la ayuda de algunos políticos de peso pesado. Afortunadamente, la promoción de CEPT de un sistema unificado tuvo sentido, y en 1984 el proyecto de GSM recibió el endoso de la Comisión de las Comunidades Europeas. Esto tendría implicaciones considerables en los años venideros.

En 1985, el crecimiento del compromiso de solución de la crisis llegó a ser evidente cuando la República Federal de Alemania, Francia e Italia firmaron un acuerdo para el desarrollo de GSM. El Reino Unido agregó su nombre al acuerdo el año siguiente. Por este tiempo, el Groupe Speciale Mobile de CEPT podría discutir persuasivamente que los estándares que desarrollaban llevaran a cabo la llave a la solución técnica y económicamente viable.

Desde el comienzo, se tenía en mente que en el nuevo estándar era probable emplear tecnología digital en lugar de tecnología análoga y funcionar en la banda de frecuencia 900MHz. La tecnología digital ofreció una combinación atractiva, el funcionamiento y eficacia espectral. Es decir, proporcionaría la transmisión de alta calidad y permitiría a más usuarios utilizar simultáneamente la banda de radio limitada disponible. Además, tal sistema permitiría el desarrollo de características avanzadas como seguridad en las conversaciones y de comunicación de datos.

Yendo por el camino digital, también sería posible emplear una escala muy grande de tecnología integrada del silicio. Los microteléfonos podían ser más baratos y más pequeños. También permitirían introducir los primeros terminales hand-held, aun cuando en los comienzos en términos de tamaño y peso estos serían prácticamente indistinguibles a un ladrillo.

Finalmente, el acercamiento digital complementó cuidadosamente el Integrated Services Digital Network (ISDN) que eran desarrollados por los sistemas land-based de las telecomunicaciones a través del mundo y con qué GSM tendría que obrar recíprocamente.

1.2.4 El catalizador político (1986-87)

El tiempo había llegado para que GSM busque algo más substancial que el endoso de los políticos. La acción fue requerida, no menos porque los bloques de la frecuencia que se empleará por el nuevo estándar eran encajados a presión por las redes análogas. En conclusión, la solución del GSM estaba bajo amenaza incluso antes de emerger.

Afortunadamente, por 1986 la crisis inminente de la capacidad excesiva había comenzado a alarmar a la Comunidad Europea, la demanda comenzaba a observarse, incluso las proyecciones más optimistas. Y la defensa del Groupe Speciale Mobile de la tecnología celular digital estaba a la mano para ofrecer la luz en el extremo del túnel. Sin embargo, para que la visión se convierta en una realidad, los fabricantes tuvieron que ser convencidos de que el proyecto tenía un futuro.

La presión de países como Francia y República Federal de Alemania animó a la Comisión de las Comunidades Europeas que limitará la situación a los jefes de los Estados miembros en una reunión en diciembre de 1986. El resultado era una recomendación y un directorio que entre ellos puso las fundaciones políticas para el desarrollo de GSM.

La recomendación delineó una introducción coordinada de GSM que se apoyará por el EEC[1]. Se programó para un lanzamiento de un sistema limitado de servicios antes de 1991. La directiva se aseguró de que cada Estado miembro reservara los bloques de la frecuencia 900MHz requeridos para el programa de

[1] EEC: End-Entity Certificate

expansión. Aunque éstos eran algo más pequeños que la cantidad abogada por el CEPT, la industria finalmente había alcanzado la ayuda política que necesitó para avanzar con sus objetivos.

1.2.5 El memorandum de la comprensión[1] (1986-87)

La generación tanto de la actividad técnica y política tenía el potencial de dar vuelta en una pesadilla logística. Llegó a estar claro que la situación exigió que haya una organización sola, permanente en el timón. Esta necesidad fue resuelta cuando GSM asumió la responsabilidad total de coordinar el desarrollo de la especificación completa de GSM. En 1986 el núcleo permanente de GSM fue formado y sus jefaturas fueron establecidas en París.

Estaba todo muy bien, conviniendo la tecnología y los estándares para este producto nuevo. ¿Pero qué sobre la creación de un mercado? Era esencial forjar un acuerdo comercial entre los operadores potenciales que se confiarían a poner el estándar en ejecución para una fecha particular. Sin tal acuerdo no podría haber red. Sin la red no habría terminales. Sin red y terminales no habría servicio. El templo de Stephen del Ministerio de Comercio del Reino Unido y la industria fue cargado con la tarea de bosquejar el primer Memorándum de Comprensión. Le convencieron de que si el proyecto fuera a tener un futuro por lo menos tres países tendrían que confiar para desplegar tecnología GSM antes del 1 de julio de 1991. En el acontecimiento, él no necesitó preocuparse. Tal era la potencia de la visión de GSM que el 7 de septiembre de 1987 trece países firmaron el MoU en Copenhague. Habían 15 firmas en total: Francia, Alemania, Italia, Suecia, Noruega, Dinamarca, Finlandia, España, los Países Bajos, Bélgica, Portugal, Irlanda y, del Reino Unido, dos operadores independientes: Cellnet y Racal-Vodafone, así como los servicios de DTI[2], para establecer su identidad de mercado y para darles un borde en un ambiente cada vez más competitivo.

[1] MoU: Memorandum of Understanding
[2] DTI: Digital Tecnology International

1.2.6 La visión se convierte en una realidad (1987-88)

Mientras que el EEC establecía una fundación política para una red Pan-Europea, en el frente técnico el Groupe Speciale Mobile se había estado determinando una gama de ofertas que determinarían la forma de los servicios del móvil de la segunda generación.

En 1986, el núcleo permanente de GSM había llevado a cabo una serie de ensayos de la validación en París. Probaron ocho o nueve diversos diseños en la búsqueda para una trayectoria de radio apropiada, porque en el corazón de convertirse un nuevo estándar digital era la solución de preguntas, referente a la corrección de la confiabilidad y de error. Afortunadamente, la tecnología que podría proporcionar tal solución estaba justo a la vuelta de la esquina.

Una de las conclusiones más importantes dibujadas de las pruebas tempranas era que el nuevo estándar debe emplear tecnología de múltiple acceso por división de tiempo (TDMA). La fuerza de su funcionamiento técnico se aseguró de que la banda estrecha TDMA tuviera la ayuda importantes fabricantes como Nokia, Ericsson y Siemens. Esto prometió la flexibilidad inherente en tener el acceso a una amplia gama de proveedores y el potencial de conseguir el producto más rápido en el mercado.

En el curso de 1987, el núcleo permanente de GSM supervisó el desarrollo de una plataforma única de prueba de GSM. Hasta el momento un gran programa de tiempo, dinero y el esfuerzo habían sido tomados en cuenta en el proyecto. Ahora hicieron frente a los pioneros de la industria con la gran pregunta. ¿Trabajaría?

Después de una secuencia de pruebas de validación, la respuesta era resonante: sí. El estándar digital del GSM trabajó, no obstante con algunas modificaciones leves a la especificación propuesta. La etapa ahora fue fijada. En febrero de 1988, todos los trece operadores del MoU publicaron la invitación simultáneamente.

1.2.7 Todos los sistemas despejados (1990-91)

Si el 1 de julio de 1991, la fecha del lanzamiento no fuera resuelta había un peligro verdadero que la confianza en la tecnología GSM fuera fatalmente minada. Fue convenido que la evolución de la especificación estaría dividida en dos fases mutuamente compatibles. Además, había sido considerado siempre que el despliegue de la red sería terminado en etapas. El plan era que la cobertura de ciudades, capitales europeas y sus aeropuertos estarían disponible en el lanzamiento. Esto sería seguido por las autopistas y ampliado gradualmente para abarcar durante años territorios restantes.

Afortunadamente, el proyecto dio una alza significativa cuando en 1989 la responsabilidad del desarrollo de la especificación pasó del núcleo permanente de GSM al Instituto Europeo de los Estándares de las Telecomunicaciones (ETSI) nuevamente creado.

ETSI acordó estados iguales a los administradores, a los operadores y a los fabricantes, que en sí mismo tenían un impacto considerable en la velocidad del desarrollo. Era la combinación de un ambiente cooperativo y de los recursos mejorados que permitieron a la mayoría de la fase 1 de las especificaciones del GSM 900 ser publicados en 1990.

Además, el PCN[1] del Reino Unido resultó ser más una oportunidad que una amenaza. Los nuevos operadores decididos a utilizar la especificación de GSM, modificada levemente debido a la frecuencia más alta, y el desarrollo de qué se sabía mientras que DCS[2] 1800 fue realizado por ETSI en paralelo a la estandardización de GSM. De hecho, en 1997 DCS 1800 fue retitulado GSM 1800 para reflejar la afinidad entre las dos tecnologías.

[1] Personal Communication Network
[2] Digital Cellular System

A pesar de los progresos, los ánimos y los esfuerzos técnicos y de los administradores, el plazo para los servicios comerciales se derrumbó bajo el peso de la presión técnica. Sin embargo, en el lado positivo, una red piloto de GSM fue exhibida con éxito en la exposición del Telecom 91 de la ITU[1] en Ginebra con alrededor 11.000 llamadas hechas y ningún problema importante encontrados.

1.2.8 Barreras a superar (1991-92)

Para un número de operadores europeos como la red D2 alemana, la inhabilidad de lanzar un servicio comercial en 1991 frustraba. Las redes mismas eran completamente operacionales. El problema era que no había terminales GSM disponibles.

La piedra angular de GSM es el *Roaming*[2] internacional. Para que esto sea posible todas las redes y microteléfonos tienen que ser idénticos. Con tantos fabricantes creando tantos productos en tantos países, era crítico que cada tipo de terminal estuviese sujeta a un régimen riguroso de aprobación. Los terminales Rogue podían causar daño a las nuevas redes.

La crisis que enfrentaban las operadoras GSM en 1991 era que el proceso de aprobación no estaba disponible. En retrospectiva, es fácil entender la razón del retraso. Tal prueba requirió la creación de una pieza compleja de equipo que simuló el sistema entero GSM en el cual el terminal tendría que funcionar. El desarrollo del software del simulador resultó ser considerablemente más complejo de lo que había sido considerado. Pero sin un proceso de prueba no podría haber terminales y sin los terminales ninguno de los operadores podría firmar con los clientes. La situación estaba cerca de ser crítica.

[1] International Telecommunications Union
[2] Capacidad que poseen los teléfonos móviles de registrarse a una red distinta de la que proviene

La solución fue la introducción de un tipo de aprobación interina (ITA[1]). Esencialmente, éste era un procedimiento en el cual solamente un subconjunto de los parámetros de la aprobación fue probado para asegurarse de que el terminal en la pregunta no crearía ningún problema para las redes. A pesar de la preocupación considerable expresada por algunos operadores, los terminales ITA llegaron a estar extensamente disponibles en el curso de 1992. Los terminales verdaderos handheld golpearon el mercado en el final de ese año y GSM finalmente había comenzado a sonar.

1.2.9 Empuje de un millón de clientes (1992-93)

Una característica notable de la revolución de GSM era que una vez comenzado, pronto llegó a ser imparable. Después que el método de prueba ITA había sido convenido en abril de 1992, la disponibilidad de aumento de terminales estimuló la aparición de los primeros servicios de red comerciales genuinos. Sin embargo, debe ser recordado que éstos fueron obligados dentro de las limitaciones de las especificaciones de la fase 1.

En la práctica, entonces, el lanzamiento verdadero de GSM ocurrió en la última parte de 1992. Entre los corredores tempranos estaban Dinamarca (dos operadores), Finlandia (dos operadores), Francia, Alemania (dos operadores), Italia, Portugal (dos operadores) y Suecia (tres operadores). Entonces el 17 de de junio 1992 el primer acuerdo de *Roaming* fue firmado entre Finlandia Telecom y Vodafone en el Reino Unido.

En paralelo a la llegada de las redes, los profesionales de la industria tomaban la oportunidad de hacer su propio establecimiento de red la cuál debía pronto convertirse en el acontecimiento global principal de GSM. En 1987, el primer cuál debía convertirse en un acontecimiento anual dedicado a la promoción mundial de la tecnología GSM fue efectuado por servicios técnicos de los organizadores IBC de la conferencia. La conferencia celular Pan-europea Digital

[1] Interim Type Approval

fue llevada a cabo en Londres y atrajo a 150 delegados. El acontecimiento se ha efectuado en Berlín, Atenas, Lisboa y, en 1995, fue relanzado como el congreso mundial de GSM en Madrid.

Antes del fin de 1993, GSM había incursionado a través de 1 millón de suscriptores con el millón siguiente ya en el horizonte. El MoU podría ahora jactarse de 70 miembros a partir de 48 países y 25 acuerdos de *Roaming*, ya habían sido firmados, sellados y entregados.

Y, quizás el más significativo de todos, la compañía australiana Telstra había agregado su nombre en calidad de miembro cada vez mayor del MoU. Parecía que GSM no iba a seguir siendo un fenómeno exclusivamente europeo.

1.2.10 Crecimiento y crecimiento (1993-94)

Una de las características definitorias de cualquier revolución es que, no obstante se planean cuidadosamente, nadie puede realmente estar seguro a donde conducirán. Una característica inesperada de la nueva industria móvil era que llevó las semillas de la liberalización del mercado entero de las telecomunicaciones. Irónico, detrás de 1987 solamente dos de los signatarios originales de GSM MoU no eran monopolios controlados por el estado. Hoy, la mayoría de los miembros son operadores independientes.

La competición ha desempeñado un papel importante en el éxito de GSM. Los nuevos operadores trajeron habilidades frescas de la comercialización y un acercamiento más comercial la cuál había sido hasta ahora algo desconocido. Los titulares tuvieron que moverse rápidamente para que siguieran siendo competitivos. Con algunos países teniendo una opción de dos o aún tres operadores de GSM, las tarifas fueron forzadas a bajar y el consumidor gozó de estándares de calidad y de servicio al cliente.

En septiembre de 1993, Mercury One-2-One lanzó la primera red 1800 DCS en el Reino Unido. En los años siguientes, licenciando las administraciones a través del mundo emplearía este sistema, que utilizó especificaciones modificadas de GSM, como el medio para introducir la competición adicional en el mercado móvil.

Entonces en 1994, la Comisión Federal de las Comunicaciones[1] de los E.E.U.U. designó grandes bloques del espectro en la banda de 1900MHz. El punto era introducir redes inalámbricas digitales al país en forma de una nueva clase de servicio personal de las comunicaciones del mercado total. En el espíritu competitivo de los tiempos, la FCC se aseguró deliberadamente de que las licencias de las PCS fueran neutrales en lo que concierne al tipo de tecnología que se empleará. Para GSM, uno de los mercados más lucrativos del mundo hizo señas.

1.2.11 Globalización (1995-96)

GSM era un sistema previsto para asegurarse de que europeos podrían comunicarse en su continente (Roaming) y utilizar su teléfono donde quiera que viajaran.

Entonces a la larga vinieron los australianos. Cuando en 1992 Telstra firmó con el MoU, era la primera indicación que la energía y la flexibilidad de GSM tenían potencial verdadero en el mercado global. En apenas cuatro años, la red GSM de la compañía tenía más de 1 millón de suscriptores.

La tendencia mundial hacia la desregulación y la liberalización de telecomunicaciones significó que un anfitrión de nuevos jugadores se incorporaba, el mercado móvil. La especificación GSM era una característica rica y ofreció una gama de servicios que permitirían a nuevos operadores distinguirse de sus rivales. Y en el corazón de la tendencia hacia GSM la ventaja era su capacidad

[1] FCC: Federal Communication Commision

para comunicarse globalmente. En resumen, el mundo estaba listo para GSM. Solamente algunos años después de que Telstra había firmado con GSM se habían ampliado más allá de Europa y de Australia, estableciendo una presencia en las áreas tan diversas como la India, África, Asia y el mundo árabe.

Antes de junio de 1995, el MoU fue colocado formalmente como asociación en Suiza. Ahora tenía 156 miembros sirviendo a 12 millones de clientes en 86 países. El mismo año se consideró la terminación de la estandardización de la fase 2 de GSM y de una demostración de la comunicación del fax, vídeo y de datos vía GSM. También produjo una adaptación de PCS 1900 para resolver las oportunidades creadas por la subasta reciente de la FCC en los E.E.U.U..

En los Estados Unidos, los nuevos operadores PCS[1] reconocieron las ventajas de un estándar abierto que creaba un mercado global y monopólico para el producto. Esto tenía la ventaja de hacer el despliegue de la red extremadamente rentable. Una vez que la FCC hubiera abierto la puerta, los vendedores principales de GSM desarrollaron rápidamente una variación de GSM modificada para requisitos particulares de la banda de frecuencia 1900MHz.
En noviembre de 1995, las comunicaciones personales americanas lanzaron el primer servicio comercial de GSM en los E.E.U.U. Esto fue alcanzado apenas algunos meses después de que el APC hubiera obtenido su licencia, una hazaña que habría sido imposible con cualquier otra tecnología. Antes de mayo de 1997, había 15 redes de las PCS 1900, ahora GSM 1900, y con alrededor de 400.000 usuarios. Con los suscriptores viniendo abordo en un índice de 2.000 al día, GSM ha puesto ya una huella indeleble en el mercado norteamericano.

1.3 Fases de GSM [2]

A finales de los 80, los grupos involucrados en el desarrollo del estándar GSM se dieron cuenta que en el tiempo estimado no podrían completar las especificaciones

[1] Personal Cellular System

para el rango entero de los servicios y características de GSM que fueron originalmente planeadas. Debido a esto, se decidió que GSM seria lanzado en fases, con la fase 1 consistiendo en un grupo de servicios y características limitadas. Cada nueva fase encaja en los nuevos servicios ofrecidos por fases ya existentes.

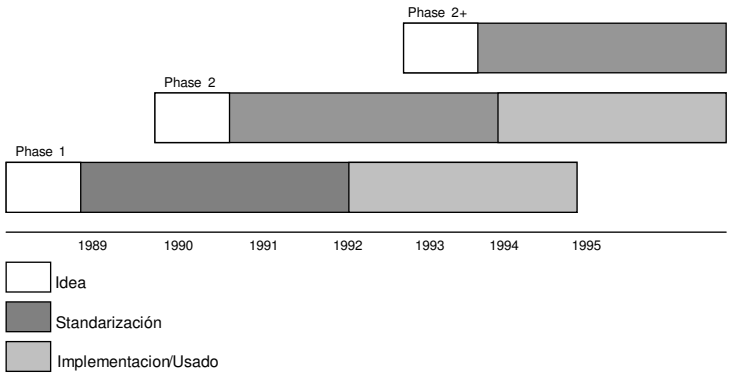


Figura 1-1 fases GSM

1.3.1 Fase 1

La fase 1 contiene los servicios más comunes incluyendo:

- Telefonía de voz
- Roaming Internacional
- Servicios básicos de fax y datos (a mas de 9.6kbits/s)
- Establecimiento de llamada
- SMS (*Short Message Service*)
- Bloqueo de llamadas

La fase 1 también incorporó características como cifrado y tarjetas de Módulos de Identidad de Suscriptor (SIM). Las especificaciones de la fase 1 entonces fueron cerradas y no pueden ser modificados.

1.3.2 Fase 2

Otras características adicionales fueron introducidas en la fase 2 de GSM incluyendo:

- Aviso de carga

- Identificación de Línea de Llamada
- Llamada en espera
- Retención de llamada
- Llamada en conferencia
- Grupos de usuario cerrados
- Capacidad de comunicaciones de datos adicionales

1.3.3 Fase 2+

La estandarización de grupos ya había comenzado a definir la siguiente fase, 2+. El programa de la fase 2+ cubrirá múltiples números de suscriptores y una variedad de características orientadas a los negocios.

Algunas de las características ofrecidas por la fase 2+ son:

- Perfiles de servicios múltiples
- Planes de numeración privados
- Interrelación con los estándares GSM 1800, GSM 1900 y DECT[1]

Las prioridades e itinerarios para nuevas características y funciones dependen primordialmente en el interés mostrado por compañías operadoras, fabricantes y desarrollos técnicos en áreas relacionadas.

1.4 Componentes de Red GSM [2]

La red GSM esta dividida en dos sistemas. Cada uno de estos sistemas comprende un número de unidades funcionales las cuales son componentes individuales de la red móvil. Los dos sistemas son:

- Sistema de Conmutación (*SS, Switching System*)
- Sistema de Estación Base (*BSS, Base Station System*)

[1] Digital Enhanced Cordless Telecommunications

Como todas las redes de telecomunicación, las redes GSM son operadas, mantenidas y administradas desde centros computarizados.

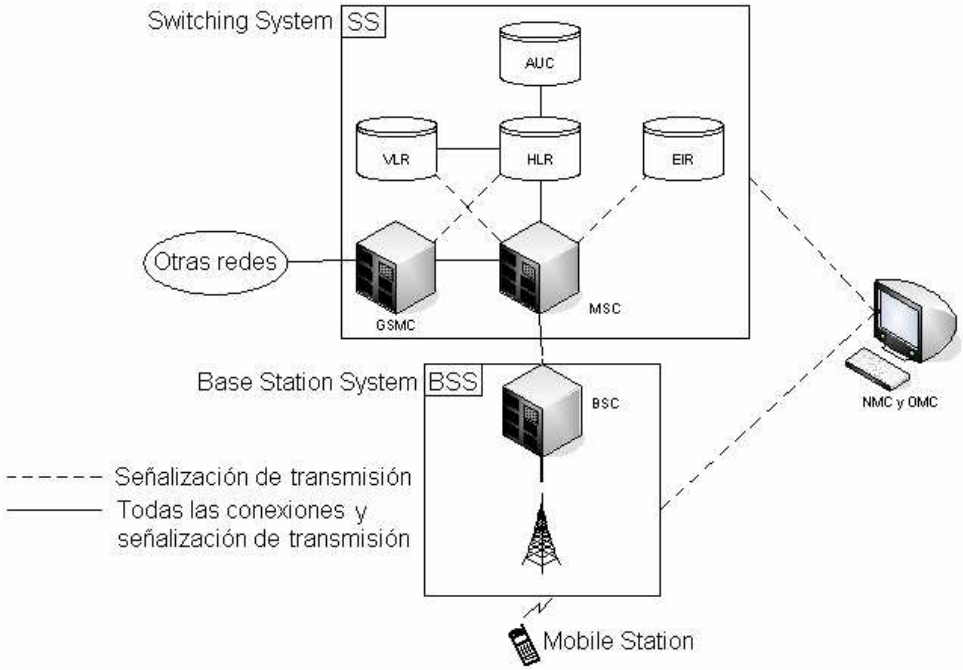


Figure 1-2 (Modelo de sistema)

El sistema de conmutación (SS) es responsable de la realización del procesamiento de la llamada y de las funciones relacionadas al suscriptor. Esto incluye las siguientes unidades funcionales:

- *Mobile services Swithching Center (MSC)*
- *Home Location Register (HLR)*
- *Visitor Location Register (VLR)*
- *AUthentication Center (AUC)*
- *Equipment Identity Register (EIR)*

La BSS ejecuta todas las funciones de radio relacionadas. La BSS comprende las siguientes unidades funcionales:

- *Base Station Controller (BSC)*
- *Base Transceiver Station (BTS)*

El OMC (*Operation and Maintenance Center*) ejecuta todas las operaciones de mantenimiento para la red tales como: monitoreo del tráfico de la red y las alarmas de la red. El OMC posee acceso tanto al SS como al BSS.
Los MS (*Mobile Stations*) no pertenecen a ninguno de estos sistemas

1.4.1 Componentes del SS (*Switching System*)

1.4.1.1 MSC (Mobile services Switching Center)

El MSC ejecuta las funciones de conmutación telefónica para la red móvil. Controla llamadas para y desde otros sistemas de datos y telefonía, como la PSTN (Public Switched Telephony Network), ISDN (Integrated Services Digital Network) redes de datos públicas, redes privadas y otras redes móviles.

1.4.1.2 Funcionalidad del Gateway

La funcionalidad del *Gateway* permite a un MSC preguntar al HLR de la red para poder enrutar una llamada a una estación móvil. El MSC es llamado *Gateway* MSC (GMSC) Por ejemplo. Si una persona conectada a una red publica quiere hacer una llamada a un suscriptor de la red móvil GSM, entonces el Exchange de la red pública accederá a la red GSM conectando la llamada al GMSC primeramente. Lo mismo para una llamada desde un MS a otro MS.

1.4.1.3 HLR (Home Location Register)

El HLR es una base de datos de red centralizada que almacena y administra todas las suscripciones móviles pertenecientes a una operadora específica. Actúa como un almacenamiento permanente de la inscripción de suscripción de una persona hasta que la suscripción es cancelada. La información almacenada incluye:

- Identidad del suscriptor
- Servicios suplementarios del suscriptor
- Información de localidad de suscriptor

- Información de autenticación de suscriptor

El HLR puede ser implementado en el mismo nodo de red como el MSC o como una base de datos única. Si la capacidad de un HLR es excedida por un número de suscriptores, pueden agregarse HLR adicionales.

1.4.1.4 VLR (Visitor Location Register)

La base de datos VLR contiene la información acerca de los suscriptores móviles localizados en un área de servicio MSC. Hay un VLR para cada MSC en una red. El VLR almacena temporalmente la información de suscripción para que el MSC pueda servir a todos los suscriptores visitando el área de servicio MSC. El VLR puede ser considerado como un HLR distribuido mientras mantiene una copia de la información del HLR almacenada del suscriptor.

Cuando el suscriptor se conecta a una nueva área de servicio MSC, el VLR conectado a es MSC pide información del suscriptor del HLR del suscriptor. El HLR manda una copia de la información al VLR y actualiza su propia información de localidad. Cuando un suscriptor hace una llamada, el VLR ya tendrá la información requerida para configurar la llamada.

1.4.1.5 AUC (AUthentication Center)

La función principal del AUC es de autenticar los intentos de uso de la red de los suscriptores. De esta forma, es utilizado para proteger los operadores de la red contra el fraude. El AUC es una base de datos conectada al HLR el cual lo provee con los parámetros de autenticación y llaves de cifrado utilizados para garantizar la seguridad de la red.

1.4.1.6 EIR (Equipment Identity Register)

El EIR es una base de datos que contiene información de identidad del equipo móvil, el cual ayuda a bloquear llamadas de MS's robados, no

autorizados o defectuosos. Debe notarse que debido a la separación entre suscriptor y equipo en GSM. El bloqueo de equipo de estaciones móviles no resulta en un bloqueo automático de un suscriptor.

1.4.2 Componentes de una BSS (Base Station System)

1.4.2.1 BSC (Base Station Controller)

El BSC administra las funciones de radio relacionadas a una red GSM. Es un conmutador de alta capacidad que provee funciones como *Handover*[1] de estación móvil, asignación de canales de radio y la recolección de datos de configuración de la celda. Un número de BSC puede ser controlada por cada MSC.

1.4.2.2 Base Transceiver Station (BTS)

La BTS controla la interface de radio para el MS. EL BTS comprende el equipo de radio como *trasnceivers* y antenas las cuales son necesitadas en cada celda en el red. Un grupo de BTS son controladas por un BSC.

1.4.3 Centros de Monitoreo de Red

1.4.3.1OMC (Operation and Maintenance Center)

Un OMC un centro de monitoreo computarizado el cual esta conectado a otros componentes de la red como MSC y BSC a través de enlaces de red de datos X.25. En el OMC se puede saber información del estado de la red y se puede monitorear y controlar una variedad de parámetros de sistema. Puede haber uno o varios OMC en una red dependiendo del tamaño de esta.

1.4.3.2 NMC (Network Managment Center)

El control centralizado de una red es realizado en un centro de administración de la red llamado NMC. Solamente un NMC es requerido para una red, y esta

[1] Handover es el cambio de una llamada continua a un canal o celda diferente.

controla la OMC subordinada. Las ventajas de esta jerarquía es que el personal del NMC puede concentrarse en problemas de larga duración en el sistema, mientras que el personal de cada OMC puede concentrarse en problemas pequeños.

La funcionalidad del OMC y NMC puede ser combinada en el mismo nodo físico de la red o puede ser implementado en diferentes locaciones.

1.4.4 MS (Mobile Stations)

Un MS es utilizado por un suscriptor móvil para comunicarse con la red móvil. Existen diferentes tipos de MS cada uno permite al suscriptor hacer y recibir llamadas. Fabricantes de MS ofrecen una variedad de diseños y características para cumplir con las necesidades de los diferentes mercados.

El rango del área de cobertura de un MS depende de la potencia de salida del MS. Diferentes tipos de MS tienen diferentes tipos de capacidad de potencia de salida y consecuentemente diferentes rangos. Por ejemplo los MS de mano posee una menor potencia de salida y rango que un MS instalada en un automóvil con montaje de antena en el techo.

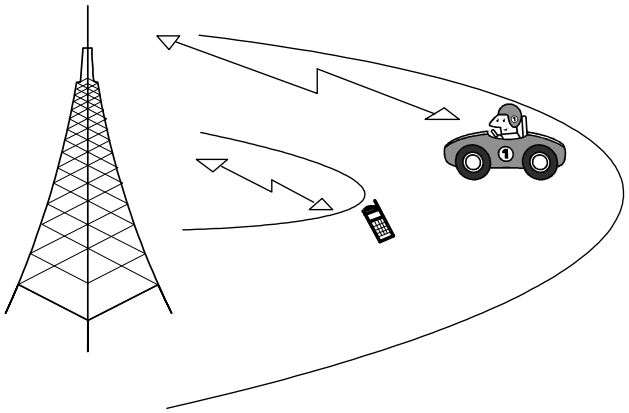


Figure 1-3 Rangos para diferentes tipos de MS

Un MS GSM consiste en:

Un Terminal móvil

Un SIM (Subscriber Identity Module)

No como otros estándares, en GSM el suscriptor esta separado del Terminal móvil. Cada información de suscriptor es almacenada como una tarjeta inteligente (SIM). El SIM puede ser conectado en cualquier Terminal móvil GSM. Esto conlleva a ventajas de seguridad y portabilidad para los suscriptores. Por ejemplo, el Terminal móvil del suscriptor A puede ser robado, sin embargo el SIM del terminal móvil del suscriptor A puede ser usado en otro terminal móvil de otra persona y las llamadas serán cargadas al suscriptor A.

Capítulo 2: INTRODUCCION A J2ME

2. INTRODUCCIÓN A J2ME [3]

2.1 Descripción del capítulo

La empresa Sun Microsystems lanzó a mediados de los años 90 el lenguaje de programación Java que, aunque en un principio fue diseñado para generar aplicaciones que controlaran electrodomésticos como lavadoras, congeladoras, etc., debido a su gran robustez e independencia de la plataforma donde se ejecutase el código, desde sus comienzos se utilizó para la creación de componentes interactivos integrados en páginas Web y programación de aplicaciones independientes. Estos componentes se denominaron applets y casi todo el trabajo de los programadores se dedicó al desarrollo de éstos. Con los años, Java ha progresado enormemente en varios ámbitos como servicios HTTP, servidores de aplicaciones, acceso a bases de datos (JDBC[1])... Como se ve, Java se ha ido adaptando a las necesidades tanto de los usuarios como de las empresas ofreciendo soluciones y servicios tanto a unos como a otros. Debido a la explosión tecnológica de estos últimos años Java ha desarrollado soluciones personalizadas para cada ámbito tecnológico. Sun ha agrupado cada uno de esos ámbitos en una edición distinta de su lenguaje Java. Estas ediciones son Java 2 Standard Edition, orientada al desarrollo de aplicaciones independientes y de applets, Java 2 Enterprise Edition, enfocada al entorno empresarial y Java 2 Micro Edition, orientada a la programación de aplicaciones para pequeños dispositivos. En esta última edición de Java es en la que se centrará todo el estudio en el presente capitulo.

La edición Java 2 Micro Edition fue presentada en 1999 por Sun Microsystems con el propósito de habilitar aplicaciones Java para pequeños dispositivos. En esta presentación, lo que realmente se enseñó fue una primera versión de una nueva Java Virtual Machine (JVM) que podía ejecutarse en dispositivos Palm. Para empezar se puede decir que Java Micro Edition es la versión del lenguaje Java que está orientada

[1] JDBC: Java Data Base Connectivity

al desarrollo de aplicaciones para dispositivos pequeños con capacidades restringidas tanto en pantalla gráfica, como de procesamiento y memoria (teléfonos móviles, PDAs[1], Handhelds, Pagers, etc.). La tardía aparición de esta tecnología, (se ha visto que la tecnología Java nació a mediados de los 90 y Java Micro Edition apareció a finales), puede ser debido a que las necesidades de los usuarios de telefonía móvil ha cambiado mucho en estos últimos años y cada vez demandan más servicios y prestaciones por parte tanto de los terminales como de las compañías. Además el uso de esta tecnología depende del asentamiento en el mercado de otras, como GPRS, íntimamente asociada a J2ME y que no ha estado al alcance hasta hace poco. J2ME es la tecnología del futuro para la industria de los dispositivos móviles. Actualmente las compañías telefónicas y los fabricantes de móviles están implantando los protocolos y dispositivos necesarios para soportarla.

2.2 Análisis comparativo

Antes de comenzar con los conceptos necesarios de Java Micro Edition es necesario hacer un breve estudio de las distintas versiones de Java, los cuales ya fueron mencionados en el apartado anterior.

Sun, dispuesto a proporcionar las herramientas necesarias para cubrir las necesidades de todos los usuarios, creó distintas versiones de Java de acuerdo a las necesidades de cada uno. Según esto, el paquete Java 2 se puede dividir en 3 ediciones distintas. J2SE (Java Standard Edition) orientada al desarrollo de aplicaciones independientes de la plataforma, J2EE (Java Enterprise Edition) orientada al entorno empresarial y J2ME (Java Micro Edition) orientada a dispositivos con capacidades restringidas. Las características de cada una de las versiones son:

1. **Java 2 Platform, Standard Edition (J2SE)**: Esta edición de Java es la que en cierta forma recoge la iniciativa original del lenguaje Java. Tiene las siguientes características:

[1] PDA: Personal Digital Assistant

- Inspirado inicialmente en C++, pero con componentes de alto nivel, como soporte nativo de strings y recolector de basura.
- Código independiente de la plataforma, precompilado a bytecodes intermedio y ejecutado en el cliente por una JVM (Java Virtual Machine).
- Modelo de seguridad tipo *sandbox* proporcionado por la JVM.
- Abstracción del sistema operativo subyacente mediante un juego completo de APIs de programación.

Esta versión de Java contiene el conjunto básico de herramientas usadas para desarrollar Java Applets, así cómo las APIs orientadas a la programación de aplicaciones de usuario final: Interfaz gráfica de usuario, multimedia, redes de comunicación, etc.

2. **Java 2 Platform, Enterprise Edition (J2EE)**: Esta versión está orientada al entorno empresarial. El software empresarial tiene unas características propias marcadas: está pensado no para ser ejecutado en un equipo, sino para ejecutarse sobre una red de ordenadores de manera distribuida y remota mediante Ejes (Enterprise Java Beans). De hecho, el sistema se monta sobre varias unidades o aplicaciones. En muchos casos, además, el software empresarial requiere que se sea capaz de integrar datos provenientes de entornos heterogéneos. Esta edición está orientada especialmente al desarrollo de servicios web, servicios de nombres, persistencia de objetos, XML, autenticación, APIs para la gestión de transacciones, etc. La tarea de esta especificación es ampliar la J2SE para dar soporte a los requisitos de las aplicaciones de empresa.
3. **Java 2 Platform, Micro Edition (J2ME)**: 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 o electrodomésticos inteligentes. Esta edición tiene unos componentes básicos que la diferencian de las 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 vez del uso de la JVM clásica, inclusión

de un pequeño y rápido recolector de basura y otras diferencias que se verán más adelante.

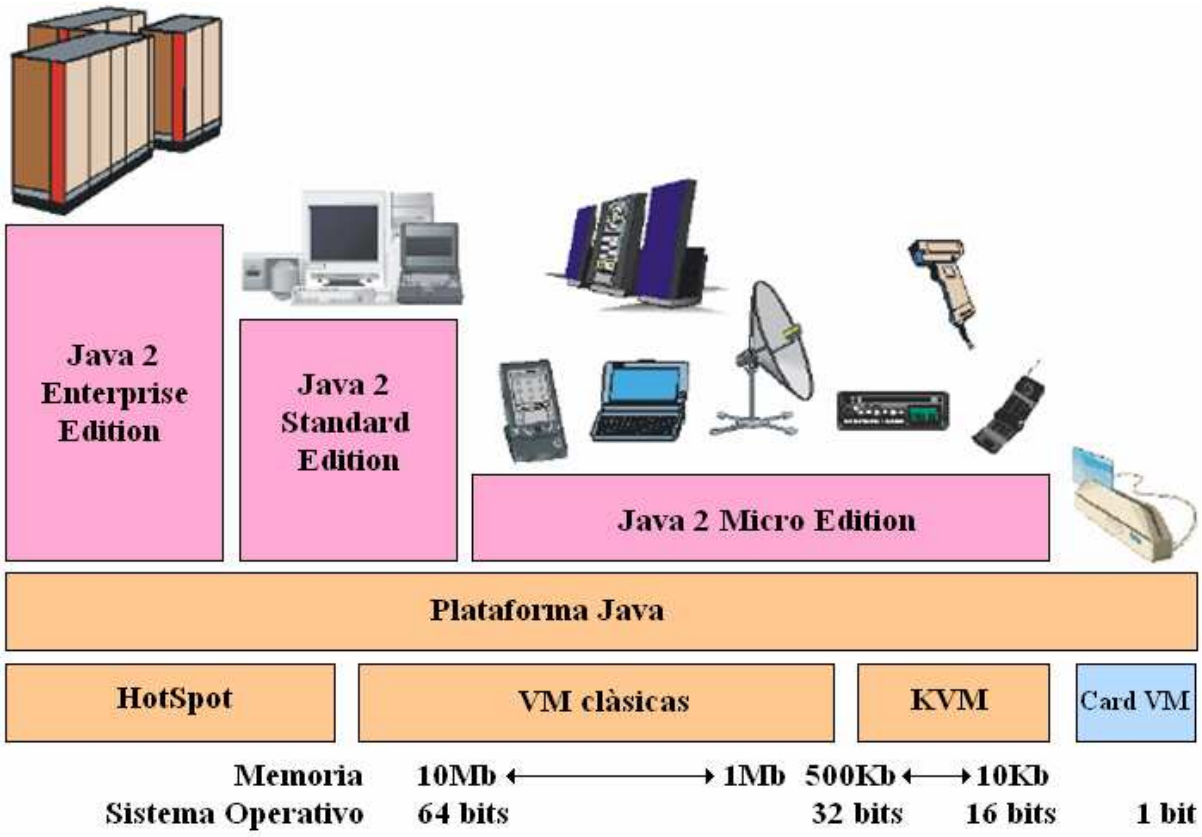


Figura 2.1 Arquitectura de la plataforma Java 2 de Sun

La Figura 2.1 muestra la arquitectura de la plataforma Java 2.

En la actualidad no es realista ver Java como un simple lenguaje de programación, si no como un conjunto de tecnologías que abarca a todos los ámbitos de la computación con dos elementos en común:

- El código fuente en lenguaje Java es compilado a código intermedio interpretado por una Java Virtual Machine (JVM), por lo que el código ya compilado es independiente de la plataforma.

- Todas las tecnologías comparten un conjunto más o menos amplio de APIs básicas del lenguaje, agrupadas principalmente en los paquetes java.lang y java.io.

Un claro ejemplo de éste último punto es que J2ME contiene una mínima parte de las APIs de Java. Esto es debido a que la edición estándar de APIs de Java ocupa 20Mb, y los dispositivos pequeños disponen de una cantidad de memoria mucho más reducida. En concreto, J2ME usa 37 clases de la plataforma J2SE provenientes de los paquetes java.lang, java.io, java.util. Esta parte de la API que se mantiene fija forma parte de lo que se denomina "configuración", que se hablará más extensamente en el siguiente capitulo. Otras diferencias con la plataforma J2SE vienen dadas por el uso de una máquina virtual distinta de la clásica JVM denominada KVM. Esta KVM tiene unas restricciones que hacen que no posea todas las capacidades incluidas en la JVM. Estas diferencias se verán más detenidamente cuándo se analice las capacidades de la KVM en el siguiente capitulo.

Como se ve, J2ME representa una versión simplificada de J2SE. Sun separó estas dos versiones ya que J2ME estaba pensada para dispositivos con limitaciones de proceso y capacidad gráfica. También separó J2SE de J2EE porque este último exigía unas características muy pesadas o especializadas de E/S, trabajo en red, etc. Por tanto, separó ambos productos por razones de eficiencia. Hoy, J2EE es un súper conjunto de J2SE pues contiene toda la funcionalidad de éste y más características, así como J2ME es un subconjunto de J2SE (excepto por el paquete javax.microedition) ya que, como se mencionó, contiene varias limitaciones con respecto a J2SE.

Figura 2.2 Relación entre las APIs de la plataforma Java.

Sólo de manera muy simplista se puede considerar a J2ME y J2EE como versiones reducidas y ampliadas de J2SE respectivamente (ver Figura 2.2): en realidad cada una de las ediciones está enfocada a ámbitos de aplicación muy distintos. Las necesidades

computacionales y APIs de programación requeridas para un juego ejecutándose en un móvil difieren bastante con las de un servidor distribuido de aplicaciones basado en EJB.

2.3 Nociones básicas de J2ME

Ya se ha visto qué es Java Micro Edition y se ha enmarcado dentro de la plataforma Java. A continuación se presentan los componentes que forman parte de esta tecnología:

- Por un lado se tiene una serie de máquinas virtuales Java con diferentes requisitos, cada una para diferentes tipos de pequeños dispositivos.
- Configuraciones, que son un conjunto de clases básicas orientadas a conformar el corazón de las implementaciones para dispositivos de características específicas. Existen 2 configuraciones definidas en J2ME: Connected Limited Device Configuration (CLDC) enfocada a dispositivos con restricciones de procesamiento y memoria, y Connected Device Configuration (CDC) enfocada a dispositivos con más recursos.
- Perfiles, que son unas bibliotecas Java de clases específicas orientadas a implementar funcionalidades de más alto nivel para familias específicas de dispositivos.

Un entorno de ejecución determinado de J2ME se compone entonces de una selección de:

a. Maquina virtual.
b. Configuración.
c. Perfil.
d. Paquetes Opcionales.

La arquitectura de un entorno de ejecución se puede ver en la figura 2.3. A continuación se estudiará en profundidad cada uno de estos tres componentes.

Figura 2.3 Entorno de ejecución.

2.3.1 Máquinas Virtuales J2ME

Una máquina virtual de Java (JVM) es un programa encargado de interpretar código intermedio (bytecode) de los programas Java precompilados a código máquina ejecutable por la plataforma, efectuar las llamadas pertinentes al sistema operativo subyacente, observar las reglas de seguridad y corrección de código definidas para el lenguaje Java. De esta forma, la JVM proporciona al programa Java independencia de la plataforma con respecto al hardware y al sistema operativo subyacente. Las implementaciones tradicionales de JVM son, en general, muy pesadas en cuanto a memoria ocupada y requerimientos computacionales. J2ME define varias JVMs de referencia adecuadas al ámbito de los dispositivos electrónicos que, en algunos casos, suprimen algunas características con el fin de obtener una implementación menos exigente.

Ya se ha visto que existen 2 configuraciones CLDC y CDC, cada una con unas características propias. La VM (Virtual Machine) de la configuración CLDC se denomina KVM y la de la configuración CDC se denomina CVM. A continuación se verán las características principales de cada una de ellas:

- **KVM**

  Corresponde con la Máquina Virtual más pequeña desarrollada por Sun. Su nombre KVM proviene de Kilobyte (haciendo referencia a la baja ocupación de memoria, entre 40Kb y 80Kb). Se trata de una implementación de Máquina Virtual reducida y especialmente orientada a dispositivos con bajas capacidades computacionales y de memoria. La KVM está escrita en lenguaje C, aproximadamente unas 24000 líneas de código, y fue diseñada para ser:

  - Pequeña, con una carga de memoria entre los 40Kb y los 80 Kb, dependiendo de la plataforma y las opciones de compilación.
  - Alta portabilidad.
  - Modulable.
  - Lo más completa y rápida posible y sin sacrificar características para las que fue diseñada.

  Sin embargo, esta baja ocupación de memoria hace que posea algunas limitaciones con respecto a la clásica *Java Virtual Machine* (JVM):

  1. No hay soporte para tipos en coma flotante. No existen por tanto los tipos double ni float. Esta limitación está presente porque los dispositivos carecen del hardware necesario para estas operaciones.
  2. No existe soporte para JNI[1] debido a los recursos limitados de memoria.
  3. No existen cargadores de clases (*class loaders*) definidos por el usuario. Sólo existen los predefinidos.
  4. No se permiten los grupos de hilos o hilos *daemon*. Cuándo se quiera utilizar grupos de hilos se utilizan los objetos *Colección* para almacenar cada hilo n el ámbito de la aplicación.
  5. No existe la finalización de instancias de clases. No existe el método Object.finalize().
  6. No hay referencias débiles[2].

[1] Java Native Interface

[2] Un objeto que está siendo apuntado mediante una referencia débil es un candidato para la recolección de basura. Estas referencias están permitidas en J2SE, pero no en J2ME.

7. Limitada capacidad para el manejo de excepciones debido a que el manejo de éstas depende en gran parte de las APIs de cada dispositivo por lo que son éstos los que controlan la mayoría de las excepciones.
8. Reflexión[1]

Aparte de la no inclusión de estas características, la verificación de clases merece un comentario aparte. El verificador de clases estándar de Java es demasiado grande para la KVM. De hecho es más grande que la propia KVM y el consumo de memoria es excesivo, más de 100Kb para las aplicaciones típicas. Este verificador de clases es el encargado de rechazar las clases no válidas en tiempo de ejecución. Este mecanismo verifica los *bytecodes* de las clases Java realizando las siguientes comprobaciones:

- Ver que el código no sobrepase los límites de la pila de la VM.
- Comprobar que no se utilizan las variables locales antes de ser inicializadas.
- Comprobar que se respetan los campos, métodos y los modificadores de control de acceso a clases.

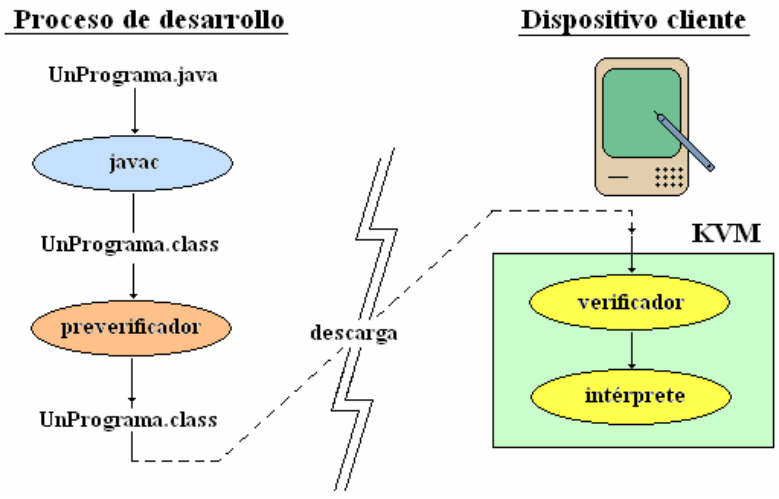


Figura 2.4 Preverificación de clases en CDLC/KVM.

[1] La reflexión es el mecanismo por el cual los objetos pueden obtener información de otros objetos en tiempo de ejecución como, por ejemplo, los archivos de clases cargados o sus campos y métodos.

Por esta razón los dispositivos que usen la configuración CLDC y KVM introducen un algoritmo de verificación de clases en dos pasos. Este proceso puede apreciarse gráficamente en la Figura 2.4.

La KVM puede ser compilada y probada en 3 plataformas distintas:

1. Solaris Operating Environment.
2. Windows
3. PalmOs

- **CVM**

La CVM (Compact Virtual Machine) ha sido tomada como Máquina Virtual Java de referencia para la configuración CDC y soporta las mismas características que la Máquina Virtual de J2SE. Está orientada a dispositivos electrónicos con procesadores de 32 bits de gama alta y en torno a 2Mb o más de memoria RAM. Las características que presenta esta Máquina Virtual son:

1. Sistema de memoria avanzado.
2. Tiempo de espera bajo para el recolector de basura.
3. Separación completa de la VM del sistema de memoria.
4. Recolector de basura modularizado.
5. Portabilidad.
6. Rápida sincronización.
7. Ejecución de las clases Java fuera de la memoria de sólo lectura (ROM).
8. Soporte nativo de hilos.
9. Baja ocupación en memoria de las clases.
10. Proporciona soporte e interfaces para servicios en Sistemas Operativos de Tiempo Real.
11. Conversión de hilos Java a hilos nativos.
12. Soporte para todas las características de Java2 v1.3 y librerías de seguridad, referencias débiles, Interfaz Nativa de Java (JNI),

invocación remota de métodos (RMI[1]), Interfaz de depuración de la Máquina Virtual (JVMDI[2]).

2.3.2 Configuraciones

Ya se ha mencionado algo anteriormente relacionado con las configuraciones, por lo tanto, una configuración es el conjunto mínimo de APIs Java que permiten desarrollar aplicaciones para un grupo de dispositivos. Estas APIs describen las características básicas, comunes a todos los dispositivos:

1. Características soportadas del lenguaje de programación Java.
2. Características soportadas por la Máquina Virtual Java.
3. Bibliotecas básicas de Java y APIs soportadas.

Como se ha visto con anterioridad, existen dos configuraciones en J2ME: CLDC, orientada a dispositivos con limitaciones computacionales y de memoria y CDC, orientada a dispositivos con no tantas limitaciones. Ahora se verán un poco más en profundidad cada una de estas configuraciones.

- **Configuración de dispositivos con conexión, CDC** (*Connected Device Configuration*)

  La CDC está orientada a dispositivos con cierta capacidad computacional y de memoria. Por ejemplo, decodificadores de televisión digital, televisores con internet, algunos electrodomésticos y sistemas de navegación en automóviles. CDC usa una Máquina Virtual Java similar en sus características a una de J2SE, pero con limitaciones en el apartado gráfico y de memoria del dispositivo. Ésta Máquina Virtual es la que ha visto como CVM (Compact Virtual Machine). La CDC está enfocada a dispositivos con las siguientes capacidades:

  - Procesador de 32 bits.
  - Disponer de 2 Mb o más de memoria total, incluyendo memoria RAM y ROM.

[1] Remote Method Invocation
[2] JVMDI: Java Vitual Machina Depurate Interface

- Poseer la funcionalidad completa de la Máquina Virtual Java2.
- Conectividad a algún tipo de red.

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 2.1 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 reflexión
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 2.1 Librerías de configuración CDC.

- **Configuración de dispositivos limitados con conexión, CLDC** (*Connected Limited Device Configuration*).

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.

Ya se ha dicho que 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:

- Disponer entre 160 Kb y 512 Kb de memoria total disponible. Como mínimo se debe disponer de 128 Kb de memoria no volátil para la Máquina Virtual Java y las bibliotecas CLDC, y 32 Kb de memoria volátil para la Máquina Virtual en tiempo de ejecución.
- Procesador de 16 o 32 bits con al menos 25 MHz de velocidad.
- Ofrecer bajo consumo, debido a que estos dispositivos trabajan con suministro de energía limitado, normalmente baterías.
- Tener conexión a algún tipo de red, normalmente sin cable, con conexión intermitente y ancho de banda limitado (unos 9600 bps).

La CLDC aporta las siguientes funcionalidades a los dispositivos:

- Un subconjunto del lenguaje Java y todas las restricciones de su Máquina Virtual (KVM).
- Un subconjunto de las bibliotecas Java del núcleo.
- Soporte para E/S básica.
- Soporte para acceso a redes.
- Seguridad.

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

Tabla 2.2 Librerías de configuración CLDC.

Un aspecto muy a tener en cuenta es la seguridad en CLDC. Esta configuración posee un modelo de seguridad *sandbox* al igual que ocurre con los applets.

En cualquier caso, una determinada Configuración no se encarga del mantenimiento del ciclo de vida de la aplicación, interfaces de usuario o manejo de eventos, sino que estas responsabilidades caen en manos de los **perfiles**.

2.3.3 Perfiles

Anteriormente se dijo que 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. Aquí se puede encontrar grandes diferencias entre interfaces, desde el menú textual de los teléfonos móviles hasta los táctiles de los PDAs.

El perfil establece unas APIs que definen las características de un dispositivo, mientras que la configuración hace lo propio con una familia de ellos. Esto hace que a la hora de construir una aplicación se cuente tanto con las APIs del perfil como de la configuración. Se debe tener en cuenta que un perfil siempre se construye sobre una configuración determinada. De este modo, se puede pensar en un perfil como un conjunto de APIs que dotan a una configuración de funcionalidad específica.

Ya se han explicado los conceptos necesarios para entender cómo es un entorno de ejecución en Java Micro Edition. Este entorno de ejecución se estructura en capas, una construida sobre la otra como se veía en la figura 2.3.

Anteriormente se explicó que para una configuración determinada se usaba una Máquina Virtual Java específica. Por lo que, con la configuración CDC se usa la CVM y que con la configuración CLDC se emplea la KVM. Con los perfiles ocurre lo mismo. Existen unos perfiles que se construirán sobre la configuración CDC y otros que construirán sobre la CLDC. Para la configuración CDC se tienen los siguientes perfiles:

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

Y para la configuración CLDC se tienen los siguientes:

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

En la Figura 2.5 se puede ver como quedaría el esquema del entorno de ejecución al completo. Un perfil puede ser construido sobre cualquier otro. Sin embargo, una plataforma J2ME sólo puede contener una configuración.

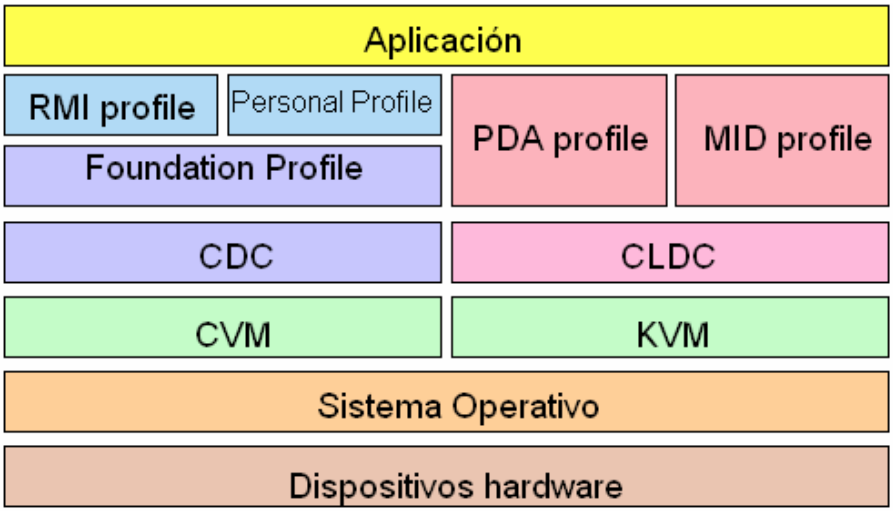


Figura 2.5 Arquitectura del entorno de ejecución de J2ME.

A continuación se verá brevemente cada uno de estos perfiles (ya que se entenderá más adelante, que para este documento el estudio interesa solamente en el perfil MIDP bajo la configuración CLDC) :

- **Foundation Profile:** Este perfil define una serie de APIs sobre la CDC orientadas a dispositivos que carecen de interfaz gráfica como, por ejemplo, decodificadores de televisión digital.

- **Personal Profile:** El *Personal Profile* es un subconjunto de la plataforma J2SE v1.3, y proporciona un entorno con un completo soporte gráfico AWT[1]. El objetivo es el de dotar a la configuración CDC de una interfaz gráfica completa, con capacidades web y soporte de *applets* Java. Este perfil requiere una implementación del *Foundation Profile*.

- **RMI Profile:** Este perfil requiere una implementación del *Foundation Profile,* se construye encima de él. El perfil RMI[2] soporta un subconjunto de las APIs J2SE v1.3 RMI. Algunas características de estas APIs se han eliminado del perfil RMI debido a las limitaciones de cómputo y memoria de los dispositivos.

- **PDA Profile:** El PDA Profile está construido sobre CLDC. Pretende abarcar PDAs de gama baja, tipo Palm, con una pantalla y algún tipo de puntero (ratón o lápiz) y una resolución de al menos 20000 píxeles (al menos 200x100 píxeles) con un factor 2:1. No es posible dar mucha más información porque en este momento este perfil se encuentra en fase de definición.

- **Mobile Information Device Profile (MIDP):** Este perfil está construido sobre la configuración CLDC. Al igual que CLDC fue la primera configuración definida para J2ME, MIDP fue el primer perfil definido para esta plataforma. Este perfil está orientado para dispositivos con las siguientes características:
  - Reducida capacidad computacional y de memoria.

[1] AWT: Abstract Window Toolkit, permite hacer interfaces gráficas mediante dispositivos de interacción con el usuario
[2] Remote Method Invocation

- Conectividad limitada (en torno a 9600 bps).
- Capacidad gráfica muy reducida (mínimo un display de 96x54 píxeles monocromo).
- Entrada de datos alfanumérica reducida.
- 128 Kb de memoria no volátil para componentes MIDP.
- 8 Kb de memoria no volátil para datos persistentes de aplicaciones.
- 32 Kb de memoria volátil en tiempo de ejecución para la pila Java.

Los tipos de dispositivos que se adaptan a estas características son: teléfonos móviles, buscapersonas (pagers) o PDAs de gama baja con conectividad. El perfil MIDP establece las capacidades del dispositivo, por lo tanto, especifica las APIs relacionadas con:

- La aplicación (semántica y control de la aplicación MIDP).
- Interfaz de usuario.
- Almacenamiento persistente.
- Trabajo en red.
- Temporizadores.

En la Tabla 2.3 se puede ver cuáles son los paquetes que están incluidos en el perfil MIDP.

Las aplicaciones que se realizan utilizando MIDP reciben el nombre de *MIDlets* (por simpatía con *APPlets*). **Decimos así que un MIDlet es una aplicación Java realizada con el perfil MIDP sobre la configuración CLDC**.

Paquetes del MIDP	Descripción
javax.microedition.lcdui	Clases e interfaces para GUIs[26]
javax.microedition.rms	*Record Management Storage.* Soporte para el almacenamiento persistente del dispositivo
javax.microedition.midlet	Clases de definición de la aplicación
javax.microedition.io	Clases e interfaces de conexión genérica
java.io	Clases e interfaces de E/S básica
java.lang	Clases e interfaces de la Maquina Virtual
java.util	Clases e interfaces de utilidades estándar

Tabla 2.3 Librerías del perfil MIDP.

En el siguiente capítulo se centrará el estudio en la creación de *MIDlets* ya que es un punto de referencia para cualquier programador de J2ME. Además, desde un punto de vista práctico MIDP es el único perfil actualmente disponible.

2.4 J2ME y las comunicaciones

Ya se ha visto que una característica importante que deben tener los dispositivos que hagan uso de J2ME, más específicamente CLDC/MIDP (a partir de ahora el estudio de J2ME se va a basar exclusivamente en el entorno CLDC/MIDP) es que necesitan poseer conexión a algún tipo de red, por lo que la comunicación de estos dispositivos cobra una gran importancia. En este apartado se verá cómo participan las distintas tecnologías en estos dispositivos y cómo influyen en el uso de la tecnología J2ME. Para ello se centrará en un dispositivo en especial: los teléfonos móviles. De aquí en adelante todo el estudio y la creación de la aplicación se realizarán 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 usarán para crear las aplicaciones.

Uno de los primeros avances de la telefonía móvil en el sector de las comunicaciones se dio con la aparición de la tecnología WAP[27]. Es un protocolo con el que se ha tratado de dotar a los dispositivos móviles de un pequeño y limitado

[26] GUI: Graphic User Interface
[27] WMA: Wireless Messaging Application

navegador web. WAP exige la presencia de una puerta de enlace encargado de actuar cómo intermediario entre Internet y el terminal. Esta puerta de enlace o *gateway* es la que se encarga de convertir las peticiones WAP a peticiones web habituales y viceversa. Las páginas que se transfieren en una petición usando WAP no están escritas en HTML[28], si no que están escritas en WML[29], un subconjunto de éste. WAP ha sido un gran avance, pero no ha resultado ser la herramienta que se prometía. La navegación es muy engorrosa (la introducción de URLs largas por teclado es muy pesada, además de que cualquier error en su introducción requiere que se vuelva a escribir la dirección completa por el teclado del móvil). Además su costo es bastante elevado ya que el pago de uso de esta tecnología se realiza en base al tiempo de conexión a una velocidad, que en realidad, no es muy buena.

Otra tecnología relacionada con los móviles es el Servicio de Mensajes Cortos SMS[30]. Actualmente este sistema permite comunicar de una manera rápida y barata con quien se quiera sin tener que establecer una comunicación con el receptor del mensaje. Con ayuda de J2ME, sin embargo, se puede realizar aplicaciones de chat o mensajería instantánea.

Los últimos avances de telefonía móvil llevan al mundo a las conocidas cómo generación 2 y 2.5 que hacen uso de las tecnologías GSM y GPRS respectivamente. GSM es una conexión telefónica que soporta una circulación de datos, mientras que GPRS es estrictamente una red de datos que mantiene una conexión abierta en la que el usuario paga por la cantidad de información intercambiada y no por el tiempo que permanezca conectado. La aparición de la tecnología GPRS no hace más que favorecer el uso de J2ME y es, además, uno de los pilares sobre los que se asienta J2ME, ya que se puede decir que es el vehículo sobre el que circularán las futuras aplicaciones J2ME.

[28] HTML: Hyper Text Markup Language
[29] WML: Wireless Markup Language
[30] SMS: Short Message Service

Otras tecnologías que favorecen la comunicación son Bluetooth y las redes inalámbricas que dan conectividad a ordenadores, PDAs y teléfonos móviles. De hecho, una gran variedad de estos dispositivos disponen de soporte bluetooth. Esto facilita la creación de redes con un elevado ancho de banda en distancias pequeñas (hasta 100 metros).

Es de esperar que todas estas tecnologías favorezcan el uso de J2ME en el mercado de la telefonía móvil.

Capítulo 3:
LOS MIDLET Y LA CONFIGURACION CLDC

3. LOS MIDLETS Y LA CONFIGURACIÓN CLDC [3]

3.1. Descripción del capítulo

En este capítulo se dará primeramente unas nociones básicas sobre los *MIDlets* y posteriormente se estudiará la configuración CLDC de J2ME. Aunque ya se ha hablado de ellos, ya se sabe qué son, en este capítulo se profundizará en su estudio.

Con respecto a los MIDlets, se verá cuáles son sus propiedades, se conocerá su ciclo de vida y estados por los que pasa. Se explicará también del gestor de aplicaciones y la relación de éste con los *MIDlets.* Los *MIDlets* son aplicaciones creadas usando la especificación MIDP. Están diseñados para ser ejecutados, como ya se sabe, en dispositivos con poca capacidad gráfica, de cómputo y de memoria. En estos dispositivos no se dispone de líneas de comandos donde poder ejecutar las aplicaciones que se quieran, si no que reside en él un *software* que es el encargado de ejecutar los *MIDlets* y gestionar los recursos que éstos ocupan.

Retomando a la configuración CLDC de J2ME, se verá con qué propósito se creó CLDC y cuales son sus objetivos y metas. Se presentarán los aspectos en que se centra la configuración CLDC. La configuración CLDC se ocupa de las siguientes áreas:

- Lenguaje Java y características de la máquina virtual.
- Librerías del núcleo de Java (java.lang.* y java.util.*).
- Entrada / Salida.
- Comunicaciones.
- Seguridad.
- Internacionalización.

Lo que se intenta con la configuración CLDC es mantener lo más pequeño posible el número de áreas abarcadas. Es mejor restringir el alcance de CLDC para no

exceder las limitaciones de memoria o no excluir ningún dispositivo en particular. En el futuro, la configuración CLDC podría incluir otras áreas adicionales.

3.2 Los Midlets

3.2.1 El Gestor de Aplicaciones

El gestor de aplicaciones o AMS[31] es el *software* encargado de gestionar los *MIDlets*. Este *software* reside en el dispositivo y es el que permite ejecutar, pausar o destruir las aplicaciones J2ME.

El AMS realiza dos grandes funciones:

- Por un lado gestiona el ciclo de vida de los *MIDlets*.
- Por otro, es el encargado de controlar los estados por los que pasa el *MIDlet* mientras está en la memoria del dispositivo, es decir, en ejecución.

3.2.1.1 Ciclo de vida de un MIDlet

El ciclo de vida de un MIDlet pasa por 5 fases (ver Figura 3.1): descubrimiento, instalación, ejecución, actualización y borrado.

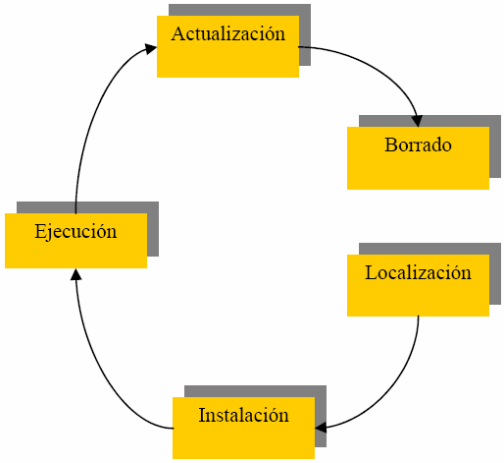


Figura 3.1 Ciclo de vida de un *MIDlet*.

El AMS es el encargado de gestionar cada una de estas fases de la siguiente manera:

[31] AMS: Application Management System

1. Descubrimiento: Esta fase es la etapa previa a la instalación del MIDlet y es dónde se selecciona a través del gestor de aplicaciones la aplicación a descargar. Por tanto, el gestor de aplicaciones tiene que proporcionar los mecanismos necesarios para realizar la elección del MIDlet a descargar. El AMS puede ser capaz de realizar la descarga de aplicaciones de diferentes maneras, dependiendo de las capacidades del dispositivo. Por ejemplo, esta descarga la se puede realizar mediante un cable conectado a la computadora o mediante una conexión inalámbrica.
2. Instalación: Una vez es cargado el MIDlet en el dispositivo, comienza el proceso de instalación. En esta fase el gestor de aplicaciones controla todo el proceso informando al usuario tanto de la evolución de la instalación como de si existiese algún problema durante ésta. Cuándo un MIDlet está instalado en el dispositivo, todas sus clases, archivos y almacenamiento persistente están preparados y listos para su uso.
3. Ejecución: Mediante el gestor de aplicaciones el usuario será capaz de iniciar la ejecución de los *MIDlets*. En esta fase, el AMS tiene la función de gestionar los estados del MIDlet en función de los eventos que se produzcan durante esta ejecución. Esto se verá un poco a profundidad más adelante.
4. Actualización: El AMS tiene que ser capaz de detectar después de una descarga si el MIDlet descargado es una actualización de un MIDlet ya presente en el dispositivo. Si es así, tiene que informar de ello, además de dar la oportunidad de decidir de querer realizar la actualización pertinente o no.
5. Borrado: En esta fase el AMS es el encargado de borrar el MIDlet seleccionado del dispositivo. El AMS pedirá confirmación antes de proceder a su borrado e informará de cualquier circunstancia que se produzca.

Hay que indicar que el MIDlet puede permanecer en el dispositivo todo el tiempo que queramos. Después de la fase de instalación, el MIDlet queda almacenado en una zona de memoria persistente del dispositivo MID. El usuario de éste dispositivo es el encargado de decidir en qué momento quiere eliminar la

aplicación y así se lo hará saber al AMS mediante alguna opción que éste nos suministre.

3.2.1.2. Estados de un MIDlet en fase de ejecución

Además de gestionar el ciclo de vida de los *MIDlets*, como se ha visto, el AMS es el encargado de controlar los estados del MIDlet durante su ejecución. Durante ésta el *MIDlet* es cargado en la memoria del dispositivo y es aquí donde puede transitar entre 3 estados diferentes: Activo, en pausa y destruido.

Cuándo un *MIDlet* comienza su ejecución, está en el estado "Activo" pero, ¿qué ocurre si durante su ejecución se recibe una llamada o un mensaje? El gestor de aplicaciones debe ser capaz de cambiar el estado de la aplicación en función de los eventos externos al ámbito de ejecución de la aplicación que se vayan produciendo. En este caso, el gestor de aplicaciones interrumpiría la ejecución del *MIDlet* sin que se viese afectada la ejecución de éste y lo pasaría al estado de "Pausa" para atender la llamada o leer el mensaje. Una vez que se termine de trabajar con el *MIDlet* y salga de él, éste pasaría al estado de "Destruido" dónde sería eliminado de la memoria del dispositivo. Cuándo se decide que el *MIDlet* pasa al estado "Destruido" y es eliminado de memoria, se refiere a la memoria volátil del dispositivo que es usada para la ejecución de aplicaciones. Una vez finalizada la ejecución del *MIDlet se* puede volver a invocarlo las veces que se desean ya que éste permanece en la zona de memoria persistente hasta el momento que se desee desinstalarlo.

3.2.1.3. Estados de un MIDlet

Un *MIDlet* durante su ejecución pasa por 3 estados diferentes. Como ya se ha mencionado en el apartado anterior, estos tres estados son:

- Activo: El MIDlet está actualmente en ejecución.

- Pausa: El *MIDlet* no está actualmente en ejecución. En este estado el *MIDlet* no debe usar ningún recurso compartido. Para volver a pasar a ejecución tiene que cambiar su estado a Activo.
- Destruido: El *MIDlet* no está en ejecución ni puede transitar a otro estado. Además se liberan todos los recursos ocupados por el *MIDlet*.

La Figura 3.2 muestra el diagrama de estados de un *MIDlet* en ejecución:

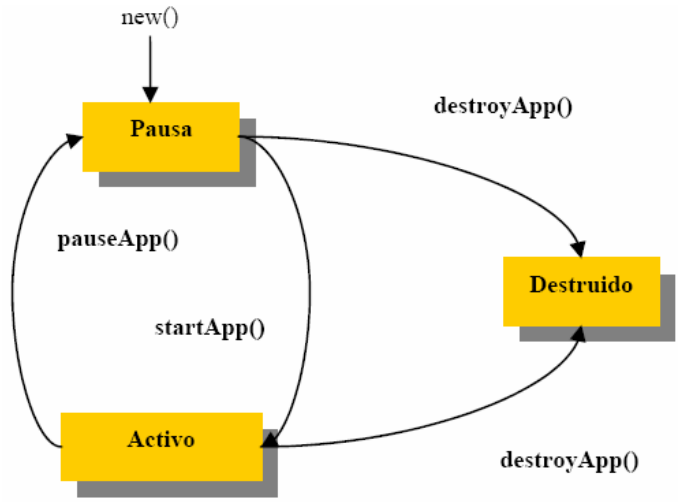


Figura 3.2 Estados de un *MIDlet*.

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

Ahora detallarán los estados por los que pasa un *MIDlet* durante una ejecución típica y cuáles son las acciones que realizan tanto el AMS como el *MIDlet*. En primer lugar, se realiza la llamada al constructor del MIDlet pasando éste al estado de "Pausa" durante un corto período de tiempo. El AMS por su parte crea una nueva instancia del *MIDlet*. Cuándo el dispositivo está preparado para ejecutar el MIDlet, el AMS invoca al método MIDlet.startApp() para entrar en el estado de "Activo". El *MIDlet* entonces, ocupa todos los recursos que necesita

para su ejecución. Durante este estado, el MIDlet puede pasar al estado de "Pausa" por una acción del usuario, o bien, por el AMS que reduciría en todo lo posible el uso de los recursos del dispositivo por parte del MIDlet. Tanto en el estado "Activo" como en el de "Pausa", el *MIDlet* puede pasar al estado "Destruido" realizando una llamada al método MIDlet.destroyApp(). Esto puede ocurrir porque el *MIDlet* haya finalizado su ejecución o porque una aplicación prioritaria necesite ser ejecutada en memoria en lugar del *MIDlet*. Una vez destruido el *MIDlet*, éste libera todos los recursos ocupados.

3.2.2. Estructura de los MIDlets

Conociendo el estado de los midlet y su ciclo de vida, se puede ver ahora cuál es su estructura. Se dirá que los *MIDlets*, al igual que los *applets* carecen de la función main(). Aunque existiera, el gestor de aplicaciones la ignoraría por completo. (Un *MIDlet* tampoco puede realizar una llamada a System.exit(). Una llamada a este método lanzaría la excepción SecurityException.)

Los *MIDlets* tienen la siguiente estructura:

```
import javax.microedition.midlet.*

public class MiMidlet extends MIDlet{
        public MiMidlet() {
                /* Éste es el constructor de clase. Aquí se debe
                inicializar las variables.
                */
        }
        public startApp(){
                /* Aquí se incluye el código que se quiere que el
                MIDlet ejecute cuándo se active.
                */
        }
```

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

3.3 La Configuración CLDC

3.3.1 Objetivos y requerimientos

Una configuración de J2ME especifica un subconjunto de características soportadas por el lenguaje de programación Java, un subconjunto de funciones de la configuración para la máquina virtual de Java, el trabajo en red, seguridad, instalación y, posiblemente, otras APIs de programación, todo lo necesario para soportar un cierto tipo de productos.

CLDC es la base para uno o más perfiles. Un perfil de J2ME define un conjunto adicional de APIs y características para un mercado concreto, dispositivo determinado o industria.

3.3.1.1 Objetivos

El objetivo principal de la especificación CLDC es definir un estándar, para pequeños dispositivos conectados con recursos limitados con las siguientes características:

- 160 Kb a 512 Kb de memoria total disponible para la plataforma Java.
- Procesador de 16 bits o 32 bits.
- Bajo consumo, normalmente el dispositivo usa una batería.
- Conectividad a algún tipo de red, normalmente inalámbrica, con conexión intermitente y ancho de banda limitado (unos 9600 bps).

Teléfonos móviles, PDAs y terminales de venta, son algunos de los dispositivos que podrían ser soportados por esta especificación. Esta configuración J2ME define los componentes mínimos y librerías requeridas por dispositivos conectados pequeños.

Aunque el principal objetivo de la especificación CLDC es definir un estándar para dispositivos pequeños y conectados con recursos limitados, existen otros objetivos que son los siguientes:

- Extensibilidad: Uno de los grandes beneficios de la tecnología Java en los pequeños dispositivos es la distribución dinámica y de forma segura de contenido interactivo y aplicaciones sobre diferentes tipos de red. Hace unos años, estos dispositivos se fabricaban con unas características fuertemente definidas y sin capacidad apenas de extensibilidad. Los desarrolladores de dispositivos empezaron a buscar soluciones que permitieran construir dispositivos extensibles que soportaran una gran variedad de aplicaciones provenientes de terceras partes. Con la reciente introducción de teléfonos conectados a internet, comunicadores, etc. La transición está actualmente en marcha. Uno de los principales objetivos de CLDC es tomar parte en esta transición permitiendo el uso del lenguaje de programación Java como base para la distribución de contenido dinámico para esta próxima generación de dispositivos.

- Desarrollo de aplicaciones por terceras partes: La especificación CLDC solo deberá incluir librerías de alto nivel que proporcionen suficiente capacidad de programación para desarrollar aplicaciones por terceras partes. Por esta razón, las APIs de red incluidas en CLDC deberían proporcionar al programador una abstracción de alto nivel como, por ejemplo, la capacidad de transferir archivos enteros, aplicaciones o páginas web, en vez de requerir que el programador conozca los detalles de los protocolos de transmisión para una red específica.

3.3.1.2 Requerimientos

Se puede clasificar los requerimientos en *hardware, software* y requerimientos basados en las características Java de la plataforma J2ME.

- Requerimientos hardware: CLDC está diseñado para ejecutarse en una gran variedad de dispositivos, desde aparatos de comunicación inalámbricos como teléfonos móviles, buscapersonas, hasta organizadores personales, terminales de venta, etc. Las capacidades del hardware de estos dispositivos varían considerablemente y por esta razón, los únicos requerimientos que impone la configuración CLDC son los de memoria. La configuración CLDC asume que la máquina virtual, librerías de configuración, librerías de perfil y la aplicación debe ocupar una memoria entre 160 Kb y 512 Kb. Más concretamente:
  - 128 Kb de memoria no volátil para la JVM y las librerías CLDC.
  - Al menos 32 Kb de memoria volátil para el entorno de ejecución Java y objetos en memoria.

  Este rango de memoria varía considerablemente dependiendo del dispositivo al que hagamos referencia.
- Requerimientos software: Al igual que las capacidades hardware, el software incluido en los dispositivos CLDC varía considerablemente. Por ejemplo, algunos dispositivos pueden poseer un sistema operativo (S.O.) completo que soporta múltiples procesos ejecutándose a la vez y con sistema de archivos jerárquico. Otros muchos dispositivos pueden poseer

un software muy limitado sin noción alguna de lo que es un sistema de ficheros. Dentro de esta variedad, CLDC define unas mínimas características que deben poseer el software de los dispositivos CLDC. Generalmente, la configuración CLDC asume que el dispositivo contiene un mínimo Sistema Operativo encargado del manejo del hardware de éste. Este S.O. debe proporcionar al menos una entidad de planificación para ejecutar la JVM. El S.O. no necesita soportar espacios de memoria separados o procesos, ni debe garantizar la planificación de procesos en tiempo real o comportamiento latente.

- Requerimientos J2ME: CLDC es definida como la configuración de J2ME. Esto tiene importantes implicaciones para la especificación CLDC:
  - Una configuración J2ME solo deber definir un complemento mínimo de la tecnología Java. Todas las características incluidas en una configuración deben ser generalmente aplicables a una gran variedad de dispositivos. Características específicas para un dispositivo, mercado o industria deben ser definidas en un perfil en vez de en el CLDC. Esto significa que el alcance de CLDC está limitado y debe ser complementado generalmente por perfiles.
  - El objetivo de la configuración es garantizar portabilidad e interoperabilidad entre varios tipos de dispositivos con recursos limitados. Una configuración no debe definir ninguna característica opcional. Esta limitación tiene un impacto significativo en lo que se puede incluir en una configuración y lo que no. La funcionalidad más específica debe ser definida en perfiles.

3.3.2 Seguridad en CLDC

Debido a las características de los dispositivos englobados bajo CLDC, en los que se hace necesaria la descarga de aplicaciones y la ejecución de éstas en dispositivos que almacenan información muy personal, se hace imprescindible hacer un gran hincapié en la seguridad. Hay que asegurar la integridad de los datos transmitidos y de las aplicaciones. Éste modelo de seguridad no es nuevo,

ya que la ejecución de applets (programas Java que se ejecutan en un navegador web) se realiza en una zona de seguridad denominada *sandbox*. Los dispositivos englobados en CLDC se encuentran ante un modelo similar al de los applets.

Este modelo establece que sólo se pueden ejecutar algunas acciones que se consideran seguras. Existen, entonces, algunas funcionalidades críticas que están fuera del alcance de las aplicaciones. De esta forma, las aplicaciones ejecutadas en estos dispositivos deben cumplir unas condiciones previas:

- Los ficheros de clases Java deben ser verificados como aplicaciones Java válidas.
- Sólo se permite el uso de APIs autorizadas por CLDC.
- No está permitido cargar clases definidas por el usuario.
- Sólo se puede acceder a características nativas que entren dentro del CLDC.
- Una aplicación ejecutada bajo KVM no debe ser capaz de dañar el dispositivo dónde se encuentra. De esto se encarga el verificador de clases que se asegura que no haya referencias a posiciones no válidas de memoria. También comprueba que las clases cargadas no se ejecuten de una manera no permitida por las especificaciones de la Máquina Virtual.

3.4 Dispositivos actuales que soportan J2ME [4]

La CLDC está orientada a dispositivos dotados de conexión y con limitaciones en cuanto a capacidad gráfica, cómputo y memoria. Ejemplo de ello son los teléfonos móviles. Actualmente, son muchos los fabricantes de estos terminales y así también la variedad de modelos de los mismos. En la figura 3.3 se presenta una muestra de la variedad de teléfonos móviles y otros dispositivos que existen en el mercado que soportan J2ME.

Figura 3.3 Dispositivos que soportan J2ME

Dentro de estos dispositivos se ha seleccionado el teléfono móvil **V600,** debido a que dentro del mercado de teléfonos móviles nacional es uno de los que más auge ha tenido comercialmente, y más específicamente, porque MOTOROLA pone a disposición en su página WEB [8] el *Motorota SDK v4.4 for J2ME* donde incluye un emulador y el *MIDway* la cual es un programa que permite transferir los midlet al terminal.

Capítulo 4:
MENSAJERIA INALAMBRICA: SMS

4. MENSAJERIA INALÁMBRICA: SERVICIOS DE MENSAJES CORTO (SMS)

4. 1 Descripción del capítulo

En GSM hay un servicio característico llamado servicio de mensajes cortos, o SMS (siglas de *Short Message Service*). Fue originalmente diseñado para enviar mensajes cortos en el canal de control de la red GSM, pero ha evolucionado gradualmente hasta convertirse en un servicio general de datos.

En este capítulo se dará una introducción respecto al servicio de mensajes cortos para conocer su definición, estructura, su arquitectura dentro de la red GSM y su forma de viajar hacia dicha red.

Posteriormente, se ampliará el SMS desde el punto de vista de aplicación dentro de la J2ME (Midlets) mencionando su estándar por medio del JSR 120: WMA, estructuras de clases y requerimientos. Para luego, llegar a la aplicación del SMS por medio del midlet en un teléfono móvil: Motorola V600, y así mostrar las especificaciones del fabricante para con los SMS.

4.2 Servicio de Mensaje Corto (SMS)

4.2.1 Definición [5]

El servicio de mensajes cortos (SMS), es un servicio que permite el envió de caracteres alfanuméricos (hasta 160 caracteres) a un móvil. El SMS permite a cualquier usuario el envío de mensajes a cualquier terminal en cualquier momento. El servicio está basado en un mecanismo de "*store-and-forward*", el cual garantiza que cuando el móvil de destino no pueda ser alcanzado (ya sea

que esté apagado o porque está fuera de cobertura), el mensaje será eventualmente enviado cuando el terminal móvil acceda nuevamente a la red.
El SMS utiliza un enlace dedicado (como en el establecimiento de una llamada), en un canal de radio dúplex. Esto tiene la ventaja que la red y el móvil están en una comunicación directa todo el tiempo que dura la transmisión del mensaje.

4.2.2 Arquitectura SMS [6]

En los actuales sistemas de telefonía móvil, el SMS es usado para implementar servicios de alcance. El SMS es un servicio "*store-and-forward*", lo que significa que los mensajes no son directamente enviados de el remitente al destinatario, sino siempre por medio de un Centro de Servicios de Mensajes Cortos, o SMSC[32]. El SMSC reside en la red de operadoras y administra los procesos incluyendo el encolar los mensajes, facturar al remitente, etc. Muchas operadoras actualmente ofrecen interfaces basados en web para sus SMSC, así que los usuarios pueden enviar mensajes cortos a cualquier teléfono móvil desde la web.

Cuando un mensaje SMS es recibido por el SMSC, debe dirigir el mensaje a su apropiado dispositivo móvil. Para esto, el SMSC envía una Solicitud SMS al *Home location register* (HLR) para encontrar al cliente. Una vez el HLR recibe la solicitud, responderá al SMSC con el estatus de suscriptor, la cual está activo o inactivo. Si el suscriptor está inactivo, el SMSC retendrá el mensaje por un periodo de tiempo. Cuando el suscriptor se active, el HLR envía una notificación SMS a l SMSC, y el SMSC atenderá la entrega.

El SMSC transfiere el mensaje en una entrega de mensaje corto de formato de punto a punto al sistema servidor. El sistema llama al dispositivo, y si responde, el mensaje es enviado. El SMSC recibe verificación que fue recibido por el usuario final, categoriza el mensaje como "enviado" y no atenderá al envío de nuevo.

[32] Short Message Service Center

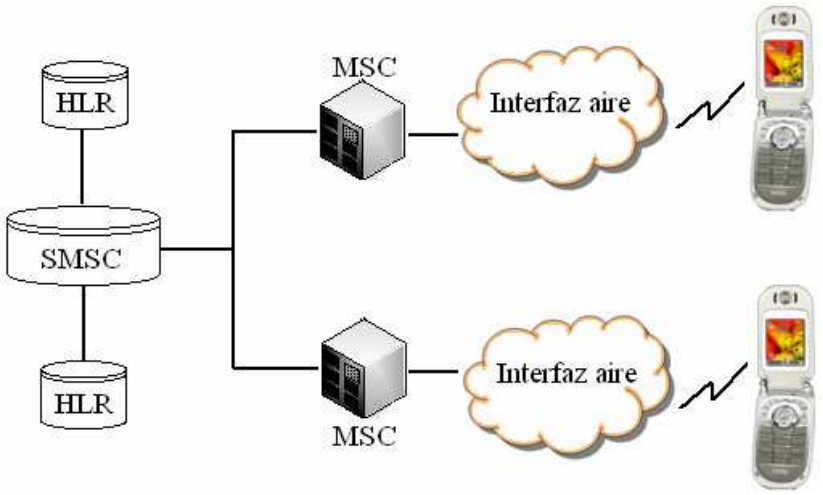


Figura 4.1 El SMSC

Por lo tanto, el SMS consiste de dos servicios básicos:

- SMS terminado en el móvil (MT SMS): De un SMSC a un MS
- SMS originado en el móvil (MO SMS): De un MS a un SMSC

4.2.3 Canales de control (medio de transporte del SMS) [5]

Los SMS viajan a través de los canales de comunicación. Los SMS pueden ser enviados o recibidos desde un MS independientemente el estado en que se encuentre. Estos estados son: Idle o desocupado y el estado activo.

En estado desocupado, los SMS son enviados en un canal dedicado de señalización (SDCCH[33]), este resulta la forma más rápida de enviar los mensajes. Por lo contrario, en estado activo, el canal de radio es utilizado para un canal de trafico (TCH[34]). En este caso, el mensaje utiliza un canal de control asociado (SACCH[35]), aunque no es el más apropiado.

[33] SDCCH: Stand-Alone Dedicated Control Channel
[34] TCH: Traffic Channel
[35] SACCH: Slow Associated Control Channel

4.2.4 El Futuro del SMS [6]

El SMS ha sido un gran suceso en muchos países. Las operadoras en estos países han proveído también a sus suscriptores con posibilidades para personalizar sus teléfonos móviles con *ring tones* o íconos gráficos. El crecimiento es esta área servirá como un camino valorable para nuevas e interesantes maneras de usar el teléfono móvil. Pero para poder enviar fotos a colores, video clips o sonidos MP3, el máximo tamaño de SMS (160 caracteres) no es suficiente. Hay necesidad por un estándar donde los usuarios puedan enviar gran cantidad de datos. Hay también necesidad de otros métodos de facturación que trabaje en relación al tamaño del mensaje.

La respuesta a este nuevo requerimiento podría ser el nuevo estándar llamado Servicio de Mensajes Multimedia, o MMS[36], la cual actualmente se está abriendo campo. El MMS permitirá a los dispositivos de los usuarios soportar el envío y recepción de mensajes con texto, incluyendo graficas, imágenes tanto como audio y video clips.

Aunque servicios como el MMS, habilitado por WAP y UMTS, probablemente reemplazará al SMS como el medio más popular para aplicaciones inalámbricas, habrá todavía una gran cantidad de usuarios empleando SMS por un largo tiempo.

4.3 Interface de Programa de Aplicación en Mensaje Inalámbrica J2ME (The J2ME Wireless Messaging API, JSR 120) [7]

La JSR[37] 120 define una colección de APIs que provee acceso estándar para recursos de comunicación inalámbrica dentro de los MIDlets. La JSR 120 se conoce como *Wireless Messaging API* (WMA).

[36] Multimedia Messaging Service
[37] Java Specification Request

La WMA es diseñada para correr sobre configuraciones J2ME (CDC o CLDC) y mejorar el perfil. La WMA ofrecerá una colección de componentes reusables que pueden ser usados solos o en cualquier combinación con cualquier perfil J2ME.

Una muy importante característica de la WMA es que permite a los dispositivos J2ME, en nuestro caso un teléfono móvil, correr servidores de aplicaciones SMS. Se usa un servidor SMS para procesar automáticamente y responder a mensajes entrantes en la aplicación J2ME. Distinto a los servidores http tradicionales, los servidores SMS no depende de una red IP ya que las direcciones de servidores son identificados por números telefónicos. Por lo que WMA provee una noción de abrir una conexión basada sobre una serie de dirección y cuya conexión puede ser abierta en modo cliente o servidor.

La especificación JSR 120 determina que los desarrolladores pueden ser proveídos de acceso a enviar (MO mobile originated) y recibir (MT – mobile terminated) SMS en el dispositivo destino. La implementación de la especificación JSR 120 posee las siguientes características:

- Crear un SMS
- Enviar un SMS
- Recibir un SMS
- Ver un SMS
- Borrar un SMS

La interface de la MWA ha sido definida en el paquete **javax.wireless.messaging**.

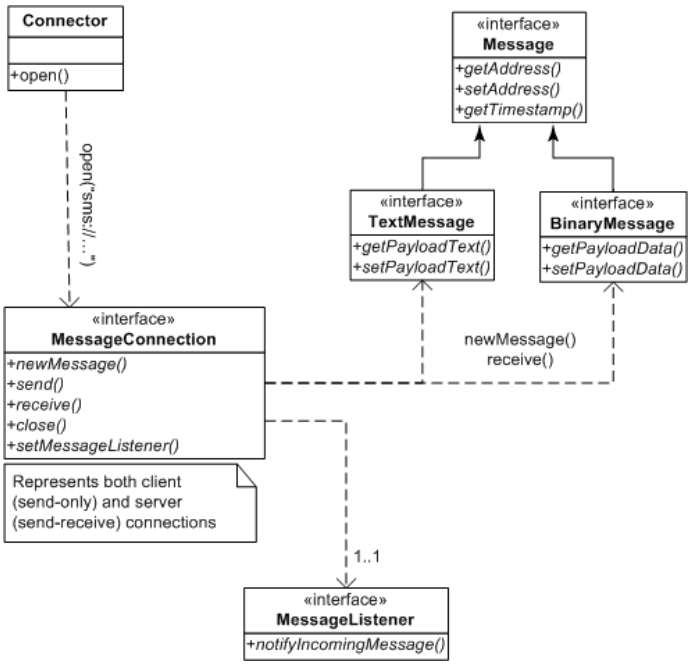


Figura 4.2 Interfaces WMA en el paquete javax.wireless.messaging

4.3.1 Clases de alto nivel en WMA [7]

Los desarrolladores de aplicación pueden acceder al WMA a través de tres interfaces de alto nivel en el paquete javax.wireless.messaging:

- La interface *Message* define la estructura de un mensaje. La interface *TextMessage* y *BinaryMessage* son derivados de *Message* y proporciona una estructura de mensaje más específica.
- La interface *MessageConnection* representa una red conexión para mensajes. Define métodos básicos para enviar y recibir mensajes. Por ejemplo, el método *MessageConnection.newMessage()* retorna instancias del *Message* para mensajes de salida; el método *MessageConnection.receive()* captura mensajes de entrada.
- La interface *MessageListener* tiene solo un método: *notifyIncomingMessage().* Una instancia *MessageListener* es registrada con un servidor *MessageConnection.* Su método *notifyIncomingMessage()* es llamado cuando hay un mensaje entrante.

La figura 4.2 muestra las interfaces del WMA y sus conexiones entre si.

4.3.2 URLs y conexiones de mensaje [8]

El conector *General Connection Framework* (GCF) de java reside en la clase *javax.microedition.io* la cual está definida en la CLDC 1.0. El conector instancia al *MessageConnection*. El URL que es pasado al método *Connector.open()* determina la conexión que será abierta. Las siguientes URL y tipos de conexión de mensaje son soportados por el WMA:

- La URL sms://+18005555555 específica una conexión para enviar mensajes SMS a el número telefónico 1-800-555-5555 (Note que el WMA no tiene un requisito de formato de número telefónico; se puede usar cualquier serie de dígitos de el teléfono y la red reconocerán)
- La URL sms://+18005555555:1234 específica una conexión para enviar mensajes SMS a el puerto número 1234 del número telefónico 1-800-555-5555.
- La URL sms://:1234 específica una conexión a servidor para recibir mensajes sobre el puerto 1234. Una conexión de servidor puede también enviar mensajes.

El "MessageConnection" puede ser abierto ya sea en modo servidor o cliente. Una conexión de servidor es abierta proporcionando una URL que especifica el identificador (número de puerto) para una aplicación en el dispositivo local para mensajes entrantes dirigidos. La especificación solicita que *notifyIncomingMessage() retorne rápidamente.* Los mensajes recibidos con este identificador serán entregados a la aplicación por esta conexión. **Una conexión de servidor puede ser usada para enviar y recibir mensajes**

(MessageConnection) Connector.open("sms://:6000");

Una conexión modo cliente es abierta proporcionando una URL la cual apunta a otro dispositivo. **Una conexión modo cliente solamente puede ser utilizada para enviar mensajes.**

(MessageConnection) Connector.open("sms://+441234567890:600");

4.3.3 Números de Puerto SMS [8]

Cuando un número de puerto esté presente en una dirección, la TP-Usuario-Dato del SMS contendrá un Usuario-Dato-Encabezado con el puerto de aplicación direccionando el esquema de elemento de información. Cuando el destinatario de dirección no contiene un número de puerto, el TP-Usuario-Dato no contendrá el encabezado de direccionamiento de puerto de la aplicación. El MIDlet de J2ME no puede recibir este tipo de mensaje pero el SMS será manipulado de la manera usual por un estándar SMS del dispositivo.

Cuando un mensaje identificando un número de puerto es enviado desde un *MessageConnection* tipo servidor, el número de puerto originado en el mensaje es establecido en el número de puerto del *MessageConnection*. Esto permite al destinatario mandar una respuesta al mensaje que será recibido por esta *MessageConnection*.

Sin embargo, cuando un *MessageConnection* tipo cliente es utilizado para mandar un mensaje con un número de puerto, el número de puerto generado es establecido a un valor de implementación especifica y cualquier posible mensaje recibido para este número de puerto no es entregado al MessageConnection.[38]

Cuando un MIDlet en modo servidor pide un número de puerto (identificador) para usar y es el primer MIDlet en pedir este identificador, este será asignado. Si otras aplicaciones requieren mismo identificador entonces una *IOException* será desplegada cuando un intento de abrir la *MessageConnection* sea realizado. Si una aplicación del sistema está utilizando este identificador, el MIDlet no asignará el identificador. Los números de puertos permitidos para las peticiones están restringidos para los SMS. En conclusión, a un MIDlet no le esta permitido enviar

[38] Referirse a la sección A.4.0 y A.6.0 del JSR 120 en anexo A

mensajes a ciertos puertos restringidos, si el intento es realizado se desplegara un *SecurityException*.[39]

4.3.4 Almacenar y Borrar Mensajes Recibidos SMS [8]

Cuando los mensajes SMS son recibidos por un MIDlet, estos son removidos del SIM de memoria donde estos son almacenados. La localización del almacenamiento (inbox) para los mensajes SMS tiene una capacidad de más de 30 mensajes. Cualquier mensaje mayor de cinco días de antigüedad será removido del SIM, de manera parecida al *FIFO*[40] *stack*.

4.3.5 Tipos de Mensajes SMS [8]

Los tipos de mensajes que pueden ser enviados son de tipo TEXT o BINARY, los métodos de codificación de los mensajes están definidos en el estándar GSM 03.38.[41]

4.4 Especificaciones en el WMA dentro del Motorola V600 [8]

4.4.1 Estructura de Mensaje SMS

Debido a que J2ME adopta un estándar, de la misma forma el WMA lo hace para todos los dispositivos, por lo tanto no importando el tipo de teléfono móvil se cumple con lo antes mencionado.

La estructura de mensajes SMS cumplirá con el Sistema de Telecomunicaciones Digital Celular GSM 0.40 v7.4.0; la realización técnica del Servicio de Mensajes Cortos (SMS) ETSI 2000

La implementación de Motorola utiliza la concatenación especificada en secciones 9.2.324.1 y 9.2.3.24.8 del Estándar GSM 03.40 para mensajes que la

[39] Referirse a la sección A.6.0 del JSR, anexo A
[40] First In, First Out
[41] Referirse a la sección A.5.0 del JSR, anexo A

aplicación JAVA envía, que son demasiado extensos para encajar en un único protocolo de mensaje SMS.

Esta implementación automáticamente concatena los mensajes de protocolo SMS recibidos y transfiere el mensaje completamente reensamblado a la aplicación vía API. La implementación soportará al menos tres mensajes SMS para ser recibidos y concatenados juntos. Además, para el envío, se posee un soporte de un mínimo de tres mensajes. Motorola notifica a los desarrolladores que no se deben enviar mensajes que tomen mas de tres mensajes protocolo SMS a menos que el dispositivo receptor pueda soportar más.

4.4.2 Notificación SMS

Ejemplos de interacción de SMS con MIDlets.

- Un MIDlet manejará un SMS entrante si el MIDlet esta registrado para recibir mensajes en el puerto (identificador) y se está corriendo.
- Cuando un MIDlet es pausado y está registrado para recibir mensajes en el número de puerto de mensajes entrantes, entonces al usuario se le preguntará si se despliega el MIDlet.
- Si el MIDlet no está corriendo y la Maquina Virtual Java no esta inicializada, entonces un registro *Push* será utilizado para inicializar la Maquina Virtual y desplegar el MIDlet J2ME. Esto solamente aplica para el MIDlet, autenticado y registrado.
- Si un mensaje es recibido, y la aplicación de autenticación, registro y el KVM no están corriendo entonces el mensaje será desechado.
- Existe un Acceso a Configuración SMS en el menú de opción de Configuración Java que permite al usuario especificar cuando y que tan frecuentemente preguntar por autorización. Antes de establecer la conexión desde el MIDlet, las opciones disponibles son:
  - ✓ Siempre preguntar por autorización
  - ✓ Preguntar una vez por aplicación
  - ✓ Nunca preguntar

4.4.3 Característica de mensajería en el V600

La siguiente es una lista de características de mensajería y soporte de clases en el dispositivo:

Característica / clase	**Implementación**
JSR-120 API Específicamente, API's definidos en el paquete javax.wireless.messaging serán implementadas para el Adaptador SMS GSM	Sí
Remover mensajes SMS	Sí
Remover SMS recibidos	Sí
Remover SMS originados	Sí
Todos los campos, métodos, y métodos inherentes para la Clase *Connector* están en el paquete javax.wireless.messaging.	Si
Todos los métodos para la interface de mensaje binario están en el paquete javax.wireless.messaging	Si
Todos los métodos para la interface de mensaje están en el paquete javax.wireless.messaging.	Si
Todos los campos, métodos, y métodos inherentes para la interface *MessageConnection* están en el paquete javax.wireless.messaging.	Si
Número de instancias para *MessageConnection* en el paquete javax.wireless.messaging	32 máximo
Número de instancias para *MessageConnection* en el paquete javax.wireless.messaging.	16
Todos los métodos para la interface *MessageListener* en el paquete javax.wireless.messaging.	Si
Todos los métodos y métodos inherentes para la interface *TextMessage* están en el paquete javax.wireless.messaging	Si
Número de referencia de 16 bit en mensajes concatenados	Si

Número de mensajes concatenados	30 mensajes en bandeja de entrada, cada uno puede ser concatenado de tres partes. Sin limitación en bandeja de salida (inmediatamente transmitidos)
Permitir a los MIDlets obtener la dirección SMSC con la propiedad de sistema wireless.messaging.sms.smsc	Sí

4.4.4 Ejemplo de empleo de métodos en el WMA

Estos son ejemplos del empleo de métodos dentro de la WMA:

Creación de una conexión servidor

```
MessageConnection messageConnection =
(MessageConnection)Connector.open("sms://:9532");
```

Creación de una conexión cliente con número de Puerto

```
MessageConnection messageConnection =
(MessageConnection)Connector.open("sms://+18473297274:9532");
```

Creación de una conexión cliente sin número de puerto

```
MessageConnection messageConnection =
(MessageConnection)Connector.open("sms://+18473297274");
```

Cierre de conexión

```
MessageConnection messageConnection.close();
```

Creación de un mensaje SMS

```
Message textMessage =
messageConnection.newMessage(MessageConnection.TEXT_MESSAGE);
```

Configuración de carga de texto para un mensaje de texto

```
((TextMessage)message).setPayloadText("Text Message");
```

Obtener una carga de texto de un mensaje de texto recibido

```
receivedText = ((TextMessage)receivedMessage).getPayloadText();
```

Obtener una carga de datos de un mensaje binario recibido

```
BinaryMessage binMsg;
byte[] payloadData = binMsg.getPayloadData();
```

Configuración de dirección con número de puerto

```
message.setAddress("sms://+18473297274:9532");
```

Configuración de dirección sin número de puerto

```
message.setAddress("sms://+18473297274");
```

Envío de mensaje

```
messageConnection.send(message);
```

Recepción de mensaje

```
Message receivedMessage = messageConnection.receive();
```

Obtener una dirección

```
String address = ((TextMessage)message).getAddress();
```

Obtener dirección de centro de servicio SMS por medio de llamada de System.getProperty():

```
String addrSMSC = System.getProperty("wireless.messaging.sms.smsc");
```

Obtener un tiempo de despliegue de mensaje

```
Message message;
System.out.println("Timestamp: " + message.getTimestamp().getTime());
```

Capítulo 5:
DISEÑO Y DESARROLLO DE LAS APLICACIONES

5. DISEÑO Y DESARROLLO DE LAS APLICACIONES.

5.1 Descripción del capítulo

Es a partir de este capítulo en donde se separa del marco teórico y se enfoca al procedimiento del diseño e implementación del proyecto.

Este capítulo permite constatar la realización del Sistema de Bloqueo y Desbloqueo de Dispositivos empleando tecnología GSM. Todo ello por medio del diagrama en bloque del sistema con su debida explicación, los diagramas de flujo de cada uno de los programas empleados en los distintos dispositivos, contendrá además la codificación de cada uno de los programas con respecto a los distintos lenguajes de programación empleado (J2ME y ensamblador) y esquemas eléctricos, cada uno de ello con su explicación para facilitar la compresión de los mismos.

5.2 Diagrama en bloque del sistema a emplear

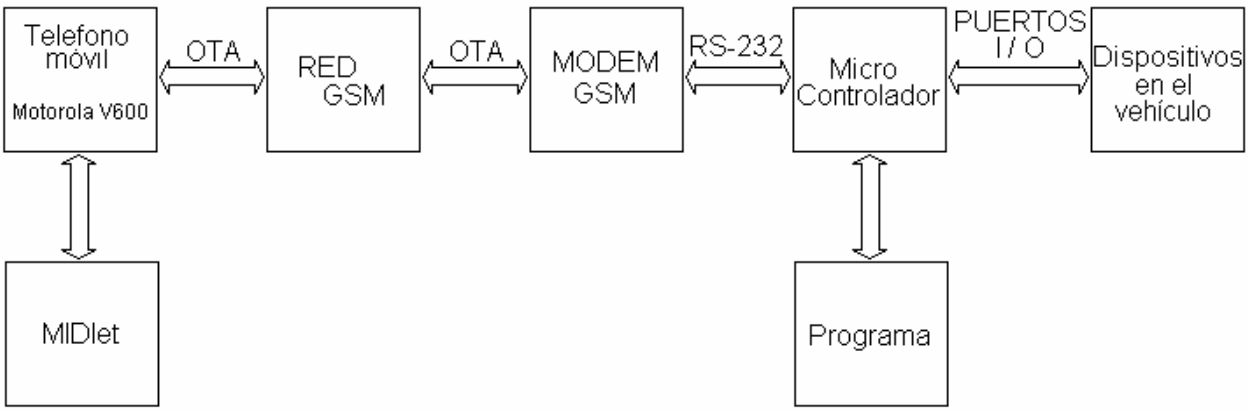


Figura 5.1 Diagrama en bloque del sistema

5.2.1 El MIDlet y el teléfono móvil Motorola V600

El teléfono móvil empleado es el motorola V600[42], dicho teléfono posee una versatilidad para con los midlets.

El midlet a realizar debe ser un interfaz gráfico de la aplicación: Sistema de activación y desactivación de dispositivos (SADD) empleando tecnología GSM, brindando un fácil manejo del mismo. Dicha tecnología a la que se refiere es el servicio adicional de toda operadora GSM, el SMS, para lo cual, se vale de la elaboración de un midlet para brindar al usuario de la aplicación un entorno similar al entorno gráfico de todos los menús, aplicaciones o funciones del Motorola V600.

El objetivo de la elaboración del midlet no es solamente el envío de mensajes, porque de lo contrario se emplearía solamente la función de mensajería del teléfono móvil, sino también, es brindar una seguridad del manejo de los mismos, y confirmar que solamente el usuario final de la aplicación podrá optar por las funciones del midlet.

Se ha previsto que las funciones mínimas del SADD serán las siguientes:

- Asignar número: la cual permitirá modificar el número telefónico del emisor y receptor (el Modem GSM), en casos de cambio de SIM dentro del teléfono móvil o de Modem o para envíos de prueba a cualquier teléfono móvil. Aunque no se seleccione esta función, la aplicación posee números por defecto.
- Password: esta función permitirá al usuario modificar su contraseña, para restringir a otras personas la activación y desactivación de dispositivos de manera no apropiada.
- Leer estado: permite al usuario solicitar al Modem que le envíe una notificación del estado de los dispositivos.

[1] Referirse al Anexo B para ver sus especificaciones técnicas

- Sistema: dicha función autoriza al usuario activar o desactivar el sistema en el vehículo. Al estar activo se podrá recibir notificaciones por parte del modem con respecto a un cambio en el estado de los dispositivos, y principalmente, permite el empleo de la siguiente función, la función de bloqueo y desbloqueo del funcionamiento del vehículo. Al estar desactivado, no se recibirán notificaciones y las funciones de bloqueo/desbloqueo del funcionamiento del vehículo será inválido.
- Funcionamiento del vehículo: la cual autoriza al usuario la activación y desactivación de los dispositivos encargados de bloquear y desbloquear el funcionamiento del vehículo.

Nota:
Para acceder y que se lleve a cabo cada una las funciones de SADD es necesario introducir el password actual para que sea verificado.
Todas las comunicaciones entre el teléfono móvil y el modem serán por medio de SMS, y contendrán el código de la función seleccionada y el password.

5.2.2 El Modem GSM y el Microcontrolador

El sistema SADD también consta de dos etapas las cuales son las encargadas de controlar y monitorear dispositivos remotamente y enviar información referente a éstos dispositivos al teléfono móvil del usuario; estas etapas son el MODEM GSM y la etapa del Circuito Microcontrolador.

El MODEM GSM es un equipo de comunicación de datos o DCE[43] el cual utiliza la red Celular GSM como medio de transmisión, éste se encarga de establecer un canal de comunicación entre el teléfono móvil del usuario y el circuito microcontrolador de manera que se puedan enviar comandos e instrucciones del usuario al circuito controlador para que éste ejecute las operaciones necesarias que satisfagan las peticiones realizadas. El MODEM GSM puede ejecutar todas las funciones que un Móvil GSM posea, se podría

[43] Data Communication Equipment

decir que en principios el MODEM es un Teléfono Móvil, pero a decir verdad esto no es del todo correcto, la verdadera similitud radica en que un Teléfono Móvil posee en su interior un MODEM, es decir el Móvil es en si un MODEM con periféricos agregados como LCD, memorias, programas, etc. que lo hacen un sistema mucho mas complejo que el simple MODEM. Estos periféricos con los que el Móvil cuenta hacen que este sea independiente de cualquier otro dispositivo para realizar su funcionamiento, en cambio la forma de representación de los datos en el MODEM es por medio de la comunicación de éste con un equipo que gobierne y entienda los comandos e instrucciones que este genera.

Los Comandos con los que todos los MODEM, ya sean estos inalámbricos o no, con los que se controlan son conocidos como Comandos AT, los cuales consisten en un *set* de instrucciones que los MODEM interpretan para realizar y ejecutar acciones como generar una llamada de voz, enviar mensajes de texto, leer mensajes de texto, etc. esto significa que el MODEM necesita de un equipo que le proporcione estos comandos para que este pueda funcionar, este Equipo Terminal de Datos o DTE[44] está representado en el sistema SADD por la etapa del Circuito Microcontrolador; esta etapa se encarga de generar e interpretar los comandos AT proporcionados por el MODEM, procesarlas y determinar las acciones que el MODEM debe de ejecutar para luego enviarle la secuencia de comandos AT necesaria, por ejemplo enviar un mensaje de texto al usuario que le notifique el estado de los dispositivos.

Además esta etapa es la encargada de bloquear y desbloquear físicamente los dispositivos así como también de hacer un monitoreo continuo de los sensores y cuando se detecte un cambio en éstos enviar los comandos AT al MODEM de manera que éste le envíe un mensaje de texto al móvil del usuario.

[44] Data Terminal Equipment

La comunicación entre del MODEM y el Circuito Microcontrolador se realiza por medio del estándar RS232 con las configuraciones siguientes:

- ✓ 8 bits de Datos
- ✓ No paridad
- ✓ 1 bit de Stop
- ✓ Transferencia de datos de 9600 bauds
- ✓ Control de Flujo por Hardware. Esta configuración debe estar activa tanto en el MODEM como en el microcontrolador para que se pueda establecer la comunicación.

El diseño del DTE se ha hecho en base a un par Microcontroladores PIC16F877A de manufactura de Microchip, en el cual se aprovecha el puerto comunicación Serial USART[45] que posee para la transmisión y recepción de los comandos AT que se necesitan para generar una comunicación fluida entre el Microcontrolador y el MODEM. Para manejar los niveles de voltajes que el Estándar RS232 necesita se utilizan un *Driver* para RS232 , el circuito integrado ST75185 realiza esta función, se encarga de convertir los niveles TTL[46] que utiliza el microcontrolador a niveles +12 V y –12V que son los que se utilizan en el estándar.

Las dos etapas el MODEM y el Microntrolador son etapas con funcionamiento independientes, pero para la funcionalidad del sistema SADD es imperativo la convergencia de ambas etapas, que éstas se comuniquen de una manera confiable, ya que estas dos etapas son las encargadas de ejecutar las peticiones que el usuario genere desde su teléfono móvil.

5.2.3 Los dispositivos en el vehículo

La comunicación con los dispositivos analógicos del vehículo se realizará a través de los puertos de entrada y salida del PIC, es decir se elegirá un pin del

[45] Universal Synchronous Asynchronous Receiver Transmitter
[46] Transistor Transistor Logic

puerto como salida para realizar la desactivación del vehículo cuando éste sea encendido sin consentimiento del usuario y el sistema general este activo, y se asignaran otro pines de entrada para la detección de dicho evento, y para los eventos del cierre central del vehículo.

Las funciones que se realizaran con el sistema son las siguientes:

- Notificación de estados del vehículo (automático)
- Notificación de estados del vehículo por solicitud
- Bloqueo del funcionamiento del vehículo.

5.2.3.1 Notificación de estado (automático)

Una vez que el sistema general este activo cualquier evento que se produzca en el vehículo, será notificado al PIC y este a la vez se comunicará con el Modem GSM para realizar la notificación al usuario por medio del terminal móvil, para esto se tomarán las líneas del sistema de cierre central del vehículo las cuales estarán interconectadas con él los pines de entrada del pic, de la misma manera estará interconectada la línea conmutada de encendido de vehículo, para notificar cuando se encienda el vehículo sin consentimiento. A continuación se presenta un diagrama de interconexión simple para dicha función.

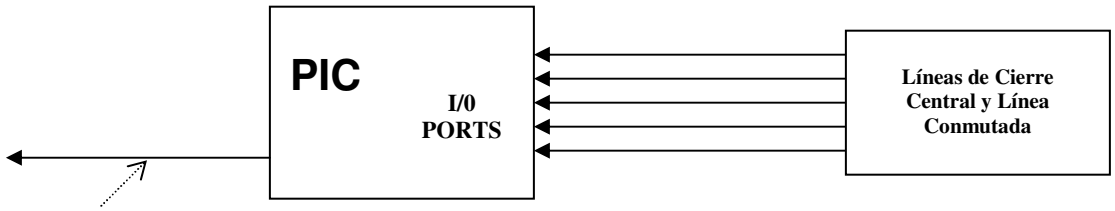


Figura 5.2 Interconexión entre dispositivos en el vehículo y el PIC para solicitud automática

5.2.3.2 Notificación de estado del vehículo por solicitud

En este caso la notificación de los estados del vehículo se hace a solicitud del usuario, es decir, este pide información del estado desde el

teléfono móvil, y el sistema le devuelve la información que este solicita, de esta manera la comunicación se realiza en ambos sentidos tanto de terminal móvil a vehículo y viceversa. El diagrama para realizar dicha función es similar al utilizado para la notificación de estados de vehículo automático.

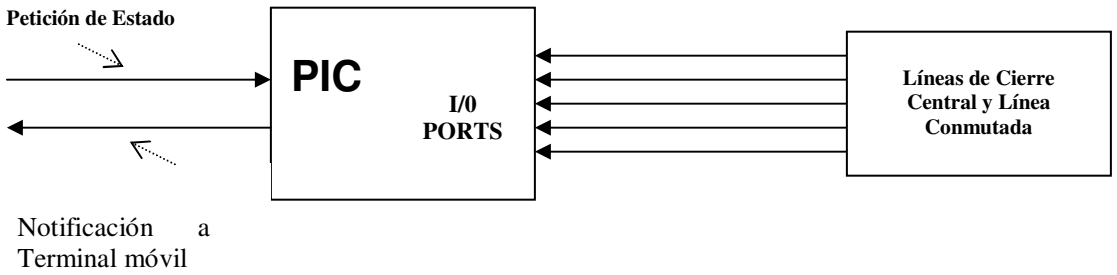


Figura 5.3 Interconexión entre dispositivos en el vehículo y el PIC para solicitud forzada

5.2.3.3 Bloqueo del funcionamiento del vehículo

En el bloqueo del funcionamiento del vehículo, es decir cuando este está encendido y se desea apagar, se hace a través del corte de flujo de corriente de los inyectores a tierra, es decir se dejan los inyectores como circuito abierto con respecto a tierra, lo cual impedirá que el vehículo siga encendido, para realizar dicha función, el móvil se encargará de enviar la petición de bloqueo, el módem recibe dicha petición y se comunica con el PIC, el cual por medio del pin de salida del puerto asignado, activará un relé el cual dejará el circuito de los inyectores como circuito abierto y el automóvil detendrá su marcha. A continuación se presenta un diagrama de interconexión simple para realizar dicha función.

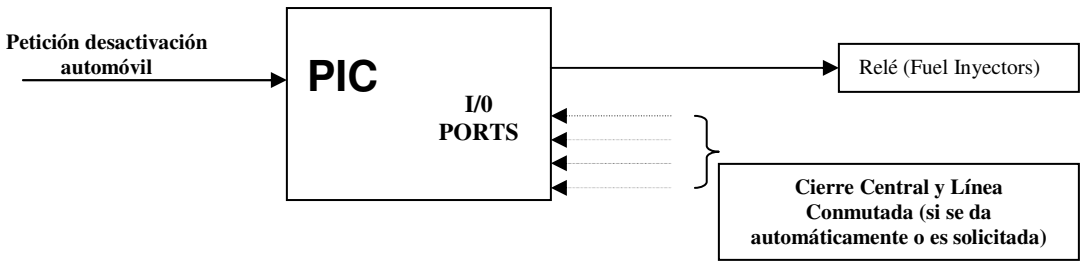


Figura 5.4 Interconexión entre dispositivos en el vehículo y el PIC para solicitud de bloqueo de funcionamiento

5.3 Diseño del MIDlet: Flujogramas

Para simplificar la compresión del flujograma, éste será dividido en subgrupos los cuales se mostrarán posteriormente, estos son:

- Flujograma principal
- Flujograma de configuración
- Flujograma de subgrupo “Sistema”
- Flujograma de subgrupo “Funcionamiento del Vehículo”
- Flujograma de Solicitud de Estado
- Flujograma de proceso WMA de envío
- Flujograma de verificación de password

FLUJOGRAMA PRINCIPAL:

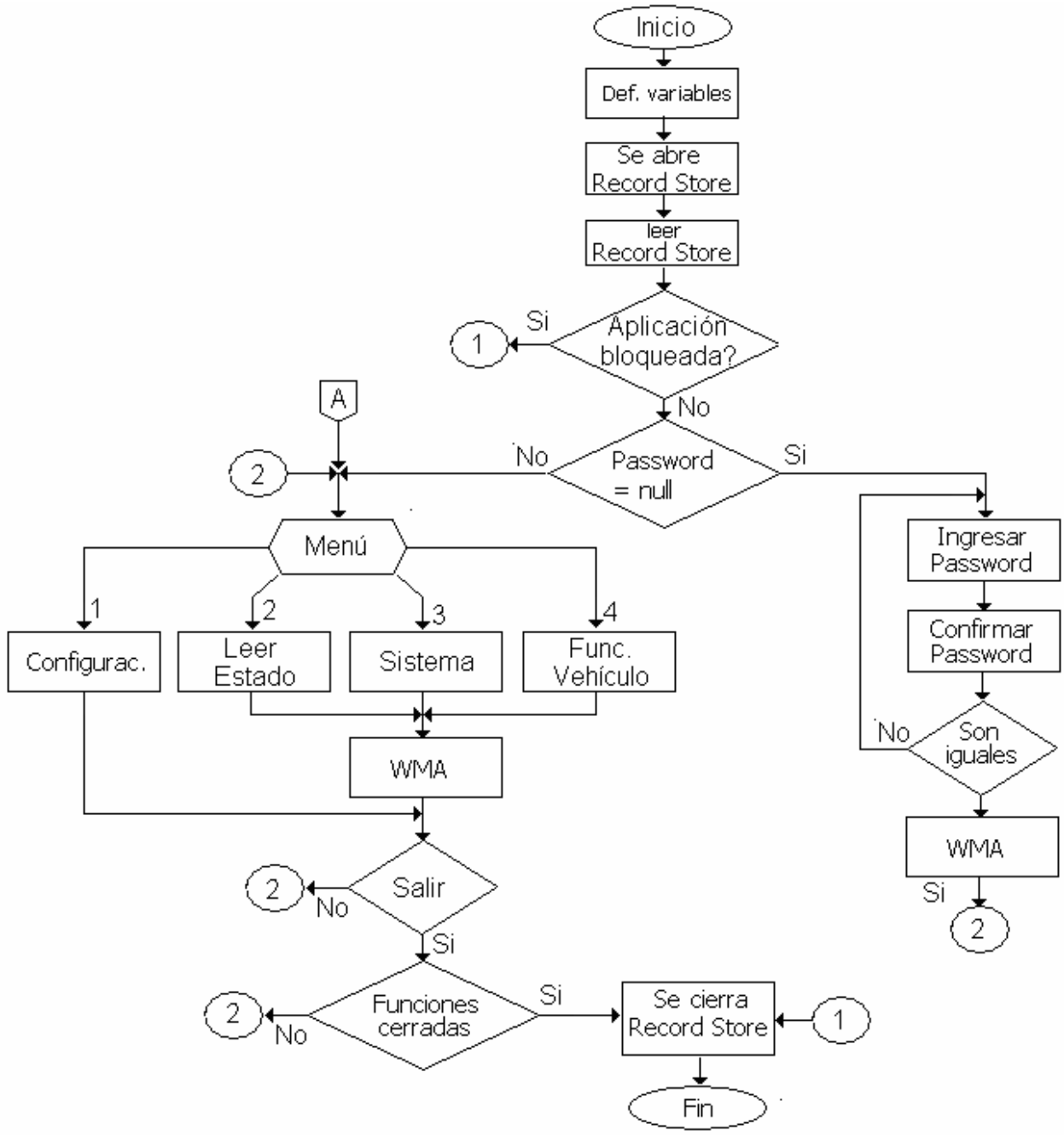


Figura 5.5 Flujograma principal del MIDlet de Activación y desactivación de dispositivos

FLUJOGRAMA DE CONFIGURACIÓN:

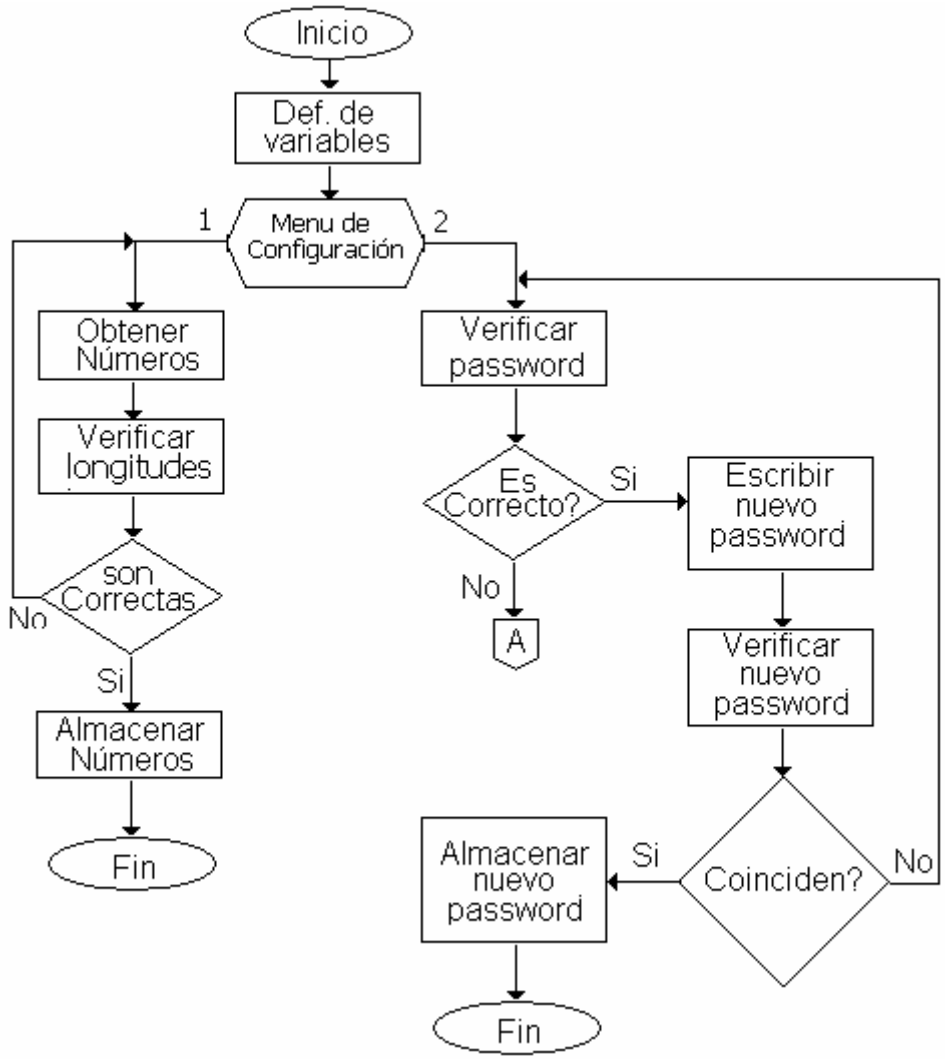


Figura 5.6 Flujogramas de Menú de Configuración

Flujograma de subgrupos “Sistema de activación y desactivación de dispositivos” y “Funcionamiento de vehículo”. Ambos procesos están basados en el mismo flujograma, la única variante es el código que se escribe en la variable *contenido*.

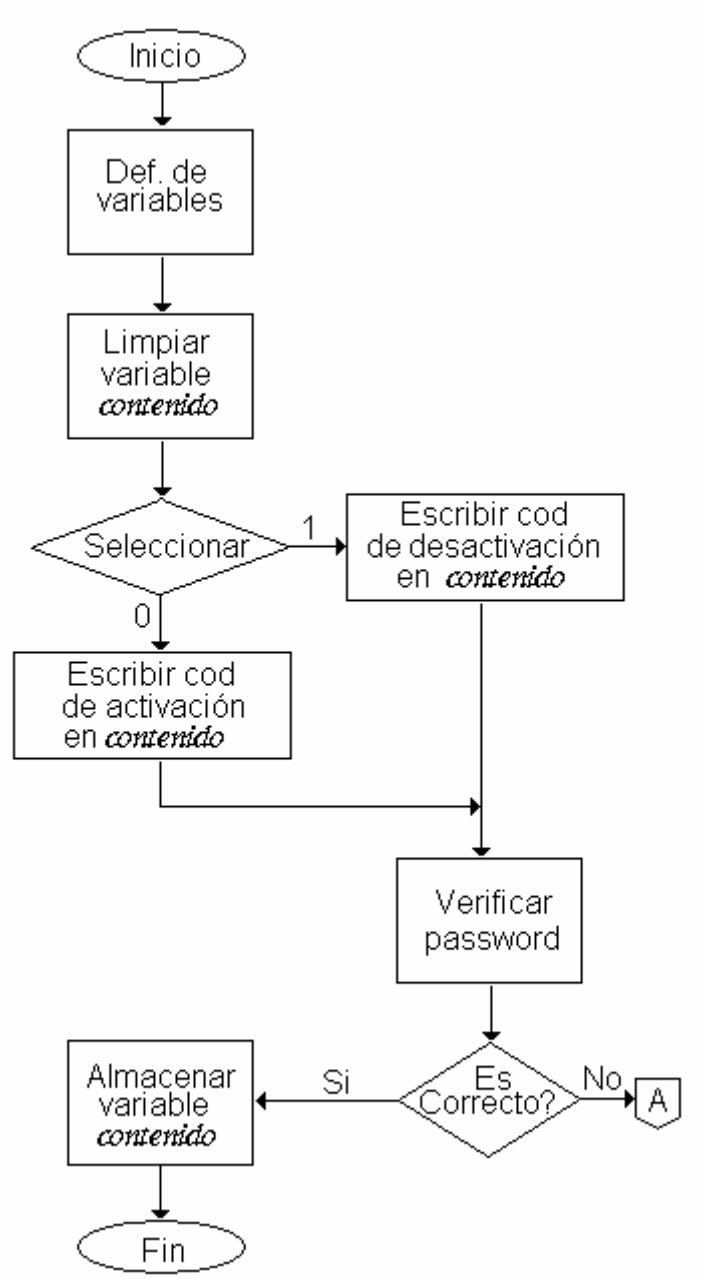
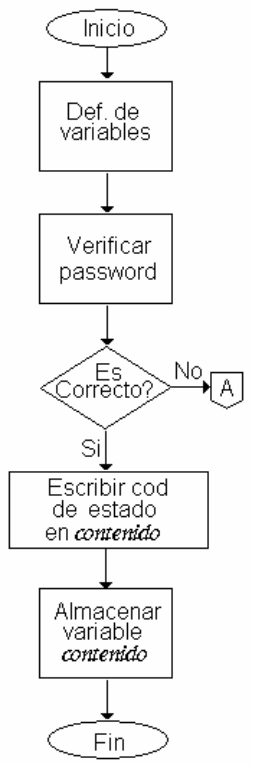
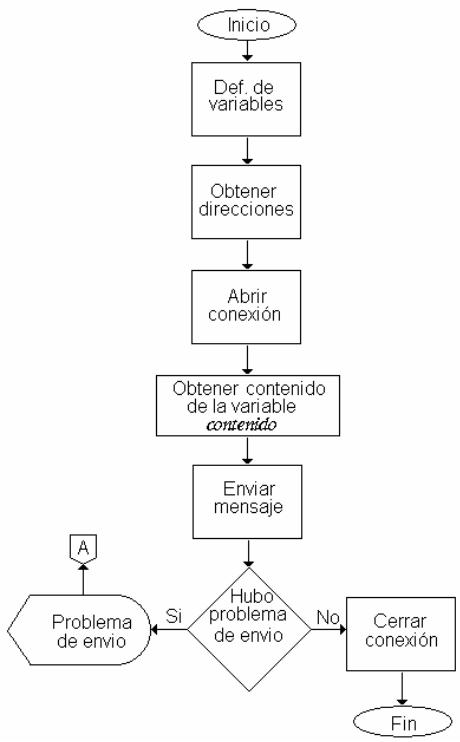


Figura 5.7 Flujograma de subgrupos "Sistema" y "Func. Vehículo"

Flujograma de solicitud de estado:



Flujograma del proceso WMA de envío:



Flujograma de verificación de Password

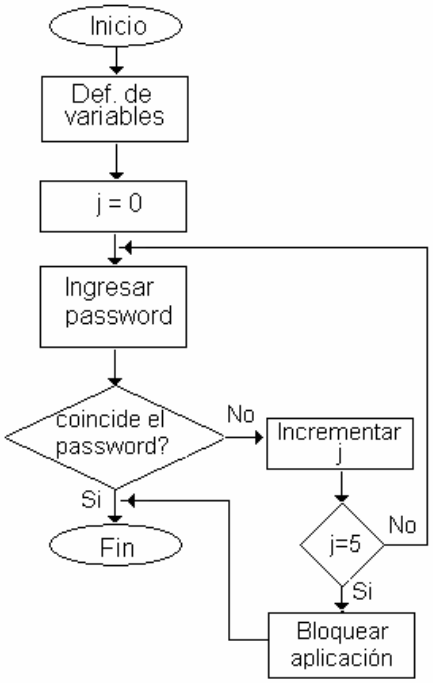


Figura 5.8 Flujogramas de subgrupos "Estado" , "WMA" y "Verif. de password"

5.4 Diseño de programa microcontrolador: flujogramas

Flujograma principal:

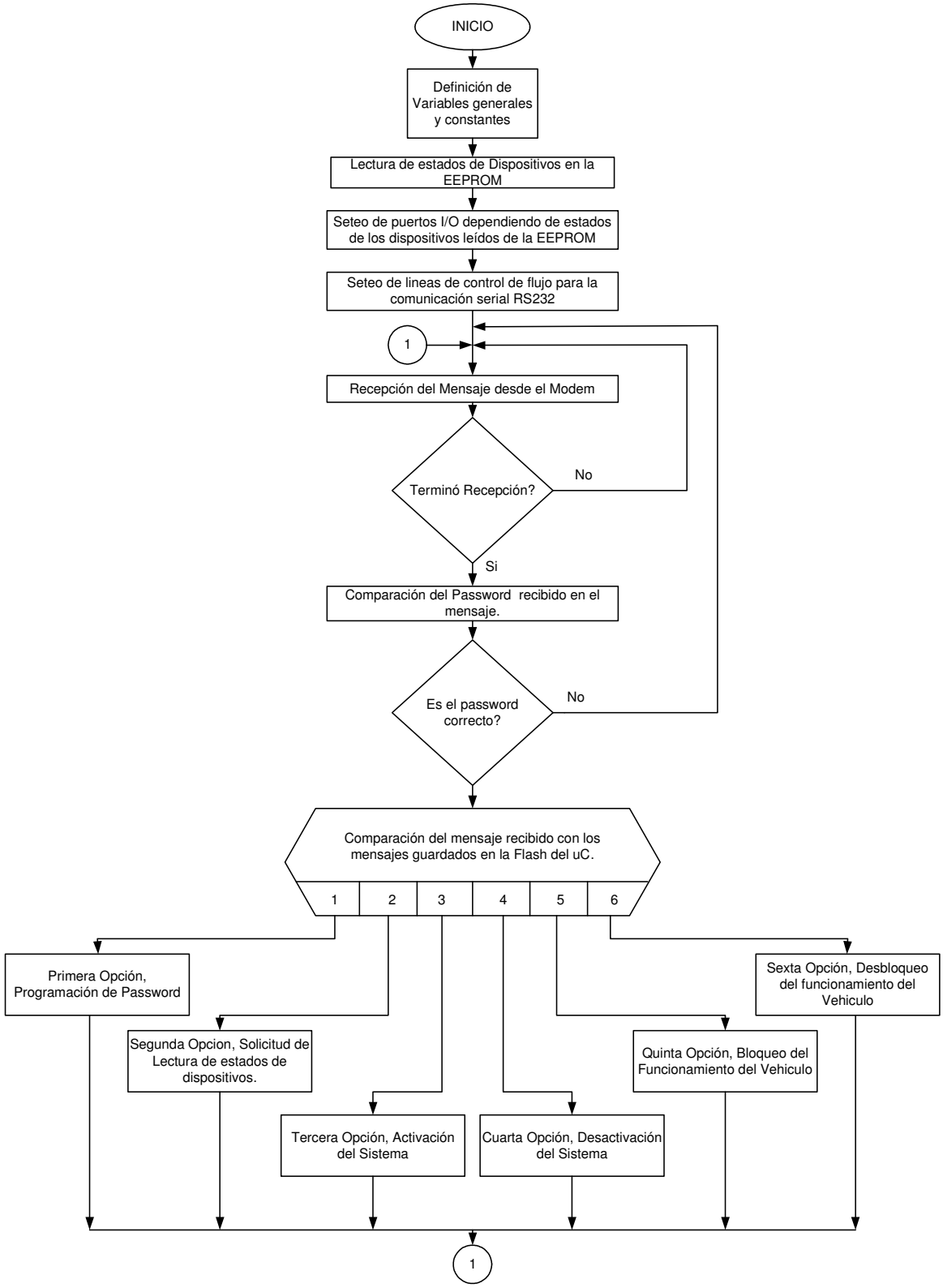


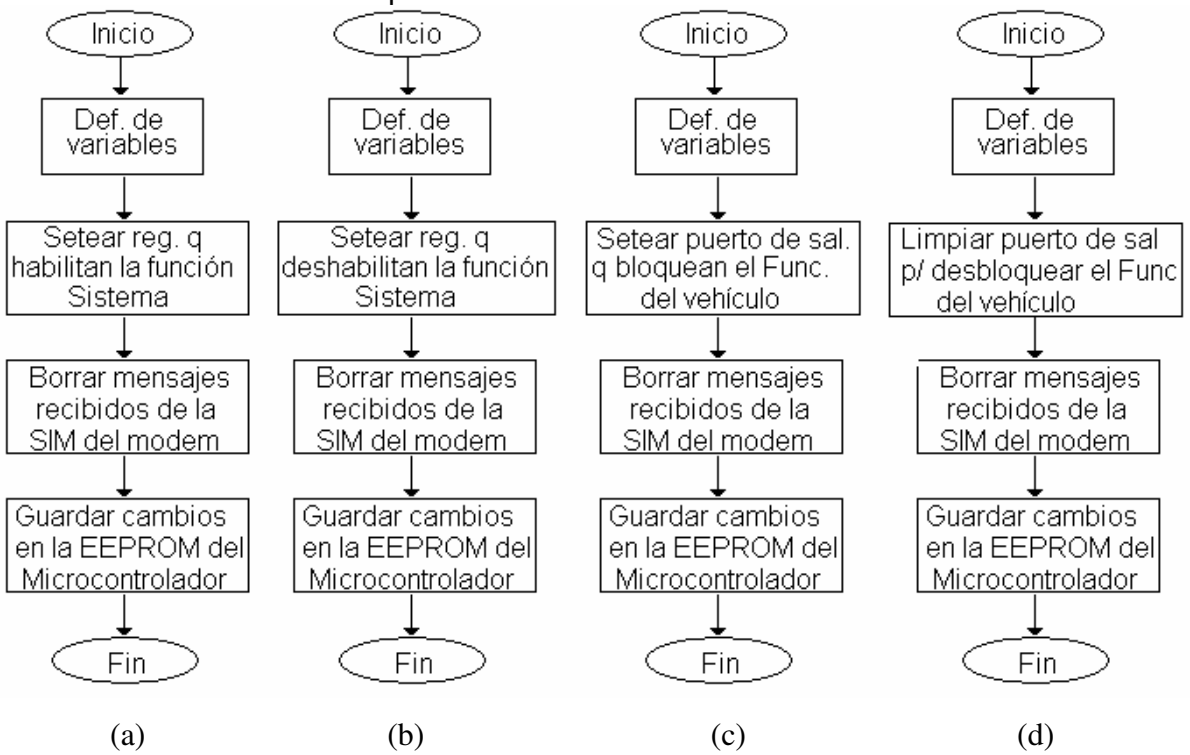
Figura 5.9: Flujograma principal de gestión entre el microcontrolador y el modem

El flujograma de la figura 5.6 representa la forma lógica de cómo el microcontrolador y el MODEM gestionarán la recepción de los mensajes tanto de notificaciones como el mismo contenido de los SMS que se recibirán en el MODEM desde el móvil del usuario. Lo primero se realiza es la inicialización de las variables y constantes para realizar las operaciones; luego se lee la memoria EEPROM del microcontrolador de manera que, cuando el sistema se active, el microntrolador seteé los puertos de salidas según como se encuentren en la memoria, esto se hace para evitar que por si en algún momento se desconecta la batería del vehículo y si el Microcontrolador quedase sin energía de alimentación esta al ser reconectada y el sistema se active de nuevo este inicie con los últimos valores con los que contaba de manera que si el vehículo tenia el funcionamiento bloqueado, al reiniciar de nuevo el sistema este vuelva a bloquear automáticamente el funcionamiento del vehículo.

En el flujograma también se encuentra especificado una etapa para la asignación de las líneas de control de flujo, de manera al iniciar la aplicación el control de flujo de habilite para la recepción de caracteres hacia el microcontrolador. Luego de realizada la inicialización de todas las variables y líneas correspondientes, la aplicación se queda a la espera que se reciban los caracteres desde el MODEM. Cuando se termina la recepción del mensaje se procede a la verificación del password que envía desde el móvil el usuario y el almacenado en la memoria del microcontrolador, de manera que si no coinciden entonces el microcontrolador reinicie sus registros y vuelva a esperar la recepción de otro mensaje.

Si el password fue el correcto entonces se procede a la determinación del la acción a realizar en función del contenido el mensaje de texto recibido, se tienen seis posibles opciones a realizar y cada una de ellas tiene un código único, si en algún momento el código recibido no coincide a ninguno de los códigos de las acciones a realizar el microcontrolador reiniciará sus registros y esperará de nuevo la recepción de otro mensaje.

Flujograma de procesos de activación y desactivación del sistema y de bloqueo y desbloqueo de funcionamiento del vehículo.



(a) (b) (c) (d)

Figura 5.10 (a) y (b) Flujograma de activación y desactivación del sistema. (c) y (d) Flujograma de bloqueo y desbloqueo del vehículo.

Los flujogramas anteriores muestran de manera simple como cuatro de las seis acciones a realizar son ejecutadas secuencialmente, después de haberse determinado que acción se debe realizar la aplicación determina, por medio del código único asignado a cada tarea, si es una activación o desactivación de las funciones del sistema SADD, o si bien es ejecutar el desbloqueo o el bloqueo del funcionamiento del vehículo. En estas tareas también se incluye el envío de los comandos AT hacia el MODEM que hará que se borre cualquier mensaje SMS que se haya recibido, esto se hace para evitar que en algún momento se llegue a llenar la memoria del SIM del MODEM o que se tengan tantos mensajes almacenados en el MODEM que hagan que el microcontrolador se confunda o ejecute acciones que se solicitaron y se ejecutaron antes.

Las otras dos opciones de acciones a realizar son las de cambio de password y el de solicitud de estado de dispositivos, las cuales sus flujogramas se detallan a continuación:

Flujograma de solicitud de cambio de password:

El siguiente flujograma muestra la secuencia que se sigue para que el microcontrolador una vez determinada la tarea correspondiente, lea el password que se recibió del mensaje de texto y se almacene de la memoria EEPROM del microcontrolador. Es de mencionar que el usuario desde la aplicación realizada para el teléfono móvil tiene que pasar la comprobación del password activo en ese momento para poder enviar el mensaje de texto al MODEM con la opción de cambiar password, esto por razones de seguridad, por lo tanto si el microcontrolador realiza esta acción es por el usuario previamente ha sido validad para realizar el cambio de password.

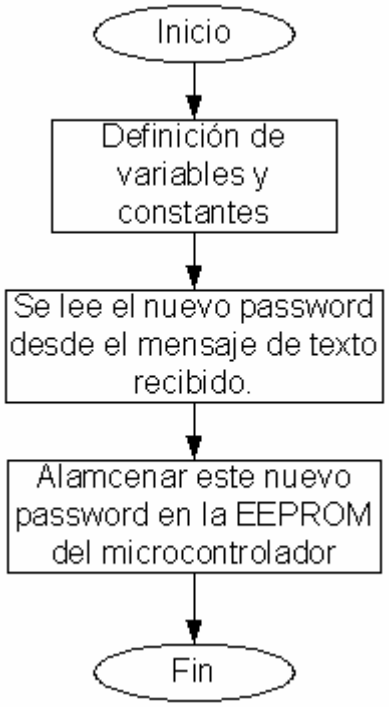


Figura 5.11 Flujograma de solicitud de cambio de password

Flujograma de solicitud de estados de dispositivos y sensores:

La figura siguiente muestra el flujograma correspondiente a como el microcontrolador verifica el estado de los dispositivos (dispositivo 1= sistema SADD; dispositivo 2=Bloqueo/desbloqueo del vehículo), teniendo cuatro posibles respuestas dependiendo de los diferentes estados en que pueden estar los dispositivos. Al final de la aplicación se realiza el envío del sms al usuario.

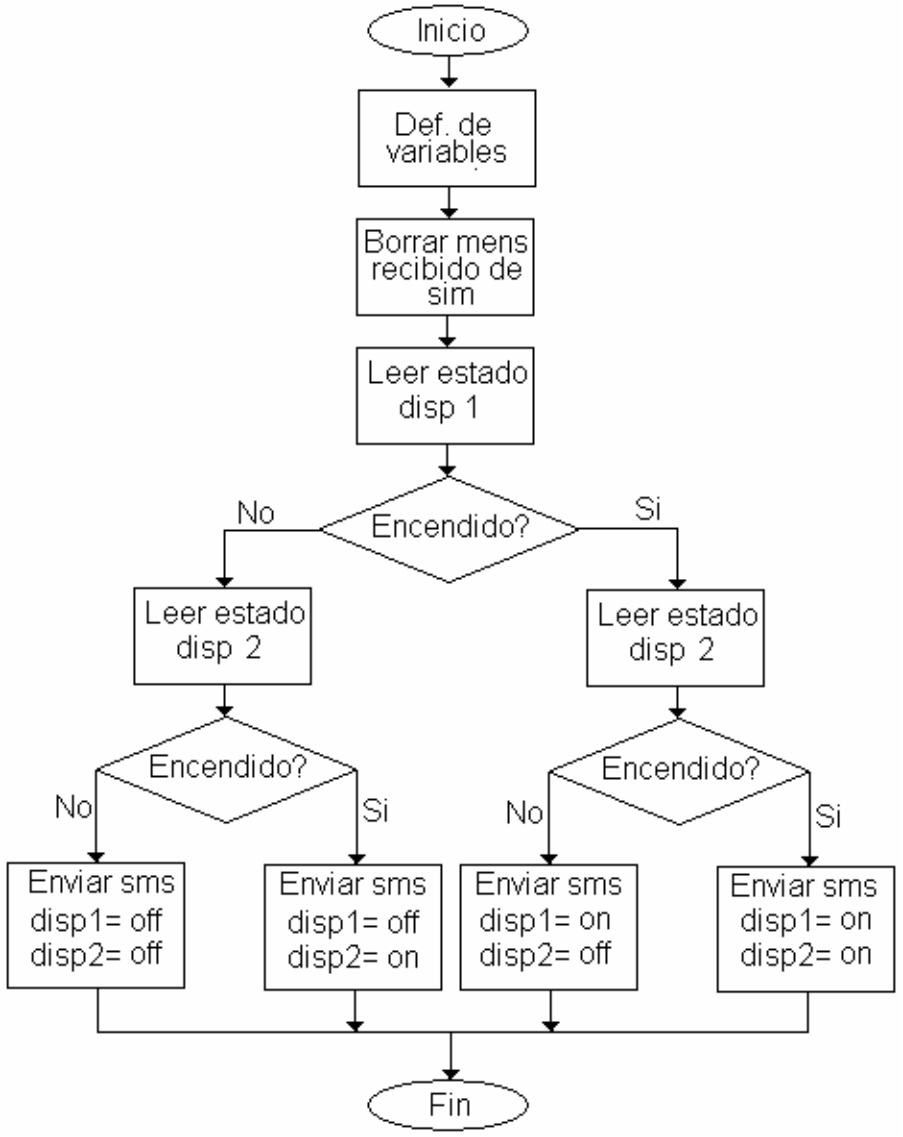


Figura 5.12: Flujograma de solicitud de estados en los dispositivos y sensores

El sistema una vez se encuentre activado, éste se encontrará a la expectativa de las interrupciones generadas por los sensores dentro del vehículo, por lo que al presentarse las interrupciones, se realizarán algunas respuestas y estos procedimientos se reflejan en el siguiente flujograma.

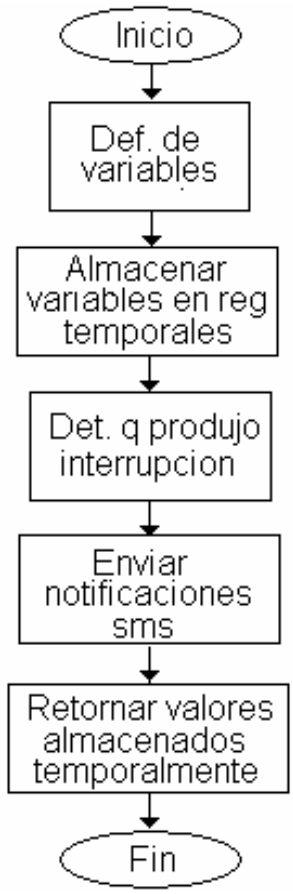


Figura 5.13: Flujograma de rutina de interrupción

5.5 Codificación de la aplicación SADD

Basados en los flujogramas del MIDlet, presentados en apartados anteriores, se ha realizado la codificación que se mostrará a continuación. Para el desarrollo de la misma, se ha empleado el programa Sun One Studio 4, Mobile Edition[47], que permite desarrollar, compilar y emular los programas de J2ME, dicha emulación se realiza en un

[47] Descargado de la página oficial de Sun Microsystems [9]

teléfono móvil general, por lo que, para emularlo en el teléfono V600 es necesario utilizar el programa: Motorola Launchpad for J2ME[48].

APLICACIÓN SADD:

```
/*
 * SADDMIDlet.java
 *
 * Aplicación que permite al usuario poder activar/desactivar el sistema de
 * notificación dentro de un vehículo, y al mismo tiempo y, como último recurso
 * bloquear/desbloquear el funcionamiento del vehículo.
 * Para ello se debe configurar el número telefónico del remitente y destinatario
 * , además de introducir el password respectivo.
 *
 * La comunicación entre el teléfono móvil y el modem en el vehículo es por medio
 * de OTA (over the air) y SMS
 *
 *
 Created on 22 de diciembre de 2005, 10:34 PM
 */


// Paquetes utilizados:

// Permite la definición de la aplicación
import javax.microedition.midlet.*;
// Permite la realización de la interfaz gráfica
import javax.microedition.lcdui.*;
// Permite la conexión genérica
import javax.microedition.io.*;
```

[48] Descargado de la página oficial de Motorola **[10]**

```
// Permite la interfaz de E/S básicos
import java.io.*;
// Utilizado para los SMS
import javax.wireless.messaging.*;
// Utilizado para el almacenamiento persistente en el dispositivo
import javax.microedition.rms.*;

/**
 *
 * author: Marvin Martinez
 * Version 1.0
 */


// Desarrollo de la Clase SADDMIDlet
public class SADDMIDlet extends javax.microedition.midlet.MIDlet implements
CommandListener, Runnable {

    /*
     *
     Definición de variables miembros de la clase (variables generales)
     *
     * Las variables son definidas de acuerdo a su pantalla correspondiente
     *
     */
    // Pantalla:Objeto display del MIDlet
    private Display pantalla;
    // Variable indice permite indicar la pantalla seleccionada
    int indice;
    // Variable indcom permite indicar que opcion de dispositivos se ha seleccionado
    int indcom;
```

```
    // Variable CONTENIDO posee el contenido del mensaje a enviar
    String contenido = null;
    /* Variable disp y func permite mostrar en pantalla la función seleccionada
       despues de introducir el password de verificación */
    String disp = null;
    String func = null;


    // Variable opciones0[] contiene los elementos del Menu
    String opciones0[] = {"Configuracion","Leer Estados","Sistema","Func. Vehículo"};
    // Menu principal
    private List menu = new List("Menu",List.IMPLICIT,opciones0,null);
    // Comando de selección de los elementos del Menu principal
    private Command seleccionar0 = new Command("SELECR",Command.OK,1);
    // Comando de salida del Menu principal
    private Command salir = new Command("SALIR",Command.EXIT,1);


    // Variable opciones1[] contiene los elementos del Menu de Configuración
    String opciones1[] = {"Prueba SMS","Password"};
    // Menu de Configuración
    private List menuConf = new List ("Configuracion de
SADD",List.IMPLICIT,opciones1,null);
    // Comando de selección del Menu de Configuración
    private Command seleccionar1 = new Command("SELECR",Command.OK,2);
    // Comando de salida del Menu de Configuración
    private Command atras0 = new Command("ATRAS",Command.BACK,2);


    // Forma llamNumodem es la pantalla de solicitud del número de remitente y
destinatario
```

```
    private Form llamNumodem = new Form("Números para prueba SMS");
    // Número de telefono movil del remitente
    String numcel = "78708041";
    // Número del modem destinatario
    String numodem = "79404287";
    // pedirNum0, campo de texto que solicita número del remitente
    private TextField pedirNum0 = new TextField("Introduzca número de remitente:
","",8,TextField.PHONENUMBER);
    // pedirNum1, campo de texto que solicita número del destinatario
    private TextField pedirNum1 = new TextField("Introduzca número de destino:
","",8,TextField.PHONENUMBER);
    // Comand de retorno al Menu de Configuración
    private Command atras1 = new Command("ATRAS",Command.BACK,1);
    // Comando de aceptación de números
    private Command okcmd0 = new Command("OK",Command.OK,1);
    // Pantalla que informa que se han aceptado los números
    private Alert alerNumodem;
    // Pantalla que informa que alguno de los números no está compuesto de 8 digitos
    private Alert alerNumal;


    // Forma llamPassw0 es la pantalla de solicitud del Password para luego cambiarlo
    private Form llamPassw0 = new Form("Password");
    // Variable password, contiene la contraseña del usuario, al inicio tiene un valor por
default
    String password = null;
    // pedirPw0, campo de texto que solicita la introduccion del password
    private TextField pedirPw0 = new TextField("Introduzca la
contraseña","",5,TextField.NUMERIC|TextField.PASSWORD);
    // Variable datpw0 guarda temporalmente el password
    String datpw0 = null;
```

```
    // Comando de cancelación y retorno al Menu principal
    private Command cancelar0 = new Command("CANCEL",Command.CANCEL,1);
    // Pantalla que informa que el password introducido es erróneo
    private Alert alerPw1;

    // Variable que define si la aplicación está  bloqueada o no
    private String setBloq = "n";
    // Variable que permite realizar el conteo de los intentos de ingreso de password
    int j=0;
    // Forma bloqPant es la pantalla de bloqueo de la aplicación SADD
    private Form bloqPant = new Form("");
    // Imagen de candado que indicará programa bloqueado
    private Image candado;
    // bloqDib permite situar el dibujo en cualquier parte de la pantalla ademas de
agregarle
    // texto
    private ImageItem bloqDib;

    // Forma llamPassw1 es la pantalla de solicitud del nuevo password
    private Form llamPassw1 = new Form("Password");
    // pedirPw1, campo de texto que solicita la introdución del nuevo password
    private TextField pedirPw1 = new TextField("Nueva contraseña
:","",5,TextField.NUMERIC|TextField.PASSWORD);
    // Variable datpw1 guarda temporalmente el nuevo password introducido
    String datpw1 = "";
    // Comando de aceptación del password introducido
    private Command okcmd1 = new Command("OK",Command.OK,1);
    // Pantalla que informa que la longitud del password es de 5 digitos
    private Alert longPassw;

    // Forma llamPassw2 es la pantalla de verificación del nuevo password
```

```
    private Form llamPassw2 = new Form("Password");
    // pedirPw2, campo de texto que solicita la introducción de la verificación del
password
    private TextField pedirPw2 = new TextField("Confirme contraseña :
","",5,TextField.NUMERIC|TextField.PASSWORD);
    // Variable datpw2 guarda temporalmente la verificación del nuevo password
    String datpw2 = "";
    // Pantalla que informa que la contraseña nueva es aceptada
    private Alert alerPw2;
    // Pantalla que informa que las contraseñas no coinciden
    private Alert alerPw3;
    // Comando de aceptación de la verificación del password
    private Command okcmd2 = new Command("OK",Command.OK,1);


    // Menu de selección de acción de la función SISTEMA
    private List llamDisp0;
    // Variable opcdis0[] contiene los elementos del menu Sistema
    String opcdis0[] = {"ACTIVAR","DESACTIVAR"};
    // Imagen de vehículo en miniatura
    private Image car;
    // Comando de selección de la acción del menu Sistema
    private Command selec0 = new Command("SELECR",Command.OK,1);
    // Variable booleana que indica que se ha solicitado la Activación
    private boolean permiso = false;
    // Pantalla que solicitud el Desbloque del vehículo antes de Desactivar el sistema
    private Alert pedirDesb;


    // Menu de selección de acción de la función FUNCIONAMIENTO DEL VEHÍCULO
    private List llamDisp1;
```

```
    // Variable opcdis1[] contiene los elementos del menu Funcionamiento del Vehículo
    String opcdis1[] = {"BLOQUEAR","DESBLOQUEAR"};
    // Imagen de vehículo bloqueado en miniatura
    public Image nocar;
    // Comando de selección de la acción del menu Funcionamiento de Vehículo
    private Command selec1 = new Command("SELECR",Command.OK,1);
    // Variable booleana que indica que se ha bloqueado el Funcionamiento del Vehículo
    private boolean bloqueo = false;
    // Pantalla de solicitud de Activación del Sistema antes de seleccinar Funcionamiento
de Vehículo
    private Alert pedirAct;


    // Forma llamVerpw es la pantalla de Verificación del password
    private Form llamVerpw = new Form("Verificar contraseña");
    // pedirVerpw, campo de texto de solicitud de contraseña para verificar
    private TextField pedirVerpw = new TextField("Verificar contraseña
:","",5,TextField.NUMERIC|TextField.PASSWORD);
    // Variable que almacena temporalmente la contraseña de verificación
    String datVerpw;
    // Comando de cancelación y retorno a Menu Principal
    private Command cancelar1 = new Command("CANCEL",Command.CANCEL,1);
    // Pantalla de aceptación de password y de inf de acción realizada
    private Alert alerVerpw0;


    // Forma llamExit es la pantalla de confirmación de salir de la aplicación
    private Form llamExit = new Form("Salir ?");
    // Variable verExit, pregunta de salida
    String verExit = "\n\n\nEsta seguro que desea salir?";
    // Comando que retorna al menu principal
```

```
    private Command regresar = new Command("No",Command.OK,1);
    // Comando que permite la salida de la aplicación
    private Command exit1 = new Command("Si",Command.EXIT,1);
    // Comando que permite la salida de la aplicación
    private Command exit2 = new Command("Salir",Command.EXIT,1);
    // Pantalla que informa que se  debe Desactivar el Sistema antes de salir
    private Alert negExit1;


    // Se define la sesión "guardar" que ayudará al almacenamiento persistente
    private RecordStore guardar;
    private RecordStore datBloq;
    // Variable longitud permite ver la longitud del registro de la sesión "guardar"
    private int longitud;
    // Pantalla de error que informa que necesita un password para autenticar las
funciones
    private Alert intPassw;


    /*
     *
     * Metodo que contiene la contrucción de cada una de las pantallas a utilizar
     * en todo la aplicación
     *
     */
    public SADDMIDlet(){
        // Se define la variable PANTALLA como la base de pantalla a desplegar
        pantalla = Display.getDisplay(this);

        // Se adiciona al Menu principal el comando salir
        menu.addCommand(salir);
```

```
        // Se adiciona al Menu principal el comando seleccionar0
        menu.addCommand(seleccionar0);
        // En el menu principal se entablece la captura de eventos de los comandos
        menu.setCommandListener(this);

        // Se adiciona al Menu de Configuración el camando atras0
        menuConf.addCommand(atras0);
        // Se adiciona al Menu de Configuración el comando seleccionar1
        menuConf.addCommand(seleccionar1);
        // En el menu de Configuración se establece la captura de eventos de los
comandos
        menuConf.setCommandListener(this);

        // En la forma llamNumodem se situa el campo de texto pedirNum0
        llamNumodem.append(pedirNum0);
        // En la forma llamNumodem se situal el campo de texto pedirNum1
        llamNumodem.append(pedirNum1);
        // Se adiciona a la forma llamNumodem el comando atras1
        llamNumodem.addCommand(atras1);
        // Se adiciona a la forma llamNumodem el comando okcmd0
        llamNumodem.addCommand(okcmd0);
        // En la forma llamNumodem se establece la captura de eventos de los comandos
        llamNumodem.setCommandListener(this);
        // Se define alerNumal
        alerNumal = new Alert("","Introduzca correctamente los
números",null,AlertType.ERROR);
        // Se especifica el tiempo de presentación en pantalla de aleNumal
        alerNumal.setTimeout(2000);

        // En la forma llamPassw0 se situa el campo de texto perdirPw0
        llamPassw0.append(pedirPw0);
```

```
        // Se adiciona a la forma llamPassw0 el comando cancelar
        llamPassw0.addCommand(cancelar0);
        // Se adiciona a la forma llamPassw0 el comando okcmd0
        llamPassw0.addCommand(okcmd0);
        // En la forma llamPassw0 se establecela captura de eventos de los comandos
        llamPassw0.setCommandListener(this);
        // Se define alerPw1
        alerPw1 = new Alert("","Contraseña incorrecta",null,AlertType.ERROR);
        // Se especifica el tiempo de presentación en pantalla de alerPw1
        alerPw1.setTimeout(2000);


        // Se crea la variable que contiene la imagen de bloqueo
        try{
            // Se define la ruta para habilitar la imagen
            candado = Image.createImage("/candado.PNG");
            // Se agrega texto y se determina la ubicación en la pantalla de la imagen
            bloqDib = new ImageItem("Programa
Bloqueado",candado,ImageItem.LAYOUT_CENTER,"");
            // Se coloca la imagen sobre la pantalla de bloqueo (a la forma)
            bloqPant.append(bloqDib);
        }
        catch(Exception E){
        }
        // Se adiciona a la forma bloqPant el comando exit2
        bloqPant.addCommand(exit2);
        // En la forma bloqPant se establece la captura de eventos de los comandos
        bloqPant.setCommandListener(this);


        // En la forma llamPassw1 se situa el campo de texto pedirPw1
```

```
        llamPassw1.append(pedirPw1);
        // Se adiciona a la forma llamPassw1 el comando cancelar0
        llamPassw1.addCommand(cancelar0);
        // Se adiciona a la forma llamPassw1 el comando okcmd1
        llamPassw1.addCommand(okcmd1);
        // Se establece la captura de eventos de los comandos en la forma llamPassw1
        llamPassw1.setCommandListener(this);
        // Se define longPassw
        longPassw = new Alert("","  La longitud del password debe ser de 5
digitos",null,AlertType.ERROR);
        // Se establece el tiempo de presentación de longPassw
        longPassw.setTimeout(3000);

        // En la forma llamPassw2 se situa el campo de texto pedirPw2
        llamPassw2.append(pedirPw2);
        // Se adiciona a la forma llamPassw2 el comando cancelar0
        llamPassw2.addCommand(cancelar0);
        // Se adiciona a la forma llamPassw2 el comando okcmd2
        llamPassw2.addCommand(okcmd2);
        // En la forma llamPassw2 se establece la captura de eventos de los comandos
        llamPassw2.setCommandListener(this);
        // Se define alerPw2
        alerPw2 = new Alert("","Contraseña aceptada",null,AlertType.CONFIRMATION);
        // Se establece el tiempo de presentación en pantalla de alerPw2
        alerPw2.setTimeout(2000);
        // Se define alerPw3
        alerPw3 = new Alert("","Escriba correctamente la
contraseña",null,AlertType.ERROR);
        // Se establece el tiempo de presentación en pantalla de alerPw3
        alerPw3.setTimeout(2000);
```

```
        // Se define la variable llamDisp0
        llamDisp0 = new List("Sistema",List.EXCLUSIVE,opcdis0,null);
        // Se setea como seleccionado la opción DESACTIVADO
        llamDisp0.setSelectedIndex(1,true);
        // Se adiciona en el Menu del Sistema el comando selec0
        llamDisp0.addCommand(selec0);
        // Se adiciona en el Menu del Sistema el comando atras0
        llamDisp0.addCommand(atras0);
        // Se establece la captura de eventos de los comandos en el Menu de Sistema
        llamDisp0.setCommandListener(this);
        // Se define pedirDesb
        pedirDesb = new Alert("","El funcionamiento del vehículo debe
desbloquearse",null,AlertType.INFO);
        // Se establece el tiempo de presentación en pantalla de pedirDesb
        pedirDesb.setTimeout(3000);

        //Se define la variable llamDisp1
        llamDisp1 = new List("Func. Vehículo",List.EXCLUSIVE,opcdis1,null);
        // Se setea como seleccionado la opción DESBLOQUEAR funcionamiento del
vehículo
        llamDisp1.setSelectedIndex(1,true);
        // Se adiciona al Menu Funcionamiento de Vehículo el comando selec1
        llamDisp1.addCommand(selec1);
        // Se adiciona al Menu Funcionamiento de Vehículo el comando atras0
        llamDisp1.addCommand(atras0);
        // Se establece la captura de eventos los comandos en el Menu Funcionamiento de
Vehículo
        llamDisp1.setCommandListener(this);
        // Se define pedirAct
        pedirAct = new Alert("","Debe activar el sistema",null,AlertType.INFO);
        // Se establece el tiempo de presentación en pantalla de pedirAct
```

```
        pedirAct.setTimeout(2000);

        // En la forma llamVerpw se establece el campo de texto pedirVerpw
        llamVerpw.append(pedirVerpw);
        // Se adiciona a la forma llamVerpw el comando cancelar1
        llamVerpw.addCommand(cancelar1);
        // Se adiciona a la forma llamVerpw el comando okcmd0
        llamVerpw.addCommand(okcmd0);
        // Se establece la captura de eventos de los comandos en la forma llamVerpw
        llamVerpw.setCommandListener(this);

        // Se establece en la forma llamExit el campo de texto verExit
        llamExit.append(verExit);
        // Se adiciona el comando regresar en la forma llamExit
        llamExit.addCommand(regresar);
        // Se adiciona el comando exit en la forma llamExit
        llamExit.addCommand(exit1);
        // Se establece la captura de eventos de los comandos en la forma llamExit
        llamExit.setCommandListener(this);
        // Se define negExit
        negExit1 = new Alert("","Debe desactivar el sistema antes de
salir",null,AlertType.INFO);
        // Se establece el tiempo de presentación en pantalla de pedirAct
        pedirAct.setTimeout(3000);


        // Se define el tipo de Alerta de intPassw
        intPassw = new Alert("","Debe introducir un password para poder autenticar el
funcionamiento",null,AlertType.ERROR);
        // Se establece el tiempo de presentación en pantalla de intPassw
        intPassw.setTimeout(4000);
```

```
    }

    /*
     *
     * Método starApp especifica la acción a realizar cada vez que el MIDlet
     * se situe en el estado Activo
     *
     */
    public void startApp() {
        // Se invoca al método para crear o abrir el registro de almacenamiento persistente
        abrirRecordStore();
        // Se invoca al método para leer el registro de almacenamiento persistente
        leerRecordStore();
        // Se lee la variable que indica si la aplicación esta bloqueada
        if (setBloq.equals("s")){
            // Si la aplicación se bloqueó en una rutina pasada, deberá mantenerse siempre
bloqueado
            pantalla.setCurrent(bloqPant);
        }
        // Aplicación no bloqueada
        else{
            // Si es primera vez que se utiliza el midlet, solicitará el ingreso de un password
            // para autenticar las funciones  de la aplicación
            if (password == null){
                // Se presenta la pantalla de ingreso de un nuevo password
                pantalla.setCurrent(llamPassw1);
            }
            // De lo contrario, si existe un password ya existente
            else{
            // Presenta el Menu principal
```

```
        pantalla.setCurrent(menu);
        }
    }
}

/*
 *
 * Método pauseApp especifica la acción a realizar cada vez que el MIDlet
 * se situe en el estado Pausa
 *
 */
public void pauseApp() {
    // Se invoca al método para cerrar el registro de almacenamiento persistente
    cerrarRecordStore();
}

/*
 *
 * Método destroyApp especifica la acción a realizar cada vez que el MIDlet
 * se situe en el estado Destruido
 *
 */
public void destroyApp(boolean unconditional) {
}

/*
 *
 * Método invVerpw especifica la acción a realizar cada vez que el usuario
 * necesite verificar el password para realizar alguna acción
 *
 */
```

```
public void invVerpw(){
    // Situa el indice de la pantalla de verificación de password
    indice = 9;
    // Presenta la pantalla de verificación de password
    pantalla.setCurrent(llamVerpw);
}

/*
 *
 * Método commandAction especifica las acciones a realizar cada vez se realicen
 * las capturas de eventos por parte de los comandos
 *
 */
public void commandAction(Command c, Displayable d){

    // En el Menu principal si se presiona el comando salir
    if (c == salir){
        // Se verifica que no este Activado el Sistema
        if (permiso == true){
            // Se muestra la pantalla que informa que debe Desactivarse el sistema
            pantalla.setCurrent(negExit1,menu);
        }
        // De lo contrario se pasa a la pantalla de confirmación de salida
        else{
        pantalla.setCurrent(llamExit);
        }
    }
    // Por lo que si se selecciona el comando seleccionar0 en el Menu principal
    else{
        // Se obtiene la función seleccionada dentro del Menu principal
        switch(menu.getSelectedIndex()){
```

```
// Se selecciona Configuración
case 0:{
   // Se muestra en pantalla el Menu de Configuración
   pantalla.setCurrent(menuConf);
   break;
}
// Se selecciona Leer Estados
case 1:{
   // Se guarda en las variables disp y func la información a mostrar
   // de las acciones realizadas
   disp = "";
   func = "Se ha solicitado estado de dispositivos";
   // Se almacena el indice de la función Leer Estados
   indice = 4;
   // Se le asigna a la variable contenido el codigo de la función
   contenido = "**5"+password;
   // Se invoca la verificación de la contraseña
   invVerpw();
   break;
}
// Se selecciona Sistema
case 2:{
   // Se presenta en pantalla el Menu Sistema
   pantalla.setCurrent(llamDisp0);
   break;
}
// Se selecciona Func. de Vehículo
case 3:{
   // Si se encuentra Activo el Sistema
   if (permiso == true){
      // Se accede al Menu de Funcionamiento de Vehículo
```

```
            pantalla.setCurrent(llamDisp1);
        }
        // De lo contrario se muestra en pantalla una petición de
        // activar el Sistema
        else{
            pantalla.setCurrent(pedirAct,menu);
        }
        break;}
    }
}

// En el Menu de Configuración si se presiona el comando seleccionar1
if (c == seleccionar1){
    // Se obtiene la función seleccionada
    switch(menuConf.getSelectedIndex()){
        // Se selecciona Asignar Número
        case 0:{
            // Se almacena el indice de la función Asignar Número
            indice = 0;
            // Se muestra la pantalla de introducción de lo números
            // del teléfono móvil y del modem
            pantalla.setCurrent(llamNumodem);
            break;
        }
        // Se selecciona Password
        case 1:{
            // Se almacena el indice de la función Password
            indice = 1;
            // Se muestra la pantalla de introducción del password
            pantalla.setCurrent(llamPassw0);
            break;
```

```
        }
    }
}

// Si se presionan los comandos atras0, cancelar1 o regresar
if ((c == atras0) || (c == cancelar1) || (c == regresar)){
    // Se retornará al Menu principal
    pantalla.setCurrent(menu);
    // Se borra el contenido de la caja de texto pedirVerpw
    pedirVerpw.setString("");
}

// Si se presionan los comandos atras1 o cancelar0
if ((c == atras1) || (c == cancelar0)){
    if ((password == null) && (datpw0 == null)){
        pedirPw1.setString("");
        pedirPw2.setString("");
        pantalla.setCurrent(intPassw,llamPassw1);
    }
    else{
        // Se retornará al Menu de Configuración
        pantalla.setCurrent(menuConf);
        // En las cajas de textos de petición de introdución de números
        // se debe mostrar los números actuales de funcionamiento
        pedirNum0.setString(numcel);
        pedirNum1.setString(numodem);
        // Se borra el contenido de las cajas de textos
        pedirPw0.setString("");
        pedirPw1.setString("");
        pedirPw2.setString("");
    }
```

```
        }

        // Si se presiona el comando okcmd0
        if (c == okcmd0){
            // Se verifica en que pantalla se presiono por medio de la variable indice
            switch(indice){
                // Se presionó en la pantalla de Asignación de Número
                case 0:{
                    // Si ambos números cumplen con la longitud de 8 digitos
                    if ((pedirNum0.size() == 8) && (pedirNum1.size() == 8)){
                        // Se obtiene el dato de las cajas de textos y se almacenan
                        // en las variables respectivas
                        numcel = pedirNum0.getString();
                        numodem = pedirNum1.getString();
                        // Se define una pantalla de aceptación de números
                        alerNumodem = new Alert("","Número de movil remitente:
"+numcel+"\n\nNúmero de modem: "+numodem,null,AlertType.CONFIRMATION);
                        // Se establece el tiempo de presentación de dicha pantalla
                        alerNumodem.setTimeout(3000);
                        // Se muestra la pantalla de aceptación de número y
                        // posteriormente se retorna al Menu de Configuración
                        pantalla.setCurrent(alerNumodem,menuConf);
                    }
                    // Si no se cumple la longitud de 8 digitos en ambos números
                    else{
                        // Se presenta una pantalla que pide introducir los números
correctamente
                        pantalla.setCurrent(alerNumal,llamNumodem);
                    }
                    break;
                }
```

```
// Se presionó en la pantalla de Password
case 1:{
   // Se obtiene el dato de la caja de texto para ser comparado
   datpw0 = pedirPw0.getString();
   // Se invoca al método para leer el registro de almacenamiento persistente
   leerRecordStore();
   // Si el dato es igual al valor del password
   if (datpw0.equals(password)){
      // Se limpia el contardor de intentos de ingreso de password
      j=0;
      // Se almacena el indice de que se ha introducido correctamente
      // el password para llamar a la pantalla de solicitud del nuevo password
      indice = 2;
      // Se presenta la pantalla de solicitud del nuevo password
      pantalla.setCurrent(llamPassw1);
   }
   // Si el dato no es igual al password
   else{
      // Se incrementa en uno el contador de intentos de ingreso de password
      j=j+1;
      // Se limpian las variables usadas en el proceso
      datpw0 = null;
      // Se limpian el cuadro de texto que solicita el ingreso de password
      pedirPw0.setString(null);
      // Si se han realizado los 5 intentos de ingreso de password
      if (j == 5){
         // Se coloca en la variable "BLOQUEADO"
         setBloq = "s";
         // Se manda a escribir en la memoria persistente
         setearBloqueo();
         // Se bloqueará la aplicación y se mostrará la pantalla de bloqueo
```

```
                    pantalla.setCurrent(bloqPant);
                }
                // Pero mientras tanto no se lleguen a los 5 intentos
                else{
                    // se presenta una pantalla informando que se ha introducido
incorrectamente
                    // el password para luego tratar de nuevo
                    pantalla.setCurrent(alerPw1,llamPassw0);
                }
            }
            break;
        }
        // Se presionó en la pantalla de verificación de password para autorizar la
acción
        case 9:{
            // Se obtiene el dato introducido en la caja de texto pedirVerpw
            datVerpw = pedirVerpw.getString();
            // Se invoca al método para leer el registro de almacenamiento persistente
            leerRecordStore();
            // Y se compara con el password
            if (password.equals(datVerpw)){
                j=0;
                // Debido a que la verificación de password se realiza en muchas
funciones
                // es necesario utilizar el indice indcom para saber en que función fue
                // invocada  la verificación de password
                switch(indcom){
                    // La Verificación de Password se ha dado para autorizar la Activar el
Sistema
                    case 1:{
                      // Se coloca en pantalla una bandera que indica que se ha activado
```

```
                    // el sistema a la par de la palabra Sistema
                    try{
                        car = Image.createImage("/Car.PNG");
                        menu.set(2," Sistema",car);
                    }
                    // Si no se puede colocar la bandera debe mandar una excepción
                    catch (Exception dib){
                        System.out.println("Imposible colocar bandera 1");
                    }
                    break;
                }
                // La Verificación de Password se ha dado para autorizar la
Desactivación del SIstema
                case 2:{
                    // Se quita en pantalla la bandera de Activación
                    menu.set(2,"Sistema",null);
                    break;
                }
                // La Verificación de Password se ha dado para autorizar el Bloqueo
del
                // funcionamiento del vehículo
                case 3:{
                    // Se coloca en pantalla una bandera que indica que el
funcionamiento del vehículo
                    // ha sido bloqueado, a la par de Func Vehículo
                    try{
                        nocar = Image.createImage("/noCar.PNG");
                        menu.set(3," Func. Vehículo",nocar);
                    }
                    // Si no se puede colocar dicha bandera se lanza una excepción
                    catch (Exception dib){
```

```
                        System.out.println("Imposible colocar bandera 2");
                    }
                    break;
                }
                // La Verificación de Password se ha dado para autorizar el
Desbloqueo del
                // funcionamiento del vehículo
                case 4:{
                    // Se quita la bandera de Bloqueo de vehículo
                    menu.set(3,"Func. Vehículo",null);
                    break;
                }
            }
            // Se crea un vinculo para que se ejecute el envío de mensajes (la cual
se define
            // más adelante)
            new Thread(this).start();
            // Se limpia el password introducido por el usuario
            pedirVerpw.setString(null);
            // Se define la presentación de un informe de la acción realizada
            alerVerpw0 = new Alert("Accion realizada: ", disp+"
"+func,null,AlertType.CONFIRMATION);
            // Se establece el tiempo de presentación en pantalla
            alerVerpw0.setTimeout(3000);
            // Se muestra el informe y posteriormente se retorna al Menu principal
            pantalla.setCurrent(alerVerpw0,menu);
        }
        // Pero si el password introducido es incorrecto
        else{
            // Se incrementa en uno el contados de intento de ingreso de password
            j=j+1;
```

```
                    // Si se tienen 5 intentos fallidos de ingreso de password
                    if (j == 5){
                        // Se indica a la variable que se debe bloquear y se manda a escribir
                        // en la memoria persistente
                        setBloq = "s";
                        setearBloqueo();
                        // Se llama a la pantalla de bloquep
                        pantalla.setCurrent(bloqPant);
                    }
                    // Pero si no se llega aun a los 5 intentos de ingreso de password
                    else{
                        // se borra el dato introducido por el usuario y se presenta una
                        // pantalla de error en el password para posteriormente regresar al
Menu Principal
                        pedirVerpw.setString(null);
                        pantalla.setCurrent(alerPw1,menu);
                    }
                }
                break;
            }
        }
    }

    // Si se presiona el comando okcmd1 (de la pantalla de introducción del nuevo
password)
    if (c == okcmd1){
        // Se verifica que la longitud del Password sea de 5 digitos
        if (pedirPw1.size()== 5){
            // Se obtiene el nuevo password introducido
            datpw1 = pedirPw1.getString();
            // Se situa el indice de pantalla de introducción de nuevo password
```

```
            indice = 3;
            // Y se llama la pantalla de verificación del nuevo password
            pantalla.setCurrent(llamPassw2);
         }
         // Si la longitud es menor a 5 digitos...
         else{
            // Se limpia el campo de texto
            pedirPw1.setString(null);
            // Se muestra una pantalla de error y luego se vuelve a la pantalla de ingreso
            // del nuevo password
            pantalla.setCurrent(longPassw,llamPassw1);
         }
      }

      // Si se presiona el comando okcmd2 (de la pantalla de verificación del nuevo
password)
      if (c == okcmd2){
         // Se obtiene la verificación del nuevo password introducido
         datpw2 = pedirPw2.getString();
         // Y se compara que la verificación de password sea igual al nuevo password
         if( datpw1.equals(datpw2)){
            // De ser iguales se almacena la contraseña en la variable password
            if (password == null){
               // Si es primera vez se crea el registro y se almacena el password
               escribirRecordStore();
            }
            else{
               // De lo contrario se modifica el registro cuantas veces el usuario cambie el
password
               modificarRecordStore();
            }
```

```
            // Se invoca al método para leer el registro de almacenamiento persistente
            leerRecordStore();
            // Se fija el codigo de la variable contenido
            contenido = "**6"+password;
            // Se fijan las variables de informe de la acción
            disp = "";
            func = "Password enviado";
            // Se limpian todas las variables involucradas en la introducción del password
            // por parte del usuario
            pedirPw0.setString(null);
            pedirPw1.setString(null);
            pedirPw2.setString(null);
            // Se crea un vinculo para que se ejecute el envío de mensajes
            // , donde se envia el password del teléfono móvil al modem
            new Thread(this).start();
            // Se presenta una pantalla que informa que la contraseña ha sido aceptada
            pantalla.setCurrent(alerPw2,menu);
         }
         else{
            // De lo contrario se limpian las variables asocidas al nuevo password
            // y se presenta un informe de error de password, para luego intentar de
nuevo
            pedirPw1.setString(null);
            pedirPw2.setString(null);
            pantalla.setCurrent(alerPw3,llamPassw1);
         }
      }

      // Si se presiona el comando selec0 (de la pantalla del Menu sistema)
      if (c == selec0){
         // Se fija información de la acción realizada
```

```
disp = "Sistema ";
// Se obtiene la función seleccionada dentro del Menu Sistema
switch(llamDisp0.getSelectedIndex()){
   // Se selecciona Activar el Sistema
   case 0:{
      // Se fija otro indice utilizada en la pantalla de verificación del password
      indcom = 1;
      // Se fija el resto de la información de la acción realizada
      func = "ACTIVADO";
      // Se fija el codigo de la función
      contenido = "**1"+password;
      // Se autoriza la utilización de la función Funcionamiento del Vehículo
      permiso = true;
      // Se invoca la Verificación de Password para autorizar la acción
      invVerpw();
      break;
   }
   // Se selecciona Desactivar el Sistema
   case 1:{
      // Se verifica que el Funcionamiento del Vehículo no esté Bloqueado
      // porque de lo contrario no se puede desactivar el sistema
      if (bloqueo == true){
         pantalla.setCurrent(pedirDesb,menu);
      }
      // Por el contrario si el Funcionamiento del Vehículo esta Desbloqueado
      else{
         // Se fija el indice utilizado en la verificación del password
         indcom = 2;
         // Se fija el resto de la información de la acción realizada
         func = "DESACTIVADO";
         // Se fija el codigo de la función
```

```
                contenido = "**2"+password;
                // No se autoriza la utilización de la función Funcionamiento del Vehículo
                // debido a que el sistema se encuentra Desactivado
                permiso = false;
                // Se invoca la Verificación del Password para autorizar la acción
                invVerpw();
            }
            break;
        }

    }
}

    // Si se presiona el comando selec1 (de la pantalla del Menu de Funcionamiento de
Vehículo)
    if (c == selec1){
        // Se fija la información  de la acción
        disp = "Funcionamiento de Vehículo";
        // Se obtiene la función seleccionada dentro del Menu Func de Vehículo
        switch(llamDisp1.getSelectedIndex()){
            // Se selecciona Bloquear el Func. del Vehículo
            case 0:{
                // Se fija el indice utilizado en la verificación del password
                indcom = 3;
                // Se fija el resto de la información de la acción realizada
                func = "BLOQUEADO";
                // Se fija el código de la función
                contenido = "**3"+password;
                // Se indica que el Bloqueo se ha realizado
                bloqueo = true;
                // Se invoca la Verificación del Password para autorizar la acción
```

```
                invVerpw();
                break;
            }
            // Se selecciona Desbloquar el Func. del Vehículo
            case 1:{
                // Se fija el indice utilizado en la verificación del password
                indcom = 4;
                // Se fija el resto de la información de la acción realizada
                func = "DESBLOQUEADO";
                // Se fija el código de la función
                contenido = "**4"+password;
                // Se indica que el Desbloqueo se ha realizado
                bloqueo = false;
                // Se invoca la verificación del Password para autorizar la acción
                invVerpw();
                break;
            }
        }
    }

    // Si se presiona el comando exit en la pantalla de confirmación de salida
    if ((c == exit1) || (c == exit2)){
        // Se invoca al método para cerrar el registro de almacenamiento persistente
        cerrarRecordStore();
        // Se sale de la aplicación y se borra de la memoria volatil destinada a las
aplicaciones
        destroyApp(false);
        notifyDestroyed();
    }
  }
```

```
    /*
     *
     * Método run, se ha construido para ser empleado en la gestión de envio de SMS
     *
     *
     */
    public void run(){
        // Se crea una conexión de gestión de SMS
        MessageConnection servidor = null;
        try {
            // Se define la conexión en modo servidor, en donde se especifica que la
            // conexión se hará por medio del número telefónico del remitente
            servidor = (MessageConnection)Connector.open("sms://+503"+numcel);
            // Se define como mensaje en forma de texto lo que se envia en el SMS
            TextMessage mensaje =
(TextMessage)servidor.newMessage(MessageConnection.TEXT_MESSAGE);
            // Se especifica la dirección del destinatario (número del modem)
            mensaje.setAddress("sms://+503"+numodem);
            // Lo que código contiene la varible contenido es lo que se enviará como
información
            mensaje.setPayloadText(contenido);
            // El SMS es enviado
            servidor.send(mensaje);
            // System.out.println(contenido);
        // Si no se puede enviar lanzará una excepción
        } catch (Throwable t) {
            System.out.println("Envio retenido: ");
            t.printStackTrace();
        }
        // Si la conexión está vacía, se cerrará dicha conexión
        if (servidor != null) {
```

```
            try {
                servidor.close();
            } catch (IOException ioe) {
            System.out.println("Cierre de conexion invalida");
            ioe.printStackTrace();
            }
        }
    }

    /*
     *
     * Método abrirRecordStore, se abre la sesión "guardar" para crear o abrir el registro
     * de almacenamiento persitente del password
     *
     */
    public void abrirRecordStore(){
        try{
            // Se abre la sesión y se le coloca nombre al registro guardar (para la
contraseña)
            guardar = RecordStore.openRecordStore("contraseña",true);
            // Se abre la sesión y se le coloca nombre al registro datBloq (para el bloqueo de
la aplicación)
            datBloq = RecordStore.openRecordStore("Bloqueo",true);
            // Presentación de datos en el simulador
            System.out.println(guardar.getName());
        }
        catch(RecordStoreException e){
            // Presentación de datos en el simulador
            System.out.println("Error al abrir el Record Store");
        }
    }
```

```
/*
 *
 * Método escribirRecordStore, se crea y almacena el password en el registro
 * de almacenamiento persitente
 *
 *
 */

public void escribirRecordStore(){
    // Se define la variable registro que almacenará temporalmente los datos
    byte[] registro;
    // Se obtine de datpw1, variable String, y se pasa a byte en la variable registro
    registro = datpw1.getBytes();
    try{
        // Se crea el registro y se almacena el dato
        guardar.addRecord(registro,0,registro.length);
        // Información del proceso para el emulador
        System.out.println("Dato almacenado");
    }
    catch(RecordStoreException e){
        // Información del error en el emulador
        System.out.println("Error al insertar registro");
    }
}


/*
 *
 * Método modificarRecordStore, se modifica el password en el registro
 * de almacenamiento persitente
```

```
 *
 */
 public void modificarRecordStore(){
    // Se define la variable registro que almacenará temporalmente los datos
    byte[] registro;
    // Se obtine de datpw1, variable String, y se pasa a byte en la variable registro
    registro = datpw1.getBytes();
    try{
        // Se almacena el nuevo password
        guardar.setRecord(1,registro,0,registro.length);
        // Información del procedimiento en el emulador
        System.out.println("Dato modificado");
    }
    catch(RecordStoreException e){
        // Información del error en el emulador
        System.out.println("Error al modificar registro");
    }
}


 /*
 *
 * Método setearBloqueo, crea y almacena la variable de bloqueo en el registro
 * de almacenamiento persitente
 *
 *
 */
 public void setearBloqueo(){
    // Se define la variable registro que almacenará temporalmente los datos
    byte[] registro;
    // Se obtine de datpw1, variable String, y se pasa a byte en la variable registro
```

```
    registro = setBloq.getBytes();
    try{
        // Se almacena la variable de bloqueo
        datBloq.addRecord(registro,0,registro.length);
        // Información del procedimiento en el emulador
        System.out.println("Aplicación bloqueada");
    }
    catch(RecordStoreException e){
        // Información del error en el emulador
        System.out.println("Error al modificar registro");
    }
}



/*
 *
 * Método leerRecordStore, se lee el password y la variable de bloqueo en el registro
 * de almacenamiento persitente
 *
 */
public void leerRecordStore(){
    // Se define la variable registro que almacenará temporalmente los datos
    byte[] registro = new byte[75];
    int longitud;
    // Se extrae byte por byte el contenido del password
    try{
        for (int i=1;i<=guardar.getNumRecords();i++){
            longitud = guardar.getRecordSize(i);
            registro = guardar.getRecord(i);
            System.out.println("Registro "+i+": "+ new String(registro,0,longitud));
            password = new String(registro,0,longitud);
```

```
            }
        }
        catch(RecordStoreException e){
            // Información del procedimiento en el emulador
            System.out.println(password);
            // Información del error en el emulador
            System.out.println("Error al leer registro");
        }
        // Se limpia la variable null
        registro = null;
        longitud = 0;
        // Se extrae byte por byte el contenido de la variable de bloqueo
        try{
            for (int i=1;i<=datBloq.getNumRecords();i++){
                longitud = datBloq.getRecordSize(i);
                registro = datBloq.getRecord(i);
                System.out.println("Registro "+i+": "+ new String(registro,0,longitud));
                setBloq = new String(registro,0,longitud);
            }
        }
        catch(RecordStoreException e){
            // Información del procedimiento en el emulador
            System.out.println(setBloq);
            // Información del error en el emulador
            System.out.println("Error al leer registro de bloqueo");
        }
        // Se limpia la variable null
        registro = null;
    }
```

```
    /*
    public void borrarRecordStore(){
        try{
            guardar.deleteRecordStore(guardar.getName());
        }
        catch(RecordStoreException e){
            System.out.println("Error al eliminar registro");
        }
    }
    */



    /*
     *
     * Método cerrarRecordStore, se cierra la sesión "guardar" donde se encuentra el
registro
     * de almacenamiento persitente
     *
     *
     */
    public void cerrarRecordStore(){
        try{
            // Se cierra la sesión
            guardar.closeRecordStore();
            datBloq.closeRecordStore();
        }
        catch(RecordStoreException e){
            // Se muestra inf del error en el emulador
            System.out.println("Error al cerrar el Record Store");
        }
    }
```

}

5.6 Codificación en los microcontroladores

5.6.1 Codificación del microcontrolador de transmisión

```
;ESTE ES UN PROGRAMA DE PARA COMUNICACION ENTRE EL PIC COMO DTE
Y EL MODEM DCE ETAPA DE TRANSMISION DE CARACTERES POR MEDIO DE LA
INTERFASE DE COMUNICION USART DEL PIC CON CONTROL DE FLUJO POR
MEDIO DE  HARDWARE, RTS Y CTS.
;
;*********************************************************
;                                                       *
;   Filename:    RS232_v2.2.asm                         *
;   Date:        28 DE AGOSTO DE 2005                   *
;   File Version:  Version 2.1                          *
;                                                       *
;   Author:  GRUPO DE TESIS                             *
;   Company:  UDB TESIS "SADD"                          *
;*********************************************************


        list            p=16f877A
        #include        <p16f877A.inc>

;*****************************************************************************************************
**********************
;DECLARACION DE REGISTROS Y VARIABLES Y CONSTANTES

CHARRX      EQU 0X20   ;DEL BANCO O REGISTRO EN DONDE SE GUARDA
                            CARACTER RECIBIDO POR USART
```

```
CHARTX      EQU 0X21  ;DEL BANCO O REGISTRO EN DONDE SE GUARDA
                      EL CARACTER A SER TRANSMITIDO ANTES DE
                      SER CARGADO AL TXREG
DTTBFLS  EQU 0X22     ;DEL BANCO 0, REGISTRO EN DONDE SE GUARDA
                      EL CARACTER LEIDO DE LAS TABLAS QUE ESTAN
                      EN FLASH
MSGLABEL EQU 0X23     ;DEL BANCO O
MARCAS      EQU 0X24  ;DEL BANCO O, REGISTRO EN DONDE LOS BIT
                      SIGNIFICAN CIERTOS EVENTOS
                      QUE HAN OCURRIDO.
                      ; BIT 0, SI ES 1,SE PRECIONO EL 1 COMO
                      SELECCION, SINO SE PRECIONO 2
                      ; SI NO SE PRESIONO NI EL 1 NI EL DOS
                      ENTONCES SE PONE EN 1 EL BIT 1.
MARCA23    EQU  0X25  ;VALOR DE MARCA QUE DICE SI 2 O 3 EL
                      VALOR DE COMPARACION
RAMTBL     EQU  0X30  ;LOCALIDADES DE RAM EN DONDE SE COMENZARA
                      A ALMACENAR EL MENSAJE RECIBIDO DESDE EL
                      MODEM
EEMARKS    EQU  0X0   ; LOCALIDAD DE LA EEPROM EN DONDE SE
                      GUARDA EL ESTADO DEL SISTEMA DESPUES DE SER
                      ALTERADO.
RXEEMSG    EQU  26H   ;REGISTRO EN DONDE SE ALMACENA LA
                      DIRECCION EN DONDE SE DESEA ALMACENAR
                      LOS DATOS EN EEPROM
DATAEE     EQU  27H   ;REGISTRO EN DONDE SE GUCARDA EL DATO
                      ANTES DE SER COLOCADO EN EEDATA

;VARIABLES DE PROPOSITO GENERALES
markcom    EQU  28H
CONTA1     EQU  0X2A
```

```
CONTA2      EQU   0X2B

;CONSTANTES
ADRMSH      EQU 05H     ;PARTE ALTA DE LA DIRECCION DONDE SE
                        ENCUENTRA LOS MENSAJES
ADRMS0L     EQU   00H   ;PARTE BAJA DE LA DIRECCION DONDE
                        SE ENCUENTRA MSG0
ADRMS1L     EQU 05H     ;PARTE BAJA DE LA DIRECCION DONDE SE
                        ENCUENTRA MSG1
ADRMS2L     EQU 11H     ;PARTE BAJA DE LA DIRECCION DONDE SE
                        ENCUENTRA MSG2
ADRMS3L     EQU 20H     ;PARTE BAJA DE LA DIRECCION DONDE SE
                        ENCUENTRA MSG3
ADRMS4L     EQU   50H   ;PARTE BAJA DE LA DIRECCION DONDE
                         SE ENCUENTRA MSG4
ADRMS5L     EQU   0X80  ;PARTE BAJA DE LA DIRECCION DONDE SE
                        ENCUENTRA MSG5
ADRMS6L     EQU   0XB0  ;PARTE BAJA DE LA DIRECCION DONDE SE
                        ENCUENTRA MSG6
ADRMS7L     EQU   0XE0  ;PARTE BAJA DE LA DIRECCION DONDE SE
                        ENCUENTRA MSG7
ADRMS8L     EQU   0XEA  ;PARTE BAJA DE LA DIRECCION DONDE SE
                        ENCUENTRA MSG8
EEADRRX     EQU   00H   ;VALOR DE DIRECCION DONDE SE
                        GUARDA EN EEPROM EL MENSAJE
                        RECIBIDO (USART)
EEADRTX     EQU   50H   ;VALOR DE LA DIRECCION DONDE
                        SE ALMACENA EN LA EEPROM EL
                        MENSAJE QUE SE HA TRANSMITIDO
```

```
;*************************************************************
;SFR Y BITS RELACIONADOS

STATUS      EQU 0X03    ;REGISTRO DE ESTADO DEL PIC
FSR         EQU  04H    ;REGISTRO PARA
                        DIRECCIONAMIENTO INDEXADO
INDF  EQU 00H           ;REGISTRO PARA DIRECCIONAMIENTO
                        INDEXADO
W           EQU  D'0'   ;DECLARACION DE REGISTRO W
F           EQU  D'1'   ;DECLARACION DE REGISTRO F
Z           EQU D'2'    ;DECLARACION DE BANDERA DE
                        ZERO
C           EQU D'0'    ;DECLARACION DE BANDERA DE
                         CARRIE
RP0         EQU  5      ;DECLARACION DE BIT DE SELECCION
                         DE BANCO
RP1         EQU 6       ;DECLARACION DE BIT DE SELECCION
                         DE BANCO

;PUERTO Y TRIS
porta  equ   0x5        ;DECLARACION DEL PUERTOA
trisa  equ   0x5        ;DECLARACION DEL REGISTRO DE
                        CONFIGURACION DEL PUERTOA
adcon1      equ   0x1f  ;DECLARACION DE REGISTRO DE CONFIGURACION
                          DE ENTRADAS DIGITALES DEL PUERTOA
trisb  equ 0x6          ;DECLARACION DEL REGISTRO DE CONFIGURACION
                        DEL PUERTO B
portb  equ 0x6          ;DECLARACION DEL PUERTOB
trisc  EQU 0X7          ;DELCARACION DEL REGISTRO DE CONFIGURACION
                        DEL PUERTO C
portc  EQU  0X07  ;DECLARACION DEL PUERTOC
```

```
trisd   equ   08h          ;DECLARACION DEL REGISTRO DE
                           CONFIGURACION DEL PUERTO D
portd   equ   08h          ;DECLARACION DEL PUERTOD
porte   equ   0x9          ;DECLARACION DEL PUERTOE
trise   equ   0x9          ;DECLARACION DEL REGISTRO DE
                           CONFIGURACION DEL PUERTO E

;REGISTROS RELACIONADOS A LA USART
RCSTA       EQU  18H   ;DECLARACION DE RESGITRO DE STATUS
                        DE RECEPCION DEL BANCO 0}
TXREG       EQU 19H    ;DEL BANCO 0, REGISTRO DE TRANSMISION
                       DE LA USART
RCREG       EQU 1AH    ;DEL BANCO O, REGISTRO DE RECEPCION
                       DE LA USART
txsta   EQU 18H    ;DECLARACION DE REGISTRO DE STATUS DE
                    TRANSMISION 18 DEL BANCO 1
spbrg  EQU 19H          ;DECLARACION DE REGISTRO DE BAUD RATE
                    GENERATOR 19 DEL BANCO 1


;SFR RELACIONADOS A LA EEPROM Y A LA FLASH
eeadrh      EQU  0FH          ;DEL BANCO 2 REGISTRO DONDE
                       SE ALMACENA LA PARTE ALTA DE LA
                        DIRECCION DE MEMORIA FLASH
eeadr  EQU  0DH        ;DEL BANCO 2 REGISTRO DONDE SE
                       ALMACENA LA PARTE BAJA DE LA
                       DIRECCION DE MEMORIA NO VOLATIL
eecon1      EQU  0CH   ;DEL BANCO 3 REGISTRO DE CONFIGURACION
eecon2      EQU  0DH   ;DEL BANCO 3 REGISTRO DE CONFIGURACION
eedata      EQU  0CH   ;DEL BANCO 2 REGISTRO DONDE SE
                        ALMACENA EL DATO LEIDO
```

```
;REGISTROS FSR RELACIONADOS CON INTERRUPCIONES

PIR1   EQU   0X0C          ;DEL BANCO 0, BANDERAS DE INTERRUPCIONES
INTCON       EQU 0X0B      ;HABILITADOR GENERAL DE INTERRUPCIONES Y
                            LOCAL ENABLES EXTERNAL INTERRUPS (INTB AND
                           PORTB CHANGE)
pie1    EQU 0X0C           ;DEL BANCO 1, HABILITADORES DE INTERRUPCIONES
                           DE PERIFERICOS.
option_reg    EQU   1H     ;BANK 1, REGISTO QUE CONTIENE
                            CONFIGURACION DEL PUERTO B COMO
                           INTERRUPCIONES, PULL UPS ENAL/DISA Y
                           ;SELECTOR DE FLANCO DE LAS
                           INTERRUPCIONES (INTEDG =BIT 6 )
                            1=RASING  0=FALLING


;*****************************************************************
;*****************************************************************
;TABLA DE MENSAJES PARA TRANSMISION

        ORG 500H
MSG0 DW 'A','T',0XD,0XA,0XCC
        ORG 505H
MSG1 DW 'A','T','+','C','M','G','R','=','1',0XD,0XCC
        ORG 511H
MSG2 DW 'A','T','+','C','M','G','D','=','1',',','4',0XD,0XCC
        ORG 520H
MSG3 DW    'A','T','+','C','M','G','S','=',"",'7','8','7','0','8','0','4','1',
        "",0XD,'S','I','S','.',' ','A','C','T','I','V','A','D','O',0X1A,0XCC
        ORG 550H
```

```
MSG4 DW 'A','T','+','C','M','G','S','=',"",'7','8','7','0','8','0','4','1',
        "",0XD,'S','I','S','.',' ','A','C','T','.','-','A','U','T','O',' ',
        'B','L','O','Q','.',0X1A,0XCC
        ORG 580H
MSG5 DW 'A','T','+','C','M','G','S','=',"",'7','8','7','0','8','0','4','1',
        "",0XD,'S','I','S','.',' ','A','C','T','.','-','A','U','T','O',' ',
        'D','E','S','B','L','O','Q','.',0X1A,0XCC
        ORG 0X5B0
MSG6 DW 'A','T','+','C','M','G','S','=',"",'7','8','7','0','8','0','4','1',
        "",0XD,'S','I','S','.',' ','D','E','S','A','C','T','.',0X1A,0XCC

;TABLA DE MENSAJES PARA COMPARACION DE LA RECEPCION

        ORG 5E0H
MSG7 DW    '+','C','M','T','I',':'
        ORG 0X5EA
MSG8 DW '+','C','M','G','R',':'


        ORG 600H
MSG9 DW 'A','T','+','C','M','G','S','=',"",'7','8','7','0','8','0','4','1',
        "",0XD,'P','U','E','R','T','A','S',' ','A','B','I','E','R','T','A',
        0X1A,0XCC


        ORG 650H
MSG10 DW 'A','T','+','C','M','G','S','=',"",'7','8','7','0','8','0','4','1',
        "",0XD,'L','U','C','E','S',' ','E','N','C','E','N','D','I','D','A','S',
        0X1A,0XCC

        ORG 680H
```

```
MSG11 DW 'A','T','+','C','M','G','S','=',"","7','8','7','0','8','0','4','1',
        "",0XD,'E','L',' ','A','U','T','O',' ','H','A',' ','S','I','D','O',' ',
        'E','N','C','E','N','D','I','D','O',0X1A,0XCC



;*****************************************************************
;*****************************************************************
;MACROS PARA SELECCION DE BANCO

BANK0:      macro               ; select register bank 0
            bcf    STATUS,RP0
            bcf    STATUS,RP1
            endm
;-----------------------------------------------
BANK1:      macro               ; select register bank 1
            bsf    STATUS,RP0
            bcf    STATUS,RP1
            endm
;-----------------------------------------------
BANK2:      macro               ; select register bank 2
            bcf    STATUS,RP0
            bsf    STATUS,RP1
            endm
;-----------------------------------------------
BANK3:      macro               ; select register bank 3
            bsf    STATUS,RP0
            bsf    STATUS,RP1
            endm
;-----------------------------------------------
;*****************************************************************
```

```
;*****************************************************************


        ORG 0X0

        GOTO MANAGEMENT

        ORG 0XA
MANAGEMENT:
        CALL  INICIALIZACION    ;SE LLAMA LA RUTINA DE INICIALIZACION
        CALL  FLOWCONTROL       ;SE LLAMA A LA RUTINA PARA
                                SETEAR LINEAS DEL CONTROL DE FLUJO
        CALL  CONFIGURACION ;SE LLAMA LA RUTINA DE CONFIGURACION
                                 DE PERIFERICOS
        CALL  RS232TX           ;SE LLAMA A LA RUTINA DE
                                HABILITACION DE RS232TX, SE
                                HABILITA LA TRANSMISION
        CALL  RS232RX           ;SE HABILITA LA RECEPCION DE
                                CARCATERES POR LA USART
        BSF         PORTD,0     ;SE SETEA EL BIT0 DEL
                                PUERTO D PARA INICIALIZAR EL
                                PIC DE RECEPCION.
        CALL  DELAY             ;SE LLAMA A LA RUTINA DE RETARDO
        BCF         PORTD,0     ;SE LIMPIA EL BIT PARA DEJAR AL
                                PIC DE RECEPCION FUNCIONANDO
        GOTO        WAITCODE  ;SALTA A ESPERAR QUE EL PIC DE
                                 RECEPCION ENVIE EL CODIGO QUE
                                REPRESENTA LA ACCION A EJECUTAR.

;*****************************************************************
```

```
;SUBRUTINAS A UTILIZAR, LECTURA DE FLASH, ESCRITURA DE EEPROM,
HABILITACION
DE TX Y RX DE LA USART

;_____________________________________________________________
;RUTINA DE INICIALIZACION DE VARIABLES Y PUERTOS

INICIALIZACION:
        BANK0
        CLRF CHARRX         ;SE LIMPIA LOS REGISTROS DE PROPOSITOS
                            GENERALES Y CONTADORES A UTILIZAR
        CLRF CHARTX         ;SE LIMPIA LOS REGISTROS DE PROPOSITOS
                            GENERALES Y CONTADORES A UTILIZAR
        CLRF DTTBFLS        ;SE LIMPIA LOS REGISTROS DE PROPOSITOS
                            GENERALES Y CONTADORES A UTILIZAR
        CLRF MSGLABEL       ;SE LIMPIA LOS REGISTROS DE PROPOSITOS
                            GENERALES Y CONTADORES A UTILIZAR
        CLRF MARCAS         ;SE LIMPIA LOS REGISTROS DE PROPOSITOS
                            GENERALES Y CONTADORES A UTILIZAR
        CLRF MARCA23        ;SE LIMPIA LOS REGISTROS DE PROPOSITOS
                            GENERALES Y CONTADORES A UTILIZAR
        INCF  MARCA23,F     ;SE INICIALIZA EL VALOR DEL REGISTRO
                            MARCA23
        CLRF RAMTBL         ;SE LIMPIA LOS REGISTROS DE PROPOSITOS
                            GENERALES Y CONTADORES A UTILIZAR
        CLRF EEMARKS        ;SE LIMPIA LOS REGISTROS DE PROPOSITOS
                            GENERALES Y CONTADORES A UTILIZAR
        CLRF RXEEMSG        ;SE LIMPIA LOS REGISTROS DE PROPOSITOS
                            GENERALES Y CONTADORES A UTILIZAR
        CLRF DATAEE         ;SE LIMPIA LOS REGISTROS DE PROPOSITOS
                             GENERALES Y CONTADORES A UTILIZAR
```

```
CLRF markcom            ;SE LIMPIA LOS REGISTROS DE PROPOSITOS
                        GENERALES Y CONTADORES A UTILIZAR
CLRF CONTA1             ;SE LIMPIA LOS REGISTROS DE PROPOSITOS
                        GENERALES Y CONTADORES A UTILIZAR
CLRF CONTA2             ;SE LIMPIA LOS REGISTROS DE PROPOSITOS
                        GENERALES Y CONTADORES A UTILIZAR
BANK1
MOVLW   b'11111'        ;los 5 bits del puerto A son entradas
                        para interruptores de propositos generales
MOVWF   trisa
MOVLW   B'11110001'     ;EL BIT0 DEL PUERTO B, ES DE INTERRUPCION
                         EXTERNA, LOS BITS DE 1-3 SON SALIDAS
                        PARA INDICACIONES
                        ;LOS BITS 4-7, SON ENTRADAS PARA LOS
                        SENSORES QUE SE TENDRIAN INSTALADOS
                        EN EL AUTO.
MOVWF   trisb
MOVLW   0X7
MOVWF   adcon1              ;configuro el puerto A con entradas
                        digitales y no entradas analogas.
MOVLW   B'10001010'     ;el puerto c es configurado de manera
                        que el bit0=out=DTR, bit1=IN=DSR,
                        bit2=out=RTS, bit3=in=CTS
                        ;bits 4 y 5 son outputs generales;
                        Bit 6 = out = TX and bit7=In=RX.
MOVWF   trisc
CLRF trisd              ;SE CONFIGURA EL PUERTO D COMO SALIDAS
                        DIGITALES
CLRF trise              ;SE CONFIGURA EL PUERTO E COMO SALIDAS
                        DIGITALES
MOVLW   0X7
```

```
        MOVWF   trise
        BANK0
        CLRF    porta       ;SE LIMPIA LOS PUERTO DE ENTRADAS Y
                            SALIDAS DEL MICRONCONTROLAR
        CLRF    portb       ;SE LIMPIA LOS PUERTO DE ENTRADAS Y
                             SALIDAS DEL MICRONCONTROLAR
        CLRF    portc       ;SE LIMPIA LOS PUERTO DE ENTRADAS Y
                            SALIDAS DEL MICRONCONTROLAR
        CLRF    portd       ;SE LIMPIA LOS PUERTO DE ENTRADAS Y
                             SALIDAS DEL MICRONCONTROLAR
        clrf    porte       ;SE LIMPIA LOS PUERTO DE ENTRADAS Y
                            SALIDAS DEL MICRONCONTROLAR
        CLRF    TXREG       ;SE LIMPIA EL REGISTRO DE TRANSMISION
                             DE LA USART
        RETURN
_____________________________________________________________

;RUTINA DE CONFIGURACION CONTROL DE FLUJO

FLOWCONTROL:
        BCF     portc,0     ;se limpia para poner en +12
                            la linea de DTR
        BCF     portc,2     ;se limpia para poner en +12
                            la linea de RTS y poder recibir
                            datos del Modem
        RETURN


;_____________________________________________________________

;RUTINA DE DE CONFIGURACION DE INTERRUPCIONES
;INTERRUPCIONES A UTILIZAR, SON LAS DE INT (RB0) AND PORTBCHANGE
(BITS 4-7 PORTB)
```

```
CONFIGURACION:
        MOVLW   0XFF
        MOVWF   option_reg  ;SE CONFIGURA EL FLANCO DE LAS
                            INTERRUPCIONES ASCENDENTE,
                            DESABILITADAS LAS PULL UPS.
        MOVLW   B'01011000' ;CONFIGURACIONES DE INTERRUPECIONES
                            NO PERIFERICAS
        MOVWF   INTCON      ;EL BIT0=BANDERA DE PORTB CHANGE INT
                            (1=OCURRIO); BIT1=BANDERA DE INT
                            EXTERNA (RB0=OCURRED)
                            ;CON BITS 3 Y 4 HABILITO INTERRUPECIONES
                            DEL PUERTO B, CON EL BIT 5 DESABILITO
                            INTERRUPCION TIMER0
                            ;CON BIT6 HABILITO LAS INTERRUPCIONES
                            PERIFERICAS, COMO LAS RECEPCION Y
                            TRANSMISION DE LA USART
                            ;BIT 7 HABILITADOR GLOBAL,SE HABILITA LUEGO.
        BANK1
        MOVLW   B'00100000' ;SE HABILITA SOLO LA INTERRUPCION
                            PERIFERICA DE RECEPCION DE LA USART.
        MOVWF   pie1
        BANK0
        BSF     INTCON,7    ;SE HABILITA LAS INTERRUPCIONES
                             GLOBALES
        RETURN

;-----------------------------------------------------------
;RUTINA DE HABILITACION DE TX Y RX EN LA USART DEL PIC

RS232TX:
```

```
        BANK1
        MOVLW   B'00000110' ;SE CONFIGURA EL PERIFERICO DE
                            COMUNICACION USART DE
MICROCONTROLADOR
        MOVWF   txsta
        MOVLW   D'25'
        MOVWF   spbrg       ;SE CONFIGURA EL VALOR NECESARIO PARA
                            TENER UN BOUD RATE DE 9600
        BANK0
        MOVLW   B'10000000' ;SE HABILITA LA TRANSMISION DE LA USART
        MOVWF   RCSTA
        BANK1
        BSF     txsta,5     ; SE SETEA EL BIT DE INICIO DE
                             TRANSMISION
        RETURN
;-------------------------------------------------------
RS232RX:
        BANK0
        BSF  INTCON,7       ;SE HABILITA LAS INTERRUPCIONES NO
ENMASCARABLES.
        BSF  RCSTA,4        ; SE HABILITA LA RECEPCION CONTINUA DE
CARACTERES.
        RETURN
;-------------------------------------------------------

;-------------------------------------------------------
;RUTINA QUE ESPERA EL CODIGO QUE SE INTERPRETA PARA DETERMINAR LA
ACCION
A REALIZAR.

WAITCODE:
```

```
        MOVF PORTE,W        ; SE LEE EL CONTENIDO DEL PUERTO E, SI ESTE
                            VALOR ES CERO NUNCA SALTARA A EJECUTAR
                            ACCION ALGUNA
        XORLW   0X1
        BTFSC   STATUS,Z    ; SI SE RECIBE EL CODIGO 1 SALTA A LA RUTINA
                            CODE1
        GOTO  CODE1
        MOVF PORTE,W
        XORLW   0X2
        BTFSC   STATUS,Z    ; SI SE RECIBE EL CODIGO 2 SALTA A LA RUTINA
                            CODE2
        GOTO  CODE2
        GOTOWAITCODE


CODE1
        CALL  TRANSMITION1  ;SE LLAMA LA RUTINA DE TRANSMISION DE
                            CARACTERES POR LA USART
        bcf     portc,2     ;SE VUELVE A HABILITAR LA RECEPCION DE
                             CARACTERES  HACIA EL MICRONCONTROLADOR

        GOTO    WAITCODE    ;SE REGRESA A ESPERAR UN NUEVO CODIGO

CODE2
      CALL TRANSMITION2     ;SE LLAMA LA RUTINA DE TRANSMITION2
        bcf     portc,2     ;SE VUELVE HABILITAR LA RECEPCION DE
                            CARACTERES HACIA EL MICRONCONTROLADOR

        GOTO    WAITCODE    ;SE REGRESA A ESPERAR UN NUEVO CODIGO
```

```
TRANSMITION1
        MOVWF     RXEEMSG   ;SE TRANSFIERE EL VALOR DE W A RXEEMSG
        MOVLW     EEADRTX   ;SE COLOCA LA CONSTANTE EEADRTX, EN W
        MOVLW     ADRMS0L   ;SE COLOCA EN W EN VALOR DE LA CONSTANTE
                            ADRMSOL
        MOVWF     MSGLABEL  ;SE ENVIA ESTE VALOR AL REGISTRO MSGLABEL
        CALL  TRANSMITIR    ;SE LLAMA A LA RUTINA QUE TRANSMITE LOS
                            CARACTERES PARA ENVIAR EL MSG0
        MOVLW     ADRMS1L   ;SE COLOCA EL VALOR DE LA DIRECCION EN
                            DONDE ESTA EL MSG1 EN EL REGISTRO
                            MSGLABEL
        MOVWF     MSGLABEL
        CALL  TRANSMITIR    ;SE TRANSMITEN LOS CARACTERES A TRAVES
                            DE LA USART
        RETURN


TRANSMITION2
        bsf     portd,2     ;SE DESHABILITA LA RECEPCION DE
                            CARACTERES
                            DESDE EL MODEM PARA EVITAR QUE SE
                            PIERDAN DATOS
        MOVLW     ADRMS0L   ;SE CARGA EL VALOR DE LA DIRECCION DONDE
                            SE ENCUENTRA EL MENSAJE 0
        MOVWF     MSGLABEL  ; SE ENVIA AL REGISTRO MSGLABEL
        CALL  TRANSMITIR    ; SE LLAMA A RUTINA QUE TRANSMITE LOS
                            CARACTERES
        MOVLW     ADRMS3L   ; SE CARGA LA DIRECCION EN  DONDE SE
                            ENCUENTRA  EL MENSAJE 3
        MOVWF     MSGLABEL  ; SE ENVIA AL REGISTRO MSGLABEL
```

```
        CALL  TRANSMITIR        ; SE LLAMA A RUTINA QUE TRANSMITE LOS
CARACTERES
        call    DELAY           ; SE ESPERA A QUE SE HAYA TERMINADO LA
                                TRANSMISION DEL SMS
        call    DELAY
        call    DELAY
        call    DELAY
        call    DELAY
        call    DELAY
        call    DELAY
        call    DELAY
        call    DELAY
        call    DELAY
        MOVLW       ADRMS0L     ;SE CARGA EL VALOR DE LA DIRECCION DONDE
                                 SE ENCUENTRA EL MENSAJE 0
        MOVWF       MSGLABEL    ; SE ENVIA AL REGISTRO MSGLABEL
        CALL  TRANSMITIR        ; SE LLAMA A RUTINA QUE TRANSMITE LOS
                                CARACTERES
        MOVLW       ADRMS2L     ; SE CARGA LA DIRECCION EN  DONDE SE
                                ENCUENTRA  EL MENSAJE 2
        MOVWF       MSGLABEL    ; SE ENVIA AL REGISTRO MSGLABEL
        CALL  TRANSMITIR        ; SE LLAMA A RUTINA QUE TRANSMITE LOS
                                CARACTERES
        bcf     portd,2
        RETURN


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

;RUTINA DE TRANSMITIR CARACTER POR CARACTER.
```

```
TRANSMITIR
        bsf     portc,2                 ;SE DETIENE LA RECPCION DE
                                        CARCATERES  PARA EVITAR QUE SE
                                        PIERDAN, SE LIMPIA RTS
WAITCTS BTFSC   PORTC,3     ;SE ESPERA A QUE CTS SE PONGA EN +12V
                                        (O LOGICO)
        GOTO    WAITCTS
        CALL LEERFLASH      ;SE LLAMA A LA RUTINA DE LECTURA DE FLASH
        MOVFDTTBFLS,W       ;SE PASA A W, EL VALOR LEIDO DE LA FLASH,
                                        (DATOS DE TABLAS)
        MOVWF   CHARTX      ;SE TRANSFIERE TAMBIEN A CHARTX, REGISTRO
                                         TEMPORAL
        XORLW   0XCC        ;SE REVISA SI SE TRANSMITE EL ULTIMO
                                        CARACTER
        BTFSC   STATUS,Z    ;SI FUE EL ULTIMO SE SALE DE LA RUTINA
        RETURN
        MOVFCHARTX,W        ; SINO LO ES SE GUARDA EL CARCATER QUE SE
                                        TRANSMITIRA EN LA MEMORIA EEPROM DEL PIC
        MOVWF   DATAEE
        CALL WRITEPROM      ; SE LLAMA LA RUTINA DE ESCRIBIR EN LA
                                        EEPROM
        MOVFCHARTX,W
        MOVWF   TXREG       ;SE CARGA ESTE MISMO VALOR EN EL REGISTRO
                                        DE  TRANSMISION DE LA USART TXREG
        CALL DELAY
        BANK1
WAITTX  BTFSS   txsta,1     ; SE ESPERA A QUE EL REGISTRO TX ESTE
VACIO
        GOTO    WAITTX
        BANK0
```

```
        INCF  MSGLABEL,F        ;QUE INCREMENTE EL CONTENIDO DEL
                                 REGISTRO
                                MSGLABEL PARA ENVIAR EL SIGUIENTE
                                CARÁCTER
        INCF  RXEEMSG,F
        GOTO        WAITCTS     ; SE ESPERA DE NUEVO A QUE SE SETEE CTS
                                PARA  ENVIAR EL NUEVO CARACTER

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

;-----------------------------------------------------
;RUTINA DE LECTURA DE FLASH

LEERFLASH:

        MOVLW       ADRMSH          ; PARTE ALTA DE LA DIRECCION DONDE SE
                                    ENCUENTRAN  LOS MENSAJES
        BANK2
        MOVWF       eeadrh          ; SE GUARDA EN EL FSR
CORRESPONDIENTE
        BANK0
        MOVF        MSGLABEL,W      ; SE CARGA EL CONTENIDO DEL REGISTRO
                                    A W
        BANK2
        MOVWF       eeadr           ; LUEGO SE CARGA EN EL FSR
                                    CORRESPONDIENTE A LA PARTE BAJA DE
                                    LA DIRECCION
        BANK3                       ; Bank 3
        BSF    eecon1, 7            ; SE APUNTA A LA MEMORIA DE
                                    PROGRAMA
```

```
        BSF    eecon1, 0              ; SE HABILITA LECTURA DE MEMORIA
        NOP                           ; SECUENCIA OBLIGADA
        NOP
        BANK2                         ; Bank 2
        MOVF        eedata, W         ; SE GUARDA EL DATO LEIDO EN W
        BANK0
        MOVWF       DTTBFLS           ; Y SE GUARDA EN LA RAM
        RETURN
;-------------------------------------------------------


;-------------------------------------------------------
;RUTINA PARA ESCRIBIR EN LA EEPROM

WRITEPROM:
        BANK0
        MOVF        RXEEMSG,W         ;SE TRANSFIERE A W LA DIRECCION EN
                                      DONDE  SE DESEA ESCRIBIR
        BANK2
        MOVWF       eeadr             ;SE GUARDA EN EL REGISTRO ADECUADO
        BANK0
        MOVF        DATAEE,W          ;SE CARGA A W EL DATO QUE SE DESEA
                                      QUEMAR  EN LA EEPROM
        BANK2
        MOVWF       eedata            ;SE CARGA EN EL REGISTRO ADECUADO
        BANK3
        BCF    eecon1,7               ;SE APUNTA A LA MEMORIA DE DATOS
        BSF    eecon1,2               ;SE HABILITA LA ESCRITURA EN EL
                                      EEPROM
        MOVLW       55h               ;SECUENCIA OBLIGADA
        MOVWF       eecon2            ;ESCRIBIR 55H EN EECON2
```

```
        MOVLW   0xAA            ;SECUENCIA OBLIGADA
        MOVWF   eecon2          ;ESCRIBIR AAH EN  EL EECON2
        BSF   eecon1,1          ;SE SETEA BIT PARA HABILITAR LA
                                ESCRITURA EN LA EEPROM
        BANK3
WAITWR  BTFSC   eecon1,1        ;ESPERAMOS QUE LA ESCRITURA FINALICE
        GOTO    WAITWR          ;SALTA A ESPERAR
        BANK0
        RETURN
;-----------------------------------------------------

;RUTINA PARA LEER EN LA EEPROM

READEPROM:
        BANK0
        MOVF    RXEEMSG,W       ;SE TRANSFIERE A W LA DIRECCION EN
                                 DONDE SE DESEA ESCRIBIR
        BANK2
        MOVWF   eeadr           ;SE GUARDA EN EL REGISTRO ADECUADO
        BANK3
        BCF   eecon1,7          ;SE APUNTA A LA MEMORIA DE DATOS
        BSF   eecon1,0          ;SE SETEA BIT PARA HABILITAR LA
                                LECTURA EN LA EEPROM
        BANK2
        MOVFeedata,w
        BANK0
        MOVWF   DATAEE          ;SE GUARDA EL DATO LEIDO EN EL
                                REGISTRO  DATAEE, PARA SER UTILIZADO
        RETURN

;-----------------------------------------------------
```

```
;SUBRUTINA QUE GENERA DELAY

DELAY:
        MOVLW 0XFF              ;SE SETEE EL VALOR DEL CONTADOR 1 Y 2
        MOVWF   CONTA1
        MOVWF   CONTA2
DECONTA1 DECF  CONTA1,1   ;SE DECREMENTA ESTE VALOR EN EL
CONTADOR 1
        MOVFCONTA1,W            ;SE ESPERA A QUE CONTADOR 1 LLEGUE A 0
        BTFSS   STATUS,Z  ;SI LO HACE SE DECREMENTA EL CONTADOR 2
        GOTODECONTA1            ; SI NO ES ASI SE VUELVE A DECREMENTAR
                                EL CONTADOR 1
        DECF CONTA2,1           ;SE  DECRMENTA CONTADOR 2
        MOVFCONTA2,W            ; SE COMPARA SI ES CERO
        BTFSC   STATUS,Z
        RETURN                  ; SALE DE LA RUTINA DELAY SI CONTADOR 2
                                ES CERO
        MOVLW   0XFF            ;SI NO LO ES SE VUELVE A SETEAR EL
                                CONTADOR  1 CON EL VALOR 0XFF
        MOVWF   CONTA1
        GOTODECONTA1            ; SE REGRESA A REALIZAR EL DECREMENTO

        END
```

5.6.2 Codificación del microcontrolador de recepción

```
;ESTE ES UN PROGRAMA DE PARA COMUNICACION ENTRE EL PIC COMO DTE
Y EL MODEM DCE ;ETAPA DE RECEPCION DE CARACTERES POR MEDIO DE LA
INTERFASE DE COMUNICION USART DEL PIC CON CONTROL DE FLUJO POR
MEDIO DE  HARDWARE, RTS Y CTS.
```

```
;
;*************************************************
;                                                *
;   Filename:                                    *
;   Date:        28 DE AGOSTO DE 2005            *
;   File Version:  Version 2.1                   *
;                                                *
;   Author:  TESIS GROUP                         *
;   Company:  UDB TESIS SADD                     *
;*************************************************


        list            p=16f877A
        #include        <p16f877A.inc>

;*********************************************************
;DECLARACION DE REGISTROS Y VARIABLES Y CONSTANTES

CHARRX      EQU 0X20    ;DEL BANCO O REGISTRO EN DONDE SE GUARDA
                        CARACTER RECIBIDO POR USART
CHARTX      EQU 0X21    ;DEL BANCO O REGISTRO EN DONDE SE GUARDA
                        EL CARACTER A SER TRANSMITIDO ANTES DE SER
                        CARGADO AL TXREG
DTTBFLS EQU 0X22        ;DEL BANCO 0, REGISTRO EN DONDE SE GUARDA
                        EL CARACTER LEIDO DE LAS TABLAS QUE ESTAN EN
                        FLASH
MSGLABEL EQU 0X23       ;DEL BANCO O
MARCAS      EQU   0X24  ;DEL BANCO O, REGISTRO EN DONDE LOS BIT
                        SIGNIFICAN CIERTOS EVENTOS       QUE HAN
                        OCURRIDO.
                        ; BIT 0, SI ES 1,SE PRECIONO EL 1 COMO SELECCION,
```

```
                        SINO SE PRECIONO 2
                        ; SI NO SE PRESIONO NI EL 1 NI EL DOS ENTONCES SE
                        PONE EN 1 EL BIT 1.
MARCA23   EQU  0X25  ;VALOR DE MARCA QUE DICE SI 2 O 3 EL VALOR DE
                        COMPARACION
RAMTBL    EQU  0X30  ;LOCALIDADES DE RAM EN DONDE SE COMENZARA A
                        ALMACENAR EL MENSAJE RECIBIDO DESDE EL MODEM
EEMARKS   EQU  0X0   ; LOCALIDAD DE LA EEPROM EN DONDE SE
                        GUARDA EL ESTADO DEL SISTEMA DESPUES DE SER
                        ALTERADO.
RXEEMSG   EQU  26H   ;REGISTRO EN DONDE SE ALMACENA LA DIRECCION
                        EN DONDE
                        SE DESEA ALMACENAR LOS DATOS EN EEPROM
DATAEE    EQU  27H   ;REGISTRO EN DONDE SE GUCARDA EL DATO ANTES
                        DE SER COLOCADO EN EEDATA

;VARIABLES DE PROPOSITO GENERALES
markcom   EQU  28H
CONTA1    EQU  0X2A
CONTA2    EQU  0X2B

;CONSTANTES
ADRMSH    EQU 05H        ;PARTE ALTA DE LA DIRECCION DONDE SE
                         ENCUENTRA LOS MENSAJES
ADRMS0L   EQU  00H       ;PARTE BAJA DE LA DIRECCION DONDE SE
                         ENCUENTRA MSG1
ADRMS1L   EQU 05H        ;PARTE BAJA DE LA DIRECCION DONDE SE
ENCUENTRA MSG2
ADRMS2L   EQU 11H        ;"
ADRMS3L   EQU 20H        ;"
ADRMS4L   EQU  50H       ;"
```

```
ADRMS5L    EQU   0X80  ;"
ADRMS6L    EQU   0XB0
ADRMS7L    EQU   0XE0
ADRMS8L    EQU   0XEA
EEADRRX    EQU   00H   ;VALOR DE DIRECCION DONDE SE GUARDA EN
                       EEPROM EL MENSAJE RECIBIDO (USART)
EEADRTX    EQU   50H   ;VALOR DE LA DIRECCION DONDE SE ALMACENA EN LA
                       EEPROM EL MENSAJE QUE SE HA TRANSMITIDO


;*********************************************************************
;SFR Y BITS RELACIONADOS

STATUS     EQU 0X03    ;REGISTRO DE ESTADO DEL PIC
FSR        EQU   04H
INDF  EQU 00H
W          EQU   D'0'  ;DECLARACION DE REGISTRO W
F          EQU   D'1'  ;DECLARACION DE REGISTRO
Z          EQU D'2'
C          EQU D'0'
RP0        EQU   5
RP1        EQU 6

;PUERTO Y TRIS
porta  equ    0x5         ;del banco0
trisa  equ    0x5         ;del banco1
adcon1     equ    0x1f    ;del banco1
trisb  equ 0x6             ;del banco 1
portb  equ 0x6             ;del banco 0
trisc  EQU 0X7             ;DEL BANCO 1
portc  EQU   0X07          ;DEL BANCO 0
```

```
trisd   equ   08h          ;DEL BANCO1
portd   equ   08h          ;DEL BANCO 0
porte   equ   0x9          ;del banco0
trise   equ   0x9          ;del banco1

;REGISTROS RELACIONADOS A LA USART
RCSTA       EQU   18H   ;DECLARACION DE RESGITRO DE STATUS DE
                         RECEPCION DEL BANCO 0}
TXREG       EQU 19H     ;DEL BANCO 0, REGISTRO DE TRANSMISION DE
                        LA USART
RCREG       EQU 1AH     ;DEL BANCO O, REGISTRO DE RECEPCION DE LA
                        USART
txsta   EQU 18H         ;DECLARACION DE REGISTRO DE STATUS DE
                        TRANSMISION 18 DEL BANCO 1
spbrg  EQU 19H          ;DECLARACION DE REGISTRO DE BAUD RATE
                        GENERATOR 19 DEL BANCO 1


;SFR RELACIONADOS A LA EEPROM Y A LA FLASH
eeadrh      EQU   0FH   ;DEL BANCO 2
eeadr EQU   0DH         ;DEL BANCO 2
eecon1      EQU   0CH   ;DEL BANCO 3
eecon2      EQU   0DH   ;DEL BANCO 3
eedata      EQU   0CH   ;DEL BANCO 2
;eedath     EQU 0EH     ;DEL BANCO 2

;REGISTROS FSR RELACIONADOS CON INTERRUPCIONES

PIR1   EQU   0X0C       ;DEL BANCO 0, BANDERAS DE INTERRUPCIONES
INTCON      EQU 0X0B    ;HABILITADOR GENERAL DE INTERRUPCIONES Y
                        LOCAL ENABLES EXTERNAL INTERRUPS (INTB AND
```

```
                        PORTB CHANGE)
pie1    EQU 0X0C        ;DEL BANCO 1, HABILITADORES DE INTERRUPCIONES
                        DE PERIFERICOS.
option_reg    EQU   1H  ;BANK 1, REGISTO QUE CONTIENE CONFIGURACION
                        DEL PUERTO B COMO INTERRUPCIONES, PULL UPS
                        ENAL/DISA Y
                        ; SELECTOR DE FLANCO DE LAS INTERRUPCIONES
                        (INTEDG =BIT 6 ) 1=RASING     0=FALLING


;*****************************************************************
;*****************************************************************
;TABLA DE MENSAJES A TRANSMITIR

                ORG 500H
MSG0 DW 'A','T',0XD,0XA,0XCC
        ORG 505H
MSG1 DW 'A','T','+','C','M','G','R','=','1',0XD,0XCC
        ORG 511H
MSG2 DW 'A','T','+','C','M','G','D','=','1',',','4',0XD,0XCC
        ORG 520H
MSG3 DW    'A','T','+','C','M','G','S','=',"",'7','8','7','0','8','0','4','1',
                "",0XD,'E','S','T','A',' ','E','S',' ','U','N','A',' ',
                'P','R','U','E','B','A',0X1A,0XCC
        ORG 550H
MSG4 DW 'A','T','+','C','M','G','S','=',"",'7','8','7','0','8','0','4','1',
        "",0XD,'S','I','S','.',' ','A','C','T','.','-','C','A','R',' ',
        'L','O','C','K','E','D',0X1A,0XCC
        ORG 580H
MSG5 DW 'A','T','+','C','M','G','S','=',"",'7','8','7','0','8','0','4','1',
        "",0XD,'S','I','S','.',' ','A','C','T','.','-','C','A','R',' ',
```

```
        'U','N','L','O','C','K','E','D',0X1A,0XCC
        ORG 0X5B0
MSG6 DW 'A','T','+','C','M','G','S','=',"",'7','8','7','0','8','0','4','1',
        "",0XD,'S','I','S','.',' ','D','E','S','A','C','T','.',0X1A,0XCC

;TABLA DE MENSAJES PARA COMPARACION DE LA RECEPCION

        ORG 5E0H
MSG7 DW    '+','C','M','T','I',':'
        ORG 0X5EA
MSG8 DW '+','C','M','G','R',':'

;*********************************************************************
;*********************************************************************
;MACROS PARA SELECCION DE BANCO

BANK0:      macro               ; select register bank 0
            bcf    STATUS,RP0
            bcf    STATUS,RP1
            endm
;-----------------------------------------------
BANK1:      macro               ; select register bank 1
            bsf    STATUS,RP0
            bcf    STATUS,RP1
            endm
;-----------------------------------------------
BANK2:      macro               ; select register bank 2
            bcf    STATUS,RP0
            bsf    STATUS,RP1
            endm
;-----------------------------------------------
```

```
BANK3:      macro               ; select register bank 3
            bsf    STATUS,RP0
            bsf    STATUS,RP1
            endm
;-------------------------------------------
;*****************************************************************


        ORG 0X0
        GOTO MANAGEMENT

        ORG  0X4
        GOTO ISR

        ORG 0XA
MANAGEMENT:
        CALL  INICIALIZACION
        CALL  FLOWCONTROL
        CALL  CONFIGURACION
        CALL  CLEARRAM
        CALL  RS232TX           ;SE LLAMA A LA RUTINA DE HABILITACION
                                DE RS232TX, SE HABILITA LA TRANSMISION
        CALL  RS232RX           ;SE HABILITA LA RECEPCION DE CARCATERES
                                POR LA USART

WAITINT     NOP                 ;LAZO SIN FIN A LA ESPERA DE LAS
                                INSTERRUPCIONES
        BSF         portc,2     ;CON UN BIT INDICADOR SE SETEA Y SE
                                LIMPIA CONSTANTEMENTE.
        CALL  DELAY
        CALL  DELAY
```

```
        CALL  DELAY
        BCF         portc,2
        CALL  DELAY
        CALL  DELAY
        CALL  DELAY
        GOTO        WAITINT

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

;*****************************************************************
;SUBRUTINAS A UTILIZAR, LECTURA DE FLASH, ESCRITURA DE EEPROM,
HABILITACION DE TX Y RX DE LA USART

;_____________________________________________________________
;RUTINA DE INICIALIZACION DE VARIABLES Y PUERTOS

INICIALIZACION:
        BANK0
        CLRF CHARRX             ;SE LIMPIAN LOS REGISTROS DE PROPOSITOS
                                GENERALES Y PUERTOS
        CLRF CHARTX
        CLRF DTTBFLS
        CLRF MSGLABEL
        CLRF MARCAS
        CLRF MARCA23
        INCF  MARCA23,F
        CLRF RAMTBL
        CLRF EEMARKS
        CLRF RXEEMSG
        CLRF DATAEE
        CLRF markcom
```

```
CLRF CONTA1
CLRF CONTA2
BANK1
MOVLW     b'11111'      ;los 5 bits del puerto A son entradas para
                        interruptores de propositos generales
MOVWF     trisa
MOVLW     B'11110001'   ;EL BIT0 DEL PUERTO B, ES DE INTERRUPCION
                        EXTERNA,
                        LOS BITS DE 1-3 SON SALIDAS PARA
                        INDICACIONES
                        ;LOS BITS 4-7, SON ENTRADAS PARA LOS
                        SENSORES QUE SE TENDRIAN INSTALADOS EN
                        EL AUTO.
MOVWF     trisb
MOVLW     0X7
MOVWF     adcon1        ;configuro el puerto A con entradas digitales y no
                         entradas analogas.
MOVLW     B'10001010'   ;el puerto c es configurado de manera que el
                        bit0=out=DTR, bit1=IN=DSR, bit2=out=RTS,
                        bit3=in=CTS
                  ;bits 4 y 5 son outputs generales; Bit 6 = out = TX
                  and bit7=In=RX.
MOVWF     trisc
CLRF trisd
CLRF trise          ; se configuran los tres bits de puerto e como salidas
                     digitales.
BANK0
CLRF porta          ; SE LIMPIAN LOS PUERTO DE ENTRADAS Y SALIDAS.
CLRF portb
CLRF portc
CLRF portd
```

```
        clrf    porte
        CLRF TXREG
;       MOVLW   0X2
;       MOVWF   MARCA23
        RETURN

;_____________________________________________________________
;RUTINA DE CONFIGURACION CONTROL DE FLUJO

FLOWCONTROL:
        BCF   portc,0           ;se limpia para poner en +12 la linea de DTR
        BCF   portc,2           ;se limpia para poner en +12 la linea de RTS
                                y poder recibir datos del Modem
        RETURN


;_____________________________________________________________
;RUTINA DE DE CONFIGURACION DE INTERRUPCIONES
;INTERRUPCIONES A UTILIZAR, SON LAS DE INT (RB0) AND PORTBCHANGE
(BITS 4-7 PORTB)

CONFIGURACION:
        MOVLW   0XFF
        MOVWF   option_reg  ;SE CONFIGURA EL FLANCO DE LAS
                            INTERRUPCIONES  ASCENDENTE,
                            DESABILITADAS LAS PULL UPS.
        MOVLW   B'01011000' ;CONFIGURACIONES DE INTERRUPECIONES NO
                            PERIFERICAS
        MOVWF   INTCON      ;EL BIT0=BANDERA DE PORTB CHANGE INT
                            (1=OCURRIO);
                            BIT1=BANDERA DE INT EXTERNA
```

```
                        (RB0=OCURRED)
                        ;CON BITS 3 Y 4 HABILITO INTERRUPECIONES
                        DEL PUERTO
                        B, CON EL BIT 5 DESABILITO INTERRUPCION TIMER0
                        ;CON BIT6 HABILITO LAS INTERRUPCIONES
PERIFERICAS,
                        COMO LAS RECEPCION Y TRANSMISION DE LA USART
                        ;BIT 7 HABILITADOR GLOBAL,SE HABILITA LUEGO.
        BANK1
        MOVLW   B'00100000' ;SE HABILITA SOLO LA INTERRUPCION
                        PERIFERICA DE
                        RECEPCION DE LA USART.
        MOVWF   pie1
        BANK0
        BSF  INTCON,7   ;SE HABILITA LAS INTERRUPCIONES GLOBALES
        RETURN

;-------------------------------------------------------
;RUTINA DE HABILITACION DE TX Y RX EN LA USART DEL PIC

RS232TX:
        BANK1
        MOVLW   B'00000110' ;SE CONFIGURA EL PERIFERICO DE
COMUNICACION
                        USART DE MICROCONTROLADOR
        MOVWF   txsta
        MOVLW   D'25'
        MOVWF   spbrg ;SE CONFIGURA EL VALOR NECESARIO PARA TENER
                         UN BOUD RATE DE 9600
        BANK0
        MOVLW   B'10000000' ;SE HABILITA LA TRANSMISION DE LA USART
```

```
        MOVWF       RCSTA
        BANK1
        BSF    txsta,5          ; SE SETEA EL BIT DE INICIO DE TRANSMISION
        RETURN

;-----------------------------------------------------------
RS232RX:
        BANK0
        BSF    INTCON,7     ;SE HABILITA LAS INTERRUPCIONES NO
ENMASCARABLES.
        BSF    RCSTA,4      ; SE HABILITA LA RECEPCION CONTINUA DE
CARACTERES.
        RETURN
;-----------------------------------------------------------

;-----------------------------------------------------------
;RUTINA DE RECEPCION  DE CARACTERES DESDE EL MODEM

INTRXUSART:

WAITRX      BTFSS       PIR1,5       ;SE ESPERERA A QUE LA RECEPCION SE
                                     HAYA COMPLETADO
        GOTO        WAITRX
                        ;ACA DEBERIAMOS REVISAR SI HUBO
                        ERRORES DE RX
        BSF    portc,0      ;UNA VES QUE SE QUE SE HA RECIBIDO
                        UN DATO (KE EN REALIDAD SON 2) DETENGO LA
                        RECEPCION DE DATOS
        MOVFRCREG,W  ; SE LEE EL CONTENIDO DEL REGISTRO DE
                        RECEPCION.
        MOVWF       INDF
```

```
        MOVWF     CHARTX     ; Y SE  GUARDA EN CHARRX
        XORLW     0X0A       ;SE COMPARA PARA VER SI FUE 0XA EL ULTIMO
                             VALOR RECIBIDO
        BTFSC     STATUS,Z
        GOTOcheck          ;SI LO FUE SE SALTA A CHEQUEAR SI FUE EL PRIMER
                             CARACTER O EL SEGUNDO RECIBIDO
        INCF  FSR,F        ;SINO SOLO SE INCREMENTE FSR
        BCF   portc,0      ; SE VUELVE HABILITAR LA RECEPCION DE
CARCATERES
                           HACIA EL MICROCONTROLADOR
        GOTOWAITRX         ; SE SALTA A ESPERAR QUE TERMINA LA SIGUIENTE
                           RECEPCION DE CARCATERES

check:INCF  CONTA1,F       ; CHEQUEA QUE CONTADOR 1 CONCUERDE CON EL
                           VALOR
                            PREFIJADO DE EL REGISTRO MARCA23
        MOVFCONTA1,W       ; SI ESTE CONCUERDA ENTONCES PARA A COMPARAR
                           EL  MESAJE RECIBIDO
        XORWF     MARCA23,W
        BTFSC     STATUS,Z
        GOTO      COMPARE  ;SI NO ES IGUAL EL RESGITRO MARCA23 CON EL
                           CONTA1,  ENTONCES SI EN LA RUTINA DE
                           RECEPCION HASTA
        BCF   portc,0      ;QUE SEAN IGUALES, Y SE VUELVE HABILITAR LA
                           RECEPCION DE CARACTERES.
        GOTO      WAITRX

;*******************************************************************************
;
;*******************************************************************************
```

```
COMPARE: CALL   RAM2        ;SE LLAMA A LA RUTINA PARA LIMPIAR LA
MEMORIA RAM
        MOVLW   RAMTBL      ;SE CARGA EN EL REGISTRO FSR, EL VALOR DE
                            DIRECCION DE LA RAM DONDE SE ALMACENAN
                            LOS CARCATERES
        MOVWF   FSR         ;RECIBIDOS POR LA USART
        MOVLW   ADRMS7L
        MOVWF   MSGLABEL
        CLRF RXEEMSG        ; SE CARGA EN LOS RESGITROS ADECUADOS LA
                            DIRECCION  DE LA MEMORA FLASH EN DONDE
                            SE ALMACENAN LOS
NEXT1       CALL LEERFLASH      ;CARCATERES QUE SE DESEAN
                                COMPARAR, SE LLAMA A LA
                                RUTINA DE LEER FLASH
        MOVFINDF,W          ;SE TRANSIFIERE ESTA DATO A DATAEE
        MOVWF   DATAEE
        CALL  WRITEPROM     ; Y SE GUARDA EN LA MEMORIA EEPROM DEL PIC
        MOVFINDF,W
        XORWF   DTTBFLS,W
        BTFSS   STATUS,Z
        GOTO    SECMESG     ; SI LOS CARACTERES EN ALGUN  MOMENTO NO
                            SON IGUALES ENTONCES SALTA A COMPARAR
                            EL SEGUNDO MENSAJE
        MOVFDTTBFLS,W       ;SINO LOS CARCATERES SE COMPARAN HASTA
                            RECIBIR EL CARACTER 0X3A
        XORLW   0X3A
        BTFSC   STATUS,Z
        GOTO    EQUMS1      ; SI SE RECIBIO ESTE CARACTER ENTONCES EL
                            MENSAJE RECIBIDO FUE IGUAL AL MENSAJE
                            ALMACENADO EN LA FLASH
        INCF  FSR,F
```

```
        INCF  MSGLABEL,F
        INCF  RXEEMSG,F
        GOTONEXT1

SECMESG  MOVLW     RAMTBL     ;SI FUE EL MENSAJE RECIBIDO NO ES
                              IGUAL AL PRIMER MENSAJE GUARDADO EN
                              LA FLASH,
        MOVWF     FSR         ;SE COMPARA CON EL SEGUNDO MENSAJE
        MOVLW     ADRMS8L     ;SE CARGA LA DIRECCION EN DONDE SE
                              ENCUENTRA ALMACENADO EL MENSAJE
        MOVWF     MSGLABEL  ; SE CARGA EN EL REGISTRO MSGLABEL
        MOVLW     0X10
        MOVWF     RXEEMSG
NEXT2       CALL LEERFLASH       ;SE LEE EL VALOR DESDE LA FLASH DEL
                              PIC
        MOVFINDF,W
        MOVWF     DATAEE      ; SE GUARDA EL VALOR LEIDO EN LA EEPROM
DEL PIC
        CALL  WRITEPROM       ; SE LLAMA A LA RUTINA DE ESCRITURA EN LA
EEPROM
        MOVFINDF,W            ;SI EL MENSAJE RECIBIDO NO ES IGUAL AL
                              MENSAJE ALMACENADO EN LA FLASH
        XORWF     DTTBFLS,W
        BTFSS     STATUS,Z
        GOTOCLEAN             ;ENTONCES SALTA A LIMPIAR LOS REGISTROS
                              PARA PREPARAR LA RUTINA PARA UNA UAN
                              PROXIMA VEZ
        MOVFDTTBFLS,W         ; SI NO FALLA NINGUN CARCATER SE ESPERA A
                              QUE  SE RECIBA EL CARACTER 0X3A
        XORLW     0X3A
        BTFSC     STATUS,Z    ; SI SE RECIBIO EL CARACTER ENTONCES QUE
```

```
                                SALTE A EQUMS2
        GOTO    EQUMS2
        INCF  FSR,F
        INCF  MSGLABEL,F
        INCF  RXEEMSG,F
        GOTONEXT2

CLEAN
        movlw 0x1           ;SE SETEA EL REGISTRO MARCA23 CON EL VALOR DE
                            0X1
        movwf MARCA23
        MOVWF   DATAEE      ; SE GUARDA ESTE DATO TAMBIEN EN LA
                            EEPROM DEL PIC
        MOVLW   0X60
        MOVWF   RXEEMSG
        CALL  WRITEPROM     ;SE ESCRIBE ESTE VALOR EN LA EEPROM
        CLRF CONTA1         ;SE LIMPIA EL CONTADOR1
        RETURN

EQUMS1    CALL  RAM1        ;SE LIMPIA LA MEMORIA RAM DEL PIC
        MOVLW   0X1
        MOVWF   PORTD       ; SE CARGA EL VALOR DE 0X1 EN EL PUERTO D Y
E
        MOVWF   porte
        call    DELAY       ; SE HACE UN RETARDO
        clrf    porte       ;LUEGO SE LIMPIA EL REGISTRO E
        MOVLW   0X2         ; SE CARGA LUEGO EN EL RESGITRO MARCA23
                             EL VALOR DE 0X2
        MOVWF   MARCA23
        MOVWF   DATAEE      ;TAMBIEN SE CARGA EN EL REGISTRO DATAEE,
                            PARA GUARDAR EL DATO EN LA EEPROM
```

```
        MOVLW   0X60
        MOVWF   RXEEMSG
        CALL  WRITEPROM     ;SE LLAMA LA RUTINA DE ESCRIBIR EN LA
                            EEPROM
        MOVLW   RAMTBL      ; SE CARGA EL PUNTERO DE LA RAM PARA LA
                            COMPARACION
        MOVWF   FSR         ;SE MUEVE A FSR PARA LA LECTURA DE LAS
                            LOCALIDADES
        CLRF CONTA1
        RETURN


EQUMS2:
        BANK0
        CALL  RAM2
        MOVLW   RAMTBL      ; SE CARGA EL PUNTERO DE LA RAM PARA LA
                            COMPARACION
        MOVWF   FSR         ;SE MUEVE A FSR PARA LA LECTURA DE LAS
                            LOCALIDADES
        movlw 0x1
        movwf MARCA23
        MOVWF   DATAEE      ;TAMBIEN SE CARGA EN EL REGISTRO DATAEE,
                            PARA GUARDAR EL DATO EN LA EEPROM
        MOVLW   0X60
        MOVWF   RXEEMSG
        CALL  WRITEPROM     ;SE LLAMA LA RUTINA DE ESCRIBIR EN LA
                            EEPROM
        CLRF CONTA1
NXTCHAR3 MOVF INDF,W
        XORLW   0X2A        ; SE COMPARA EL VALOR QUE TIENE FSR CON EL
                            VALOR DE 0X2A QUE ES *
```

```
BTFSS     STATUS,Z   ; SE CHEQUEA QUE FUE CERO EL RESULTADO
GOTO      INCFSR     ; SINO ES EL ASTERICO QUE BUSQUE EN EL
                     SIGUIENTE CARACTER
INCF  FSR,F          ; PARA ESO INCREMENTO FSR
MOVFINDF,W           ;SE CARGA EL CARCATER EN W
XORLW     0X2A       ;SE COMPARA CON EL 0X2A ASTERISCO PARA
                     VER SI SE RECIBIO EL SEGUNDO CARACTER
BTFSS     STATUS,Z   ; SE REVISA
GOTO      NONE
INCF  FSR,F          ;SI SE RECIBIO EL SEGUNDO ASTERISCO
                     ENTOCES SE INCREMENTA FSR
MOVFINDF,W           ; SE CARGA EL VALOR DEL SIGUENTE CARACTER
                     EN W
XORLW     0X31       ; SE COMPARA CON EL 1
BTFSC     STATUS,Z
GOTOFUE_1            ; SI FUE QUE SALTE A FUE_1
MOVFINDF,W           ;VUELVO A CARGAR EL VALOR AL W
XORLW     0X32       ;AHORA LO COMPARO CON 2
BTFSC     STATUS,Z
GOTO      FUE_2      ;SI FUE 2 ENTONCES QUE SALTE A FUE_2
MOVFINDF,W
XORLW     0X33       ;SE COMPARA CON EL NÚMERO 3
BTFSC     STATUS,Z
GOTO      FUE_3           ;SALTA A FUE_3
MOVFINDF,W
XORLW     0X34       ;SE COMPARA CON EL NÚMERO 4
BTFSC     STATUS,Z
GOTOFUE_4            ;SALTA A FUE_4
MOVFINDF,W
XORLW     0X35       ;SE COMPARA CON EL NÚMERO 5
BTFSC     STATUS,Z
```

```
        GOTOFUE_5
        MOVFINDF,W
        XORLW   0X36        ;SE COMPARA CON EL NÚMERO 6
        BTFSC   STATUS,Z
        GOTOFUE_6
        GOTO    NONE

INCFSR  MOVFFSR,W       ;ACA CAE SI NO SE HA RECIBIDO EL PRIMERO
ASTERISCO
        XORLW   0X7F        ; SE MIRA SI YA SE LLEGO AL FINAL DE LA RAM
                            DEL BANCO 0
        BTFSC   STATUS,Z
        GOTO    NONE
        INCF  FSR,F         ;SINO HA LLEGADO QUE SE INCREMENTE EL FSR
        GOTO    NXTCHAR3    ;Y QUE SALTE A BUSCAR EL SIGUIENTE
                            CARACTER

FUE_1   movlw 0x2
        movwfporte      ;SE ESCRIBIR EN EL PUERTO E EL VALOR DE 0X2
        MOVLW   RAMTBL      ; SE CARGA EL PUNTERO DE LA RAM PARA LA
                            COMPARACION
        MOVWF   FSR
        RETURN

FUE_2
        movlw 0x3
        movwfporte          ;SE ESCRIBIR EN EL PUERTO E EL VALOR DE 0X3
        MOVLW   RAMTBL      ; SE CARGA EL PUNTERO DE LA RAM PARA LA
                            COMPARACION
        MOVWF   FSR
        RETURN
```

```
FUE_3
        movlw 0x4
        movwf porte             ;SE ESCRIBIR EN EL PUERTO E EL VALOR DE 0X4
        MOVLW   RAMTBL          ; SE CARGA EL PUNTERO DE LA RAM PARA LA
                                COMPARACION
        MOVWF   FSR
        RETURN

FUE_4
        movlw 0x5
        movwf porte             ;SE ESCRIBIR EN EL PUERTO E EL VALOR DE 0X5
        MOVLW   RAMTBL          ; SE CARGA EL PUNTERO DE LA RAM PARA LA
                                COMPARACION
        MOVWF   FSR
        RETURN

FUE_5
        movlw 0x6
        movwf porte             ;SE ESCRIBIR EN EL PUERTO E EL VALOR DE 0X6
        MOVLW   RAMTBL          ; SE CARGA EL PUNTERO DE LA RAM PARA LA
                                COMPARACION
        MOVWF   FSR
        RETURN

FUE_6
        movlw 0x7
        movwf porte             ; SE ESCRIBIR EN EL PUERTO E EL VALOR DE 0X7
        MOVLW   RAMTBL          ; SE CARGA EL PUNTERO DE LA RAM PARA LA
                                COMPARACION
        MOVWF   FSR
```

```
        RETURN

NONE
        movlw 0x7
        movwf porte             ;SE ESCRIBIR EN EL PUERTO E EL VALOR DE 0X7
        MOVLW   RAMTBL  ; SE CARGA EL PUNTERO DE LA RAM PARA LA
                                COMPARACION
        MOVWF   FSR
        RETURN


;-------------------------------------------------------
;RUTINA PARA ESCRIBIR TODA LA RAM EN LA EEPROM


RAM1 MOVLW      RAMTBL
        MOVWF   FSR
        MOVLW   0XE0
        MOVWF   RXEEMSG
OTROREG1 MOVF INDF,W
        MOVWF   DATAEE
        CALL  WRITEPROM
        MOVF FSR,W              ;ACA CAE SI NO SE HA RECIBIDO EL PRIMERO
ASTERISCO
        XORLW   0X7F    ; SE MIRA SI YA SE LLEGO AL FINAL DE LA RAM
                                 DEL BANCO 0
        BTFSC   STATUS,Z
        RETURN
        INCF  FSR,F             ;SINO HA LLEGADO QUE SE INCREMENTE EL FSR
        INCF  RXEEMSG,F
        GOTO    OTROREG1 ;Y QUE SALTE A BUSCAR EL SIGUIENTE
```

```
                                    CARACTER

RAM2 MOVLW      RAMTBL
     MOVWF      FSR
     MOVLW      0X70
     MOVWF      RXEEMSG
OTROREG2 MOVF   INDF,W
     MOVWF      DATAEE
     CALL  WRITEPROM
     MOVF  FSR,W            ;ACA CAE SI NO SE HA RECIBIDO EL PRIMERO
                             ASTERISCO
     XORLW      0X7F        ; SE MIRA SI YA SE LLEGO AL FINAL DE LA RAM
                            DEL BANCO 0
     BTFSC      STATUS,Z
     RETURN
     INCF  FSR,F            ;SINO HA LLEGADO QUE SE INCREMENTE EL FSR
     INCF  RXEEMSG,F
     GOTO       OTROREG2 ;Y QUE SALTE A BUSCAR EL SIGUIENTE
                            CARACTER

;-------------------------------------------------------
;RUTINA DE LIMPIAR RAM

CLEARRAM:
     MOVLW      RAMTBL
     MOVWF      FSR
OTROREG3 CLRF   INDF
     MOVF  FSR,W            ;ACA CAE SI NO SE HA RECIBIDO EL PRIMERO
ASTERISCO
     XORLW      0X7F        ; SE MIRA SI YA SE LLEGO AL FINAL DE LA
                            RAM DEL BANCO 0
```

```
        BTFSC       STATUS,Z
        RETURN
        INCF  FSR,F             ;SINO HA LLEGADO QUE SE INCREMENTE EL FSR
        GOTO        OTROREG3 ;Y QUE SALTE A BUSCAR EL SIGUIENTE
                                CARACTER


;-------------------------------------------------------
;RUTINA DE LECTURA DE FLASH

LEERFLASH:

        MOVLW       ADRMSH          ; PARTE ALTA DE LA DIRECCION DONDE SE
                                    ENCUENTRAN LOS MENSAJES
        BANK2
        MOVWF       eeadrh          ; SE GUARDA EN EL FSR
                                    CORRESPONDIENTE
        BANK0
        MOVF        MSGLABEL,W      ; SE CARGA EL CONTENIDO DEL REGISTRO
                                    A W
        BANK2
        MOVWF       eeadr           ; LUEGO SE CARGA EN EL FSR
                                    CORRESPONDIENTE A LA PARTE BAJA DE
                                    LA DIRECCION
        BANK3                       ; Bank 3
        BSF   eecon1, 7             ; SE APUNTA A LA MEMORIA DE PROGRAMA
        BSF   eecon1, 0             ; SE HABILITA LECTURA DE MEMORIA
        NOP                         ; SECUENCIA OBLIGADA
        nOP
        BANK2                       ; Bank 2
        MOVF        eedata, W       ; SE GUARDA EL DATO LEIDO EN W
```

```
        BANK0
        MOVWF   DTTBFLS     ; Y SE GUARDA EN LA RAM
        RETURN
;-------------------------------------------------------


;-------------------------------------------------------
;RUTINA PARA ESCRIBIR EN LA EEPROM

WRITEPROM:
        BANK0
        MOVF    RXEEMSG,W       ;SE TRANSFIERE A W LA DIRECCION EN
                                DONDE SE DESEA ESCRIBIR
        BANK2
        MOVWF   eeadr           ;SE GUARDA EN EL REGISTRO ADECUADO
        BANK0
        MOVF    DATAEE,W  ;SE CARGA A W EL DATO QUE SE DESEA
                          QUEMAR  EN LA EEPROM
        BANK2
        MOVWF   eedata          ;SE CARGA EN EL REGISTRO ADECUADO
        BANK3
        BCF   eecon1,7          ;SE APUNTA A LA MEMORIA DE DATOS
        BSF   eecon1,2          ;SE HABILITA LA ESCRITURA EN EL EEPROM
        MOVLW   55h             ;SECUENCIA OBLIGADA
        MOVWF   eecon2          ;ESCRIBIR 55H EN EECON2
        MOVLW   0xAA            ;SECUENCIA OBLIGADA
        MOVWF   eecon2          ;ESCRIBIR AAH EN  EL EECON2
        BSF   eecon1,1          ;SE SETEA BIT PARA HABILITAR LA ESCRITURA
                                EN LA EEPROM
        BANK3
WAITWR  BTFSC       eecon1,1    ;ESPERAMOS QUE LA ESCRITURA FINALICE
```

```
        GOTO        WAITWR      ;SALTA A ESPERAR
        BANK0
        RETURN

;-------------------------------------------------------

;RUTINA PARA LEER EN LA EEPROM

READEPROM:
        BANK0
        MOVF RXEEMSG,W      ;SE TRANSFIERE A W LA DIRECCION EN DONDE
                             SE DESEA ESCRIBIR
        BANK2
        MOVWF       eeadr       ;SE GUARDA EN EL REGISTRO ADECUADO
        BANK3
        BCF   eecon1,7          ;SE APUNTA A LA MEMORIA DE DATOS
        BSF   eecon1,0          ;SE SETEA BIT PARA HABILITAR LA LECTURA EN
                             LA EEPROM
        BANK2
        MOVFeedata,w
        BANK0
        MOVWF       DATAEE
        RETURN
;-------------------------------------------------------
;SUBRUTINA QUE GENERA DELAY

DELAY:      MOVLW       0XFF  ;SE SETEE EL VALOR DEL CONTADOR 1 Y 2
        MOVWF       CONTA1
        MOVWF       CONTA2
DECONTA1 DECF CONTA1,1    ;SE DECREMENTA ESTE VALOR EN EL
                             CONTADOR 1
```

```
        MOVF CONTA1,W       ;SE ESPERA A QUE CONTADOR 1 LLEGUE A 0
        BTFSS   STATUS,Z    ;SI LO HACE SE DECREMENTA EL CONTADOR 2
        GOTO DECONTA1       ; SI NO ES ASI SE VUELVE A DECREMENTAR
                            EL CONTADOR 1
        DECF CONTA2,1       ;SE  DECRMENTA CONTADOR 2
        MOVF CONTA2,W       ; SE COMPARA SI ES CERO
        BTFSC   STATUS,Z
        RETURN              ; SALE DE LA RUTINA DELAY SI CONTADOR
                             2 ES CERO
        MOVLW   0XFF        ; SI NO LO ES SE VUELVE A SETEAR EL
                            CONTADOR 1 CON EL VALOR 0XFF
        MOVWF   CONTA1
        GOTODECONTA1        ; SE REGRESA A REALIZAR EL DECREMENTO

;-------------------------------------------------------
;RUTINA DE SERVICIO DE INTERRUPCION.

ISR:    clrf    CONTA1      ;SE LIMPIA EL CONTADOR1
        bsf     portc,0     ;SE DESHABILITA LA RECEPCION DE
                            CARCATERES.
        BTFSC   PIR1,5      ;SE COMPRUEBA SE SE DIO LA INTERRUPCION
                            DE LA USART
        CALL  INTRXUSART    ;SI FUE ASI SE SALTA ALA RUTINA DE LS
                            INTERRUPCION
        BCF   PORTC,0       ;SE VUELVE HABILITAR LA RECEPCION DE
                            CARACTERES
        RETFIE

        END
```

5.7 Diagrama en bloque del Circuito Microcontrolador

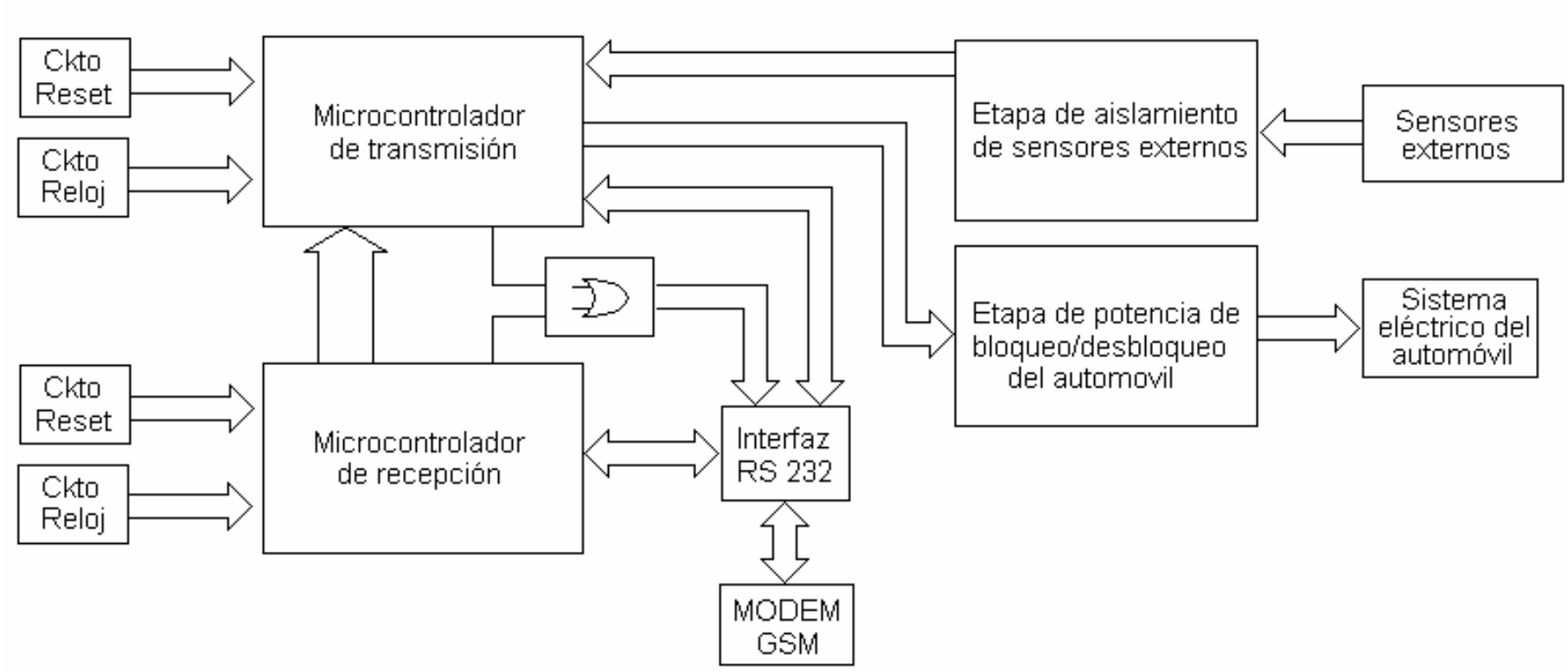


Figura 5.14 Diagrama en bloque de la etapa controladora

5.8 Diseño de diagrama Esquemático del Circuito Microcontrolador

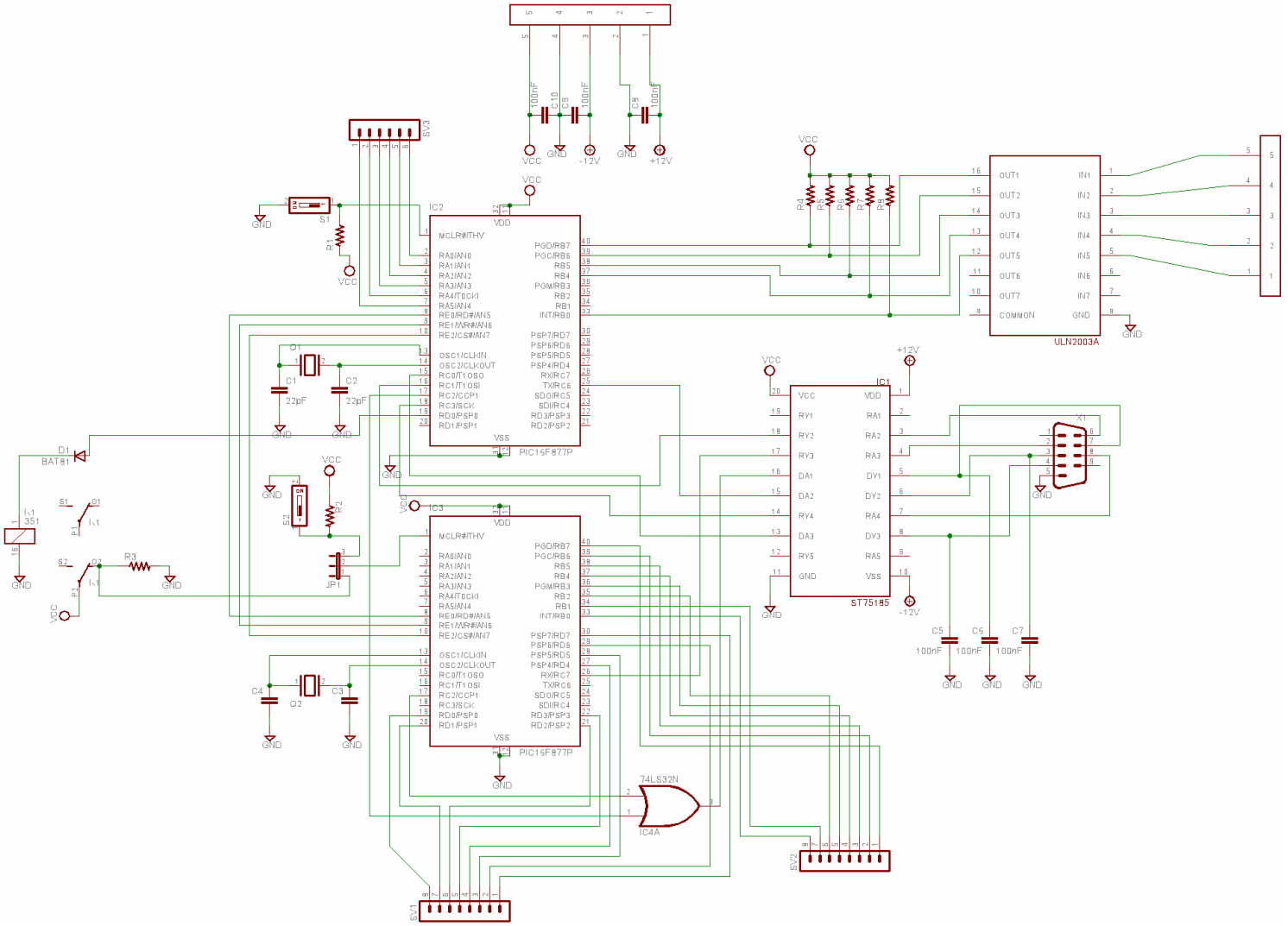


Figura 5.15 Circuito esquemático comunicador del PIC y el Modem

La figura 5.15 muestra el circuito esquemático del hardware encargado de establecer la comunicación entre el Modem y el PIC así como también el que realiza las activaciones o desactivaciones físicas de los dispositivos. Otra tarea que tiene a su cargo el circuito es la del monitoreo de las señales de los sensores de manera que si alguno sensor presenta un estado de alerta esta información se interpreta para luego tomar una decisión de que acción tomar.

El circuito consta de dos microcontroladores PIC16F877A, encargados de la recepción y transmisión de los caracteres ASCII hacia y desde el Modem, se han utilizado dos microcontroladores para independizar la recepción de la transmisión con el fin de que los tiempos de respuestas de los microcontroladores se reduzca. También se utiliza un driver de RS232 el ST75185, el cual es múltiple controlador de RS232 capaz de manejar todas las líneas del estándar RS232 para un DTE.

Además de esto se cuenta cada microcontrolador su respectivo circuito de Reset y Oscilador in dependientes, así como también una compuerta OR que se utiliza para una condición lógica entre dos líneas una de cada microcontrolador que se utiliza para el control de flujo con el Modem. Se han colocado dos conectores uno para las alimentaciones de +5, -12, +12 y GND, el otro conector es utilizado para las líneas físicas que se desean leer o controlar para sensores y activación de dispositivos.

*5.*9 Diseño del circuito programador del microcontrolador.

Para poder guardar toda la información de la programación del PIC y los contenidos de los mensajes que deben ser enviados al Modem GSM utilizado, se ha diseñado un sistema de programación para PIC, lo cual permite poder configurarlo, guardar el software que ejecutará en el proceso y además poder guardar en memoria EEPROM la información valiosa que no se desea perder una vez desconectada la alimentación.

Es por eso que se ha diseñado el siguiente sistema de programación para PIC el cual consta de componentes electrónicos tales como compuerta inversoras,

capacitores, transistor, resistencias, indicadores LED para verificar su estado y un regulador de voltaje para proveer alimentación a los diferentes dispositivos.

El circuito diseñado consta con una interfaz a puerto paralelo, la cual le permite comunicarse con el software utilizado para la programación de PIC. A continuación se presenta una breve explicación del sistema.

El interruptor permite determinar que tipo de PIC se va a programar. Colocándola en la posición superior se pueden programar microcontroladores pequeños y medianos (de 8 y 18 pines) mientras que colocándola en la posición inferior se pueden programar microcontroladores grandes (de 28 y 40 pines).

A continuación se presenta el circuito eléctrico del programador de PIC ha implementar.

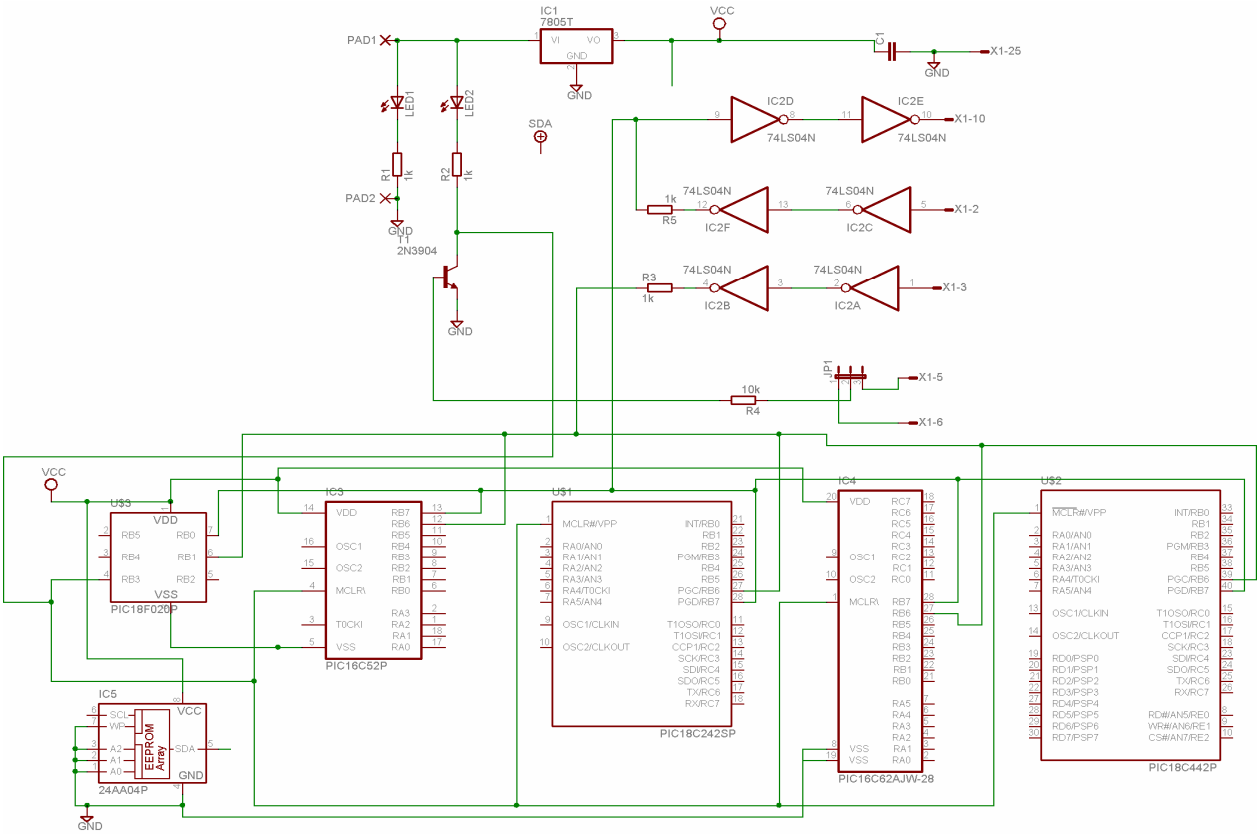


Figura 5.16 Circuito eléctrico del programador del PIC

Uno de los LED marcado permite observar que el sistema se encuentra alimentado mientras que el otro LED conectado al transistor se enciende indicando que es seguro colocar o quitar un PIC y se apaga por instantes breves cuando una lectura o programación de un PIC está en curso. Mientras este último LED este apagado no se debe quitar o insertar ningún colocar en la base.

El funcionamiento del circuito es muy simple: los pines del puerto paralelo 2, 3, 5 y 10 permiten interconectar el circuito con la PC. El pin 2 es el encargado de traer los datos (desde la PC hacia el integrado). El pin 3 es el envío de los pulsos de reloj (desde la PC hacia el integrado). Por lo que el pin 10 permite a la PC leer los datos desde el programador. El pin 5, por último, es el encargado de controlar la tensión de programación (Vpp) necesaria para cuando se quiere leer o escribir en un PIC.

Los microcontroladores PIC se programan utilizando el mismo protocolo que las memorias EEPROM seriales, para que el programador sirva tanto para PIC's como para memorias. La tensión de programación VPP es necesaria para indicarle al PIC que se desea leerlo o programarlo. Si este pin (que es compartido con la entrada de RESET del micro) se lleva a 0 V el PIC sufre un reset, si ponemos el pin en alto (5v) el PIC trabaja normalmente mientras que si se lleva el pin a 12v el PIC se inicializa en modo programación, quedando dos de los pines de E/S destinados a datos (SDA) y reloj (SCL).

El integrado 74LS04 está formado internamente por seis buffers inversores. Estos permiten por un lado obtener niveles TTL a su salida y por el otro no cargar de forma excesiva al puerto. Algunos programadores, como el NOPPP utilizan diodos y resistencias para conectar el PIC directamente el puerto paralelo. Esto funciona en muchas computadoras con fuentes poderosas pero en la mayoría de las computadoras portatiles que no disponen de tanta corriente el funcionamiento es errático o directamente no funciona. Gracias a la utilización de este buffer se puede utilizar el

circuito en cualquier puerto paralelo ya sea de una computadora de escritorio o en una portátil. Se colocan las compuertas en serie para obtener a la salida el mismo nivel de entrada, sin invertir. Las resistencias de 1K dan seguridad al sistema para evitar que circule corriente excesiva.

El control de la tensión de programación lo efectúa el transistor NPN. Estando el pin 5 del puerto paralelo a masa (en 0) se tendrá al transistor abierto por lo que la corriente proveniente de +V (12v) pasará por el diodo LED el cual no encenderá y se portará como un diodo común polarizado en directa, pasará por la resistencia limitadora de corriente del LED la cual no ofrecerá mucha resistencia y será inyectada al PIC en su terminal MCLR/VPP. Poniendo en 1 el bit que controla el pin 5 del puerto paralelo, en cambio, el transistor se cierra y hace circular masa hacia el PIC haciendo, además, encender el LED al quedar a masa el otro extremo de la resistencia limitadora de corriente.

El circuito requiere como única alimentación 12V con una corriente de 200mA. Puede usarse cualquier fuente universal siempre que se respete la polaridad. De tener una fuente de más tensión (13.5v como mucho) no hay problema, se la puede utilizar sin inconvenientes. No será necesario que la fuente sea regulada. Si se tiene una fuente de 12V con más corriente a 1A o incluso mayor se la puede utilizar también sin inconvenientes.

El diseño del circuito impreso se realizó en base al circuito eléctrico previamente presentado y a continuación se muestra la disposición de los componentes electrónicos en dicho circuito impreso.

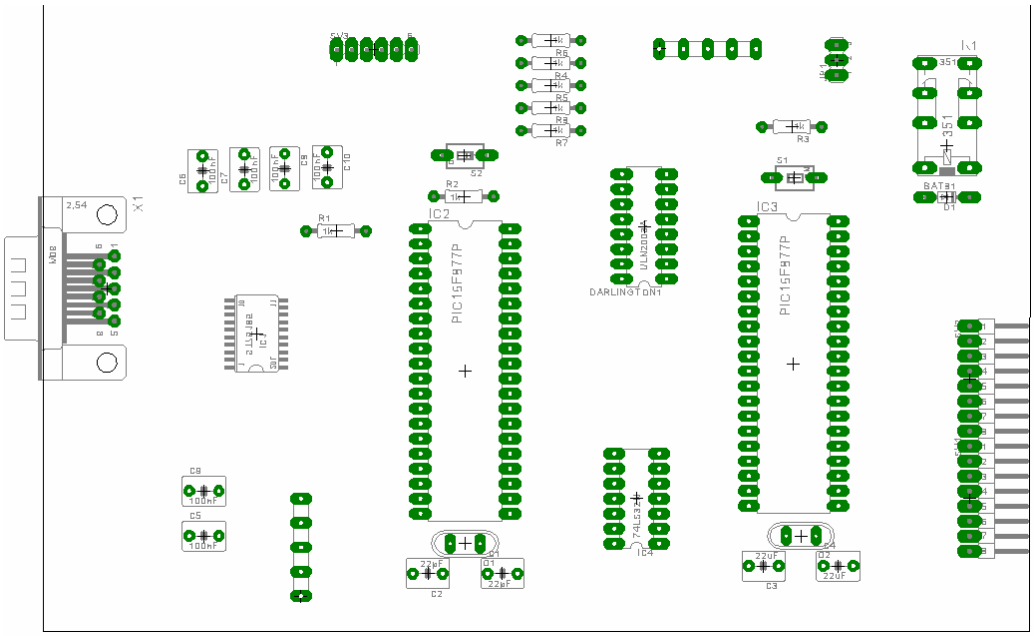


Figura 5.17 Ubicación de elementos en tableta impresa del programador del PIC

En la siguiente imagen se muestra el diseño de lado de las pistas del circuito impreso que se implementa para poder desarrollar el programador de microcontroladores, la imagen siguiente es la contraparte de la imagen anterior en la que se presentan la disposición de los elementos en la tarjeta impresa.

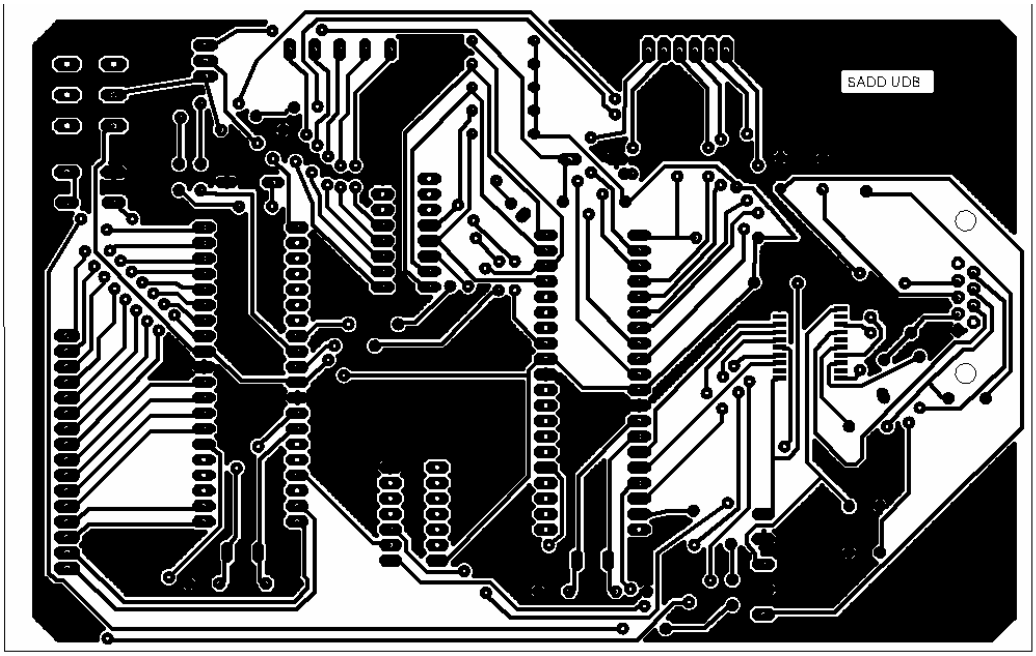


Figura 5.18 Circuito impreso del programador del PIC

Para el diseño del circuito del circuito eléctrico e impreso del circuito programador de microcontroladores se utilizó el programa de diseño electrónico Tagle Layout 4.11 lo el cual es de gran uso también para el diseño electrónico del sistema controlador principal del proyecto que se presenta posteriormente.

El programa que utilizado para el programador es el IC-Prog (Figura 5.18) dado que reúne varias características bastantes interesantes

- Es muy fácil de usar
- Permite ver el ASM del programa que se esta por cargar en el PIC (si lo obtiene desde el HEX)
- Tiene varios idiomas, entre ellos español
- Dispone de cinco espacios de memoria (Buffers) para poder tener hasta cinco programas simultáneos.
- Dentro de una única ventana reúne memoria de programa, memoria EEPROM y bits de configuración.
- Hay actualizaciones periódicas con funciones nuevas y problemas resueltos.
- Funciona tanto bajo Windows95 como Windows XP así como en versiones intermedias.

Para configurarlo sólo es necesario presionar F3 y especificar el tipo de hardware programador (seleccionar ProPic 2), indicar el puerto paralelo al cual está conectado y establecer como método de transferencia de información al puerto Direct I/O (en caso de usar Windows 95, 98 o ME) o Windows API (en caso de usar Windows NT, 2000 o XP). El retardo de I/O establecerlo en 10 que es un valor que funciona siempre. Se puede ir reduciendo y probando para lograr el menor retardo posible y así obtener la mayor velocidad de operación. En tanto se debe indicar como única línea invertida la de MCLR. Las demás líneas son normales.

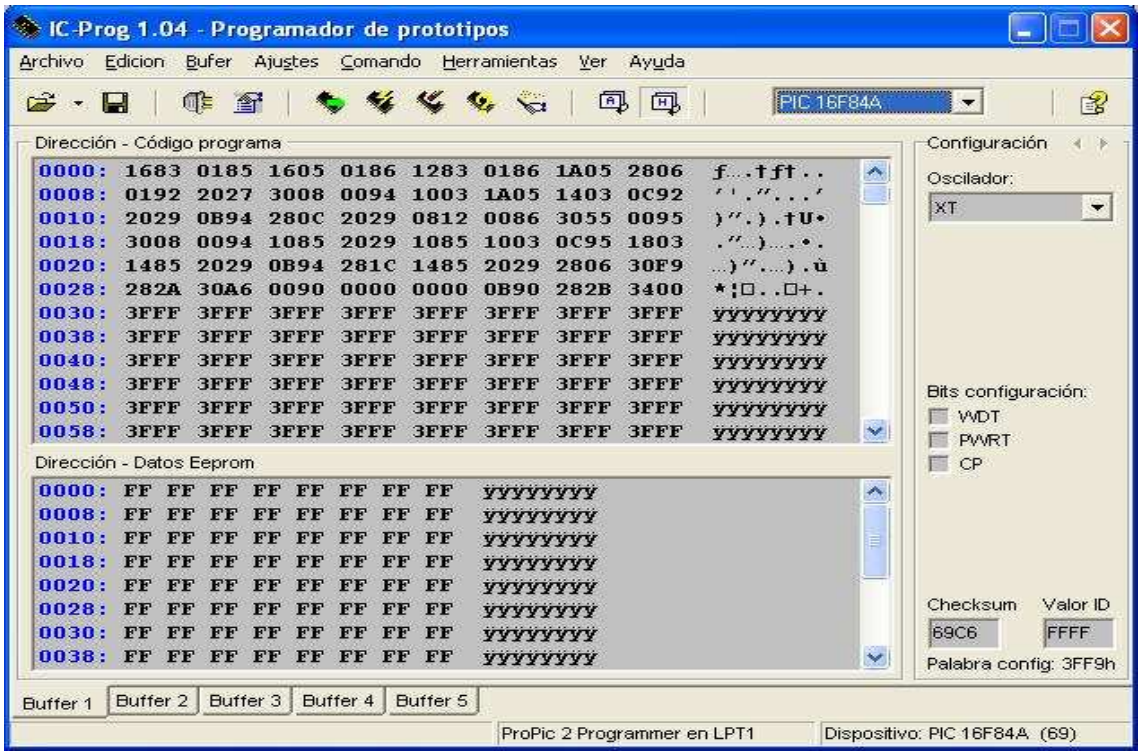


Figura 5.19 Ventana del software programador de IC-Prog

El programa no requiere instalación, bastará con descomprimir los únicos dos archivos que lo conforman (el EXE y el SYS) en cualquier carpeta. Una vez hecho esto ejecutar el EXE con doble clic y establecer la configuración.

Si al momento de terminar la configuración del programa los dos LED's están encendidos es señal de buen funcionamiento. Dentro del menú ajustes hay una opción para probar el funcionamiento del programador.

Capítulo 6:
CARGA DEL MIDLET AL TELÉFONO MÓVIL

6. CARGA DEL MIDLET AL TELEFONO MÓVIL.

6.1 Descripción del capítulo

El propósito de este capítulo es mostrar cuales son los mecanismos para descargar o cargar los midlets hacía un teléfono móvil, específicamente hacia el Motorola V600. Además de establecer cuales son los requerimientos impuestos hacia este teléfono móvil.

El mecanismo de descarga del Midlet es OTA, y los mecanismos de carga son el Bluetooth, IR (infrarrojo) y los cables (serial 232 o USB). Para los mecanismos de carga se necesita tener almacenado el midlet en la PC y por supuesto tener en el teléfono móvil y la PC la disponibilidad de emplear uno de esos mecanismos.

En este capítulo se centrará en explicar los mecanismos: OTA (este es general para todos los teléfonos móviles que soportan J2ME) y el empleo de cable USB (este es específico para el Motorola V600), y sus requerimientos.

6.2 OTA [3]

Las aplicaciones realizadas con J2ME están pensadas para que puedan ser descargadas a través de una conexión a internet. El medio empleado para garantizar esta descarga recibe el nombre de OTA[1].

Como se ha mencionado en capítulos anteriores, una aplicación J2ME está formada por un archivo JAR[1] que es el que contiene a la aplicación en sí y un archivo JAD[2] que contiene diversa información sobre la aplicación.

[1] OTA: Over The Air

6.2.1 Requerimientos Funcionales

Los dispositivos deben proporcionar mecanismos mediante los cuales se pueda encontrar los *MIDlets* que se deseen descargar. En la mayoría de los casos, se encontrarán los *MIDlets* a través de un navegador WAP o a través de una aplicación residente escrita específicamente para identificar *MIDlets*.

El programa encargado de manejar la descarga y ciclo de vida de los *MIDlets* en el dispositivo se llama Gestor de Aplicaciones o AMS. Un dispositivo que posea la especificación MIDP debe ser capaz de:

- Localizar archivos JAD vinculados a un *MIDlet* en la red.
- Descargar el MIDlet y el archivo JAD al dispositivo desde un servidor usando el protocolo HTTP 1.1 u otro que posea su funcionalidad.
- Enviar el nombre de usuario y contraseña cuando se produzca una respuesta HTTP por parte del servidor 401 (*Unauthorized*) o 407 (*Proxy Authentication Required*).
- Instalar el *MIDlet* en el dispositivo.
- Ejecutar *MIDlets*.
- Permitir al usuario borrar *MIDlets* instalados.

6.2.2 Localización de la Aplicación

El descubrimiento de una aplicación es el proceso por el cual un usuario a través de su dispositivo localiza un *MIDlet*. El usuario debe ser capaz de ver la descripción del *MIDlet* a través de un enlace que, una vez seleccionado, inicializa la instalación del *MIDlet*. Si éste enlace se refiere a un archivo JAR, el archivo y su URL[3] son enviados al AMS del dispositivo para empezar el proceso de instalación. Si el enlace se refiere a un archivo JAD se realizan los siguientes pasos:

---

[1] Java Archive
[2] Java Application Descriptor
[3] Uniform Resource Locutor

1. El descriptor de la aplicación (archivo JAD) y su URL son transferidos al AMS para empezar la instalación. Este descriptor es usado por el AMS para determinar si el *MIDlet* asociado puede ser instalado y ejecutado satisfactoriamente.
2. Este archivo JAD debe ser convertido al formato *Unicode* antes de ser usado. Los atributos del JAD deben ser comprensibles, acorde con la sintaxis de la especificación MIDP, y todos los atributos requeridos por la especificación MIDP deben estar presentes en el JAD.
3. El usuario debería de tener la oportunidad de confirmar que desea instalar el *MIDlet*. Asimismo debería de ser informado si se intenta instalar una versión anterior del *MIDlet* o si la versión es la misma que ya está instalada. Si existen problemas de memoria con la ejecución del *MIDlet* se intentarían solucionar liberando componentes de memoria para dejar espacio suficiente.

6.3 Empleo de cable USB Motorola

Para poder subir los midlets hacia el teléfono móvil Motorola V600, se utiliza el programa que Motorola pone a disposición: *Motorola MIDway 2.8*, la cual se obtiene en un CD instalador al comprar el cable USD o se obtiene por medio de internet.

Para cargar los midlets por cable USB, existe un pequeño problemita, debido a que en la mayoría de los teléfonos móviles de Motorola no se encuentra habilitado, o más bien se encuentra bloqueado, el *Cargador de Aplicaciones Java* (JAL[1]) y para poder desbloquearlo existen dos formas:

1. Pagando al distribuidor de teléfonos móviles local
2. O empleando programas que se encuentran en internet que sirven para liberar el teléfono móvil.

[1] Java Application Loader

Pues bien, la mayoría de los programas con este objetivo que se encuentran en la web tienen una opción para desbloquear el Java App. Loador. Generalmente esta opción se encuentra en la sección de java. Ejemplos de estos programas son: PST Phone Programmer, Motorola Service Software, Seemplayer, Fitter P2K tool Mod y otros. El empleado para subir la aplicación ha sido Fitter P2K Tool debido a que no se hace mucho procedimiento, a comparación de los otros.

6.3.1 Empleo del P2KTools Mod

Después de instalar el P2KTools y ejecutarlo es necesario que se conecte el cable USB al V600 y a la computadora, posteriormente se selecciona Conectar, si se han instalados correctamente los *drivers* del cable USB, el programa reconocerá el modelo del teléfono móvil.

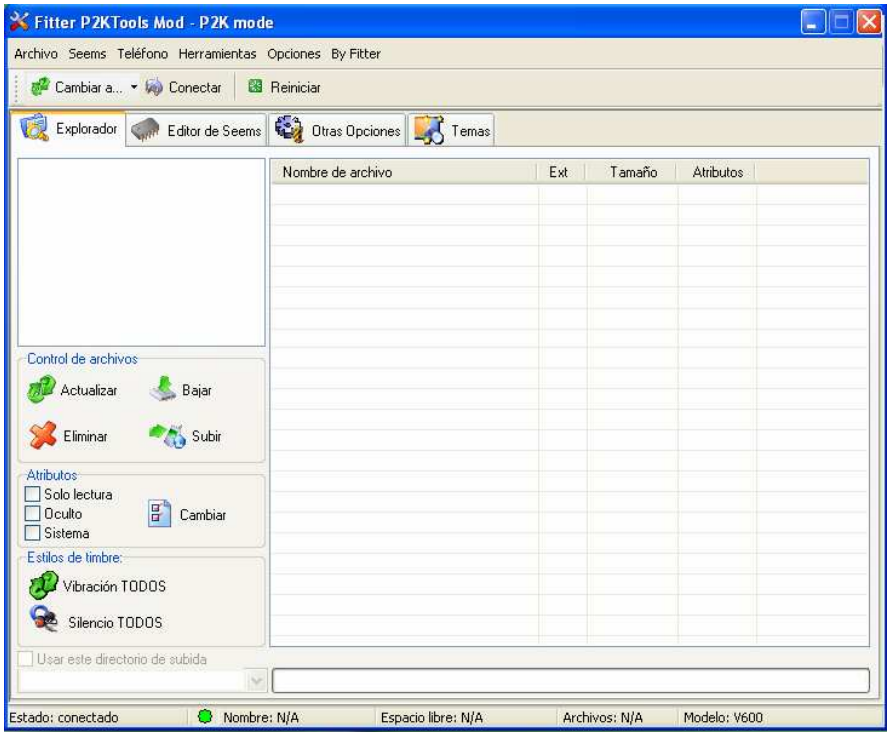


Figura 6.1 Pantalla principal del Fitter P2KTools

Una vez ya comunicado con el teléfono móvil, se selecciona *Otras Opciones* y en las opciones de la seem 0032_0001 se selecciona java. Se da clic en obtener para leer las funciones activas en el Motorola V600, se observa si la

función *Cargador de aplicaciones Java* se encuentra seleccionada y si no lo está se debe activar, posteriormente se presiona *Aplicar*.

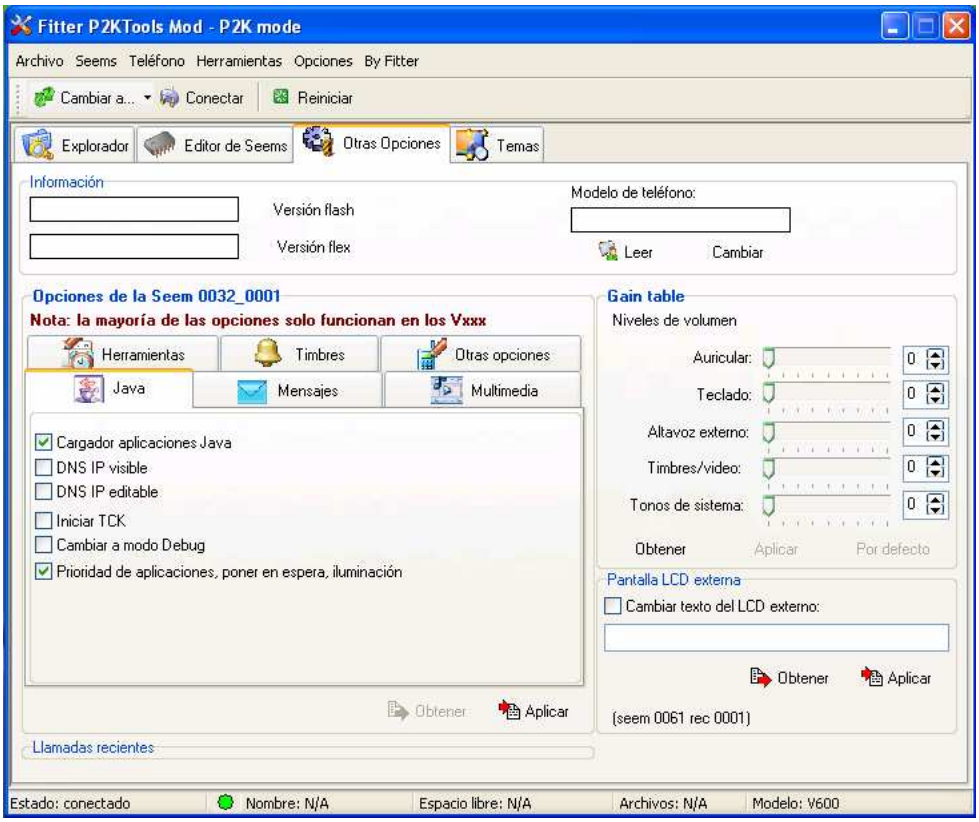


Figura 6.2 Desbloqueo de la función Cargador de Aplicaciones Java

Ahora para que dicha función esté activa en el móvil es necesario reiniciarlo, por lo que se apaga y enciende el teléfono móvil o simplemente se da clic en Reiniciar.

Ahora estando ya desbloqueado el Cargador de Aplicaciones Java se utilizará el *Motorola MIDway 2.8*

6.3.2 Empleo del Motorola MIDway 2.8

Para poder cargar correctamente el MIDlet en el Motorola V600 es necesario seguir los siguientes pasos:

1. En el teléfono móvil, se activa la función Cargador Aps Java, su ruta de acceso es: Configuración > Configurar java > Cargador Aps java. Posteriormente el móvil pedirá que se conecte el cable (a la computadora también). Después de conectarlo se observará en la pantalla del móvil “Enlace JAL está activo”.
2. Asumiendo que ya está instalado el MIDway, se ejecuta y una vez abierto dará un error porque aún no está configurado el puerto com en la que está el cable y ni la velocidad

Figura 6.3 Pantalla inicial del Motorola MIDway

3. Se configura el puerto y la velocidad. Si estos datos se desconocen, se observan por medio de darle clic con el botón derecho en el icono de Mi PC, y se selecciona la siguiente ruta propiedades > hardware > administrador de dispositivos > modems> motorola > propiedades, y se observa el puerto com y la velocidad. Ahora el programa será funcional.

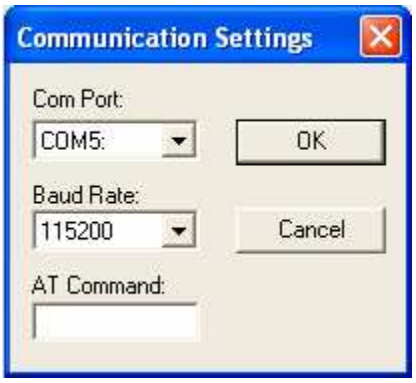


Figura 6.4 Configuración de Comunicación del MIDway

4. Se selecciona y abre el midlet, por medio de “Open Jad” o File > Open JAD. Y se presentará en pantalla toda la descripción y característica de la aplicación

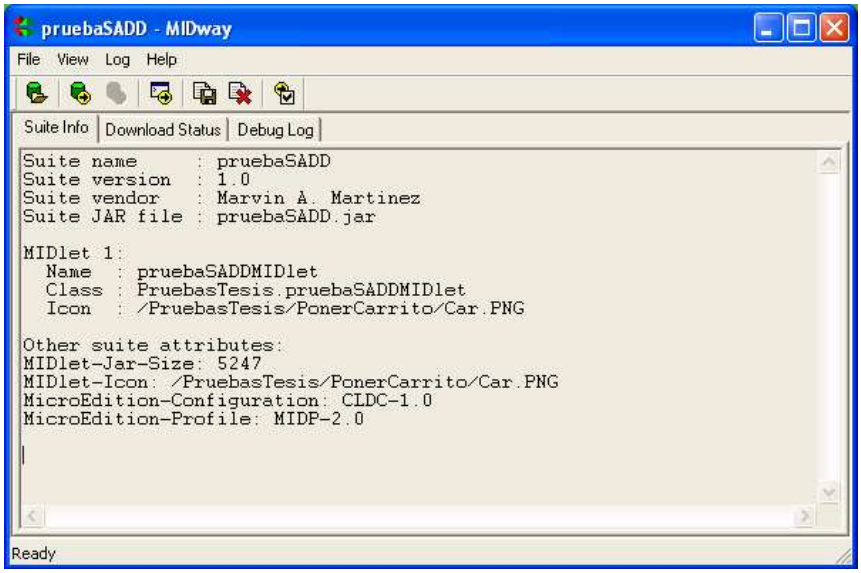


Figura 6.5 Pantalla de Información del MIDlet

5. Se selecciona “SEND JAD” o File > Send JAD. Para subir la aplicación al Motorola V600
6. En la pantalla del teléfono se observará la información del midlet. Y se selecciona BAJAR

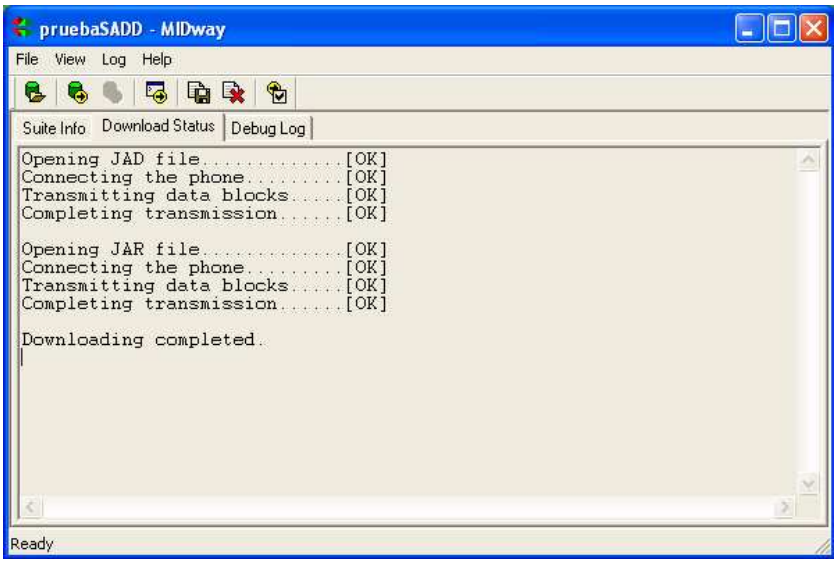


Figura 6.6 Informe de descarga del MIDway

7. Ahora se está listo para ejecutar la aplicación

6.4 Instalación de MIDlets [3]

La instalación de la aplicación es el proceso por el cual el *MIDlet* es descargado al dispositivo y puede ser utilizado por el usuario. Cuando existan múltiples *MIDlets* en la aplicación que deseamos descargar, el usuario debe ser avisado de que existen más de uno.

Durante la instalación, el usuario debe ser informado del progreso de ésta y se le debe de dar la oportunidad de cancelarla. La interrupción de la instalación debe dejar al dispositivo con el mismo estado que cuando se inició ésta. Estos son los pasos que el AMS sigue para la instalación de un *MIDlet*:

1. Si el JAD fue lo primero que descargó el AMS, el *MIDlet* debe tener exactamente la misma URL especificada en el descriptor.
2. Si el servidor responde a la petición del *MIDlet* con un código 401 (*Unauthorized*) o un 407 (*Proxy Authentication Required*), el dispositivo debe enviar al servidor las correspondientes credenciales.

3. El MIDlet y las cabeceras recibidas deben ser chequeadas para verificar que el MIDlet descargado puede ser instalado en el dispositivo. El usuario debe ser avisado de los siguientes problemas durante la instalación:
    1. Si no existe suficiente memoria para almacenar el *MIDlet*, el dispositivo debe retornar el Código de Estado (*Status Code*) 901.
    2. Si el JAR no está disponible en la URL del JAD, el dispositivo debe retornar el Código 907.
    3. Si el JAR recibido no coincide con el descrito por el JAD, el dispositivo debe retornar el Código 904.
    4. Si el archivo *manifest* o cualquier otro no puede ser extraído del JAR, o existe algún error al extraerlo, el dispositivo debe retornar el Código 907.
    5. Si los atributos "MIDlet-Name", "MIDlet-Version" y "MIDlet Vendor" del archivo JAD, no coinciden con los extraídos del archivo *manifes*t del JAR, el dispositivo debe retornar el Código 905.
    6. Si la aplicación falla en la autenticación, el dispositivo debe retornar el Código 909.
    7. Si falla por otro motivo distinto del anterior, debe retornar el Código 911.
    8. Si los servicios de conexión se pierden durante la instalación, se debe retornar el Código 903 si es posible.

La instalación se da por completa cuando el *MIDlet* está a disposición en el dispositivo, o no haya ocurrido un error irrecuperable.

6.5 Actualización de MIDlets

La actualización se realiza cuando se instala un *MIDlet* sobre un dispositivo que ya contenía una versión anterior de éste. El dispositivo debe ser capaz de informar al usuario cual es la versión de la aplicación que tiene instalada.

Cuando comienza la actualización, el dispositivo debe informar si la versión que va a instalar es más nueva, más vieja o la misma de la ya instalada y debe obtener verificación por parte del usuario antes de continuar con el proceso.

En cualquier caso, un *MIDlet* que no posea firma no debe de reemplazar de ninguna manera a otro que sí la tenga.

6.6 Ejecución de MIDlets

Cuando un usuario comienza a ejecutar un *MIDlet*, el dispositivo debe invocar a las clases CLDC y MIDP requeridas por la especificación MIDP. Si existen varios *MIDlets* presentes, la interfaz de usuario debe permitir al usuario seleccionar el *MIDlet* que desea ejecutar.

6.7 Eliminación de MIDlets

Los dispositivos deben permitir al usuario eliminar *MIDlets*. Antes de eliminar una aplicación el usuario debe dar su confirmación. El dispositivo debería avisar al usuario si ocurriese alguna circunstancia especial durante la eliminación del *MIDlet*. Por ejemplo, el *MIDlet* a borrar podría contener a otros *MIDlets*, y el usuario debería de ser alertado ya que todos ellos quedarían eliminados.

Capítulo 7:
MANUAL DE USUARIO DE LA APLICACIÓN SADD

7. MANUAL DE USUARIO DE LA APLICACIÓN SADD

7.1 Descripción del capítulo

Este capítulo está destinado al usuario final de la aplicación SADD; a lo largo de éste se conocerán las teclas empleadas, los iconos e indicadores utilizados y se verán a fondo las funciones de la aplicación, detallando los pasos a seguir para su correcta utilización.

Es necesario mencionar que el detalle del procedimiento de las funciones de la aplicación SADD visto en el presente manual está orientado al teléfono móvil: Motorola V600. Por lo tanto, algunos procedimientos de la aplicación SADD pueden variar de acuerdo a la marca y modelo del teléfono móvil que se está empleando.

Listado de funciones del menú SADD:

1. Configuración
   1.1. Prueba SMS
   1.2. Password
2. Leer Estados
3. Sistema
4. Func. Vehículo

7.1.1 Teclas a emplear

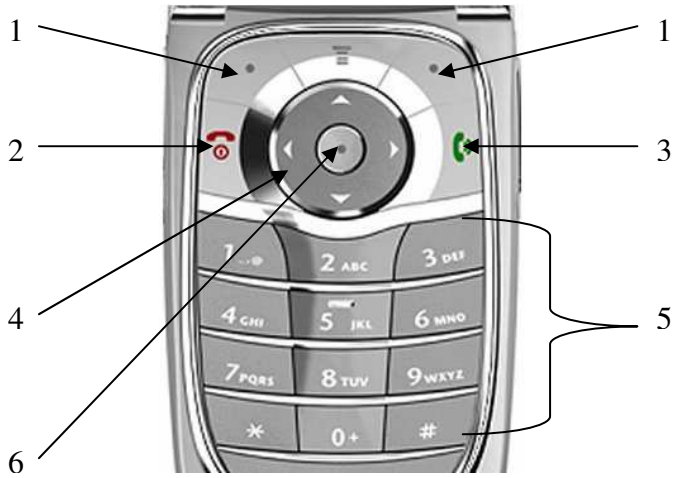


Figura 7.1 Teclas utilizadas en la aplicación SADD

1. Teclas de selección: estas teclas realizará la función que aparecerá sobre ellas en la pantalla
2. Fin de llamada: finaliza una llamada activa. Permite salir de cualquier función.
3. Inicio de llamada: permite responder a una llamada entrante, mientras la aplicación está activa, en pausa o destruida. Y permite realizar una llamada (digitar el número) mientras la aplicación está en pausa o destruida.
4. Teclas de desplazamiento: gracias a ellas se desplaza a través de los menús.
5. Teclas alfanuméricas: dentro de la aplicación SADD solamente son empleadas las teclas numéricas ya que solo se ingresan números telefónicos y el password numérico.
6. Seleccionador de menú (o seleccionadora): permite seleccionar la función del menú y acceder a los cuadros de texto.

7.1.2 Indicadores e íconos utilizados en la aplicación SADD

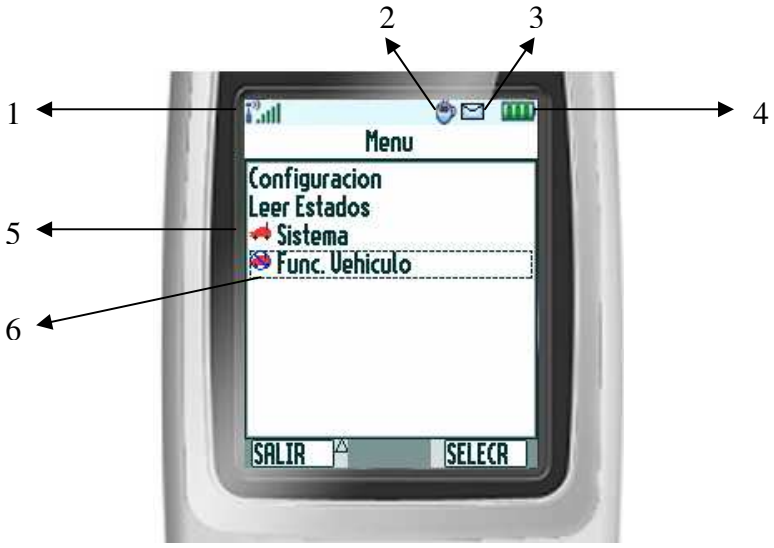


Figura 7.2 Iconos e indicadores utilizados en la aplicación SADD

1. Señal: Es recomendable que cada vez que se emplee la aplicación SADD, se observe si en el sitio a utilizar exista cobertura por parte de la operadora a la que corresponde el número de teléfono móvil.
2. Aplicación Java: este icono indica que una aplicación Java está activa, inclusive si ésta se encuentra en pausa
3. SMS recibido: indicador general que se activa cada vez que se reciba un mensaje corto
4. Carga de batería: Es recomendable que al emplear la aplicación y el teléfono, la batería no se encuentre baja para evitar que no se realicen las funciones seleccionadas.
5. Sistema Activo: este icono, , indica que el sistema ha sido activo y que se estará esperando las notificaciones SMS por parte del modem dentro del vehículo.
6. Funcionamiento bloqueado: este icono, , indica que el funcionamiento del vehículo ha sido bloqueado.

7.2 Funciones del menú SADD

Cada función del menú principal, figura 7.3, se marca por medio de las teclas de desplazamiento (arriba o abajo) y se escogen por medio de la tecla de selección que indica SELECR, o por medio de la tecla seleccionadora.

Figura 7.3 Pantalla del Menú SADD

7.2.1 Ejecutando la aplicación SADD por primera vez

Después de descargar la aplicación, éste da la opción de ejecutarlo. Al ejecutarse por primera vez la aplicación SADD este no mostrará el menú principal, por el contrario, desplegará una pantalla de ingreso del código de autenticidad de las funciones, conocido comúnmente como password, para brindar autenticidad del desarrollo de las funciones de la aplicación SADD, debido a que se sincronizará el password en la aplicación SADD con los dispositivos (módem y microcontrolador) alojados en el interior del vehículo, y al mismo tiempo brindar seguridad de un mal empleo del mismo por terceros. Además permite al teléfono móvil la creación de un registro en donde se almacenará en la memoria persistente del dispositivo el password, de manera que no se borre cada vez que se cierre la aplicación.

Figura 7.4 Pantalla de ingreso y confirmación del password

Dicha pantalla es como la que se muestra en la figura 7.4, ella consta de un cuadro de texto que se activa con presionar la tecla seleccionadora y así ingresar el password. El password es una variable numérica cuya longitud consta de 5 dígitos. Es necesario hacer énfasis que cada vez que se active un cuadro de texto, sea éste del password o de ingreso de número telefónico (como se verá más adelante), se mostrará un dato por defecto[1]: "0" ó "*". La cual simplemente debe borrarse con la tecla de selección que indique BORRAR y luego proceder a digitar el password deseado y presionar la tecla que indique OK cuando el ingreso finalice, es permitirá salir de cuadro de texto para luego después de oprimir la tecla que indique OK para continuar con la siguiente pantalla, confirmación del password, la cual se procede de igual forma, respetando que el mismo dato sea introducido en ambos cuadros de texto. Las restricciones del password se verán en el apartado de password.

Es necesario recalcar que esta pantalla solo aparecerá siempre y cuando el valor de password este vacío, es decir que solamente en la primera ejecución, ya que para presentarse el menú principal debe haber un password establecido. Por lo que la tecla que indica CANCEL esta demás, ya que al presionarla mostrará una solicitud de ingreso del password, tal como se muestra en la figura 7.5

[1] Dicho error de aparición de un dato por defecto solo se observa en los teléfono móviles Motorola, por lo que en otro teléfono se debe omitir el procedimiento de borrar el dato

Figura 7.5 Pantalla de solicitud de ingreso del password

7.2.2 Uso de mensajería

Antes de hablar de las funciones correspondientes a la aplicación SADD, es necesario mencionar un punto importante en el uso de mensajería, ya que éste posee una característica de seguridad dentro del WMA. Por lo que, si en un MIDlet se requiere el empleo de mensajería, sea esta SMS o MMS, siempre habrá una pantalla de confirmación de dicho uso.

Dentro de la aplicación SADD, existen ciertas informaciones que deben ser enviadas al modem, dentro del vehículo, vía SMS tales como el password y los códigos correspondientes a las acciones a realizar. Por lo que cada vez que se requiera enviar en la aplicación estas informaciones, se presentará una pantalla de confirmación tal como la que se muestra en la figura 7.6

Figura 7.6 Pantalla de confirmación de uso de mensajería

En dicha pantalla se brinda el número telefónico del receptor del mensaje corto, y la frecuencia de permiso de cada mensaje, para ello es recomendable indicar que se permita siempre, de lo contrario el mensaje no será enviado. Y es de tener cuidado en seleccionar: negar siempre, debido a que al hacerlo, la pantalla de confirmación ya no se mostrará más y ningun mensaje se enviará; teniendo que cerrar y abrir la aplicación para que vuelva a mostrarse la pantalla de confirmación del uso de mensajería.

7.2.3 Menú de configuración

Figura 7.7 Pantalla de Configuración de la aplicación SADD

La pantalla mostrada en la figura 7.7 es el menú de Configuración de la aplicación SADD, estas funciones son seleccionadas por las teclas de desplazamiento, y solo se puede ingresar a ellas por medio de la tecla de selección que especifique sobre ella SELECR. Dichas funciones son: Prueba SMS y Password. La primera permite especificar números de teléfonos móviles, que no sean la del modem, para realizar pruebas de envío de notificaciones desde la aplicación y la segunda permite cambiar el password.

7.2.3.1 Prueba SMS

Esta función ha sido desarrollada para observar el comportamiento de la aplicación en otros teléfonos móviles, modificando así el número del remitente y

del destinatario. Además permitir de realizar pruebas de envío de mensajes cortos a otros teléfonos móviles y no al modem.

Figura 7.8 Pantalla de la función prueba SMS

La figura 7.8 muestra la pantalla de la función prueba SMS, donde se encuentran dos cuadros de textos, el primero correspondiente al número del teléfono móvil donde estará la aplicación y el cual enviará los mensajes. Y el siguiente cuadro de texto en donde se encontrará el teléfono móvil de prueba que recibirá los mensajes. Para seleccionar uno de ellos, se utilizarán las teclas de desplazamiento (arriba o abajo), marcado ya una, se presionará la tecla seleccionadora y así se accede al cuadro permitiendo introducir el número telefónico.

Habiendo colocado correctamente ambos números se presiona la tecla que indique sobre ella OK, y desde ese momento todas las funciones en donde se envíe un mensaje corto utilizarán los números ingresados, pero por el contrario, si no se desea modificar los números se presiona la tecla que indica ATRÁS.

Como restricciones, debido a que esta función es para prueba posee por defecto los números que se utilizarán en la aplicación con el vehículo, pero podrán ser modificados en forma temporal al ingresar a esta función; es en forma temporal ya que al cerrar la aplicación SADD los números ingresados se borrarán, ya que no son almacenados en la memoria persistente del dispositivo.

Si se desea utilizar los números que funcionan en la aplicación junto con la del modem, se deben ingresar de la misma forma o se cierra y abre la aplicación SADD.

Otro aspecto a considerar, es la longitud del número telefónico, porque es de 8 dígitos. Por lo que la función no aceptará cuadros de texto vacíos o que posean números con longitudes menores a 8 dígitos; de ser así, se mostrará en pantalla una solicitud de ingreso correcto de los números.

7.2.3.2 Password

Esta función permite modificar la contraseña de autenticación de las funciones dentro de la aplicación SADD. Por lo cual, modifica el dato que está almacenado en la memoria persistente del teléfono móvil. Esta función emplea las mismas pantallas que se observan cuando se corre por primera vez la aplicación (Figura 7.4), con la variante que antes de ellas, se muestra una pantalla donde se debe ingresar el password vigente. Tal como se observa en la figura 7.9.

Figura 7.9 Pantalla de ingreso del password

Los pasos a seguir para modificar un password son: introducir la contraseña vigente, escribir la nueva contraseña y confirmar dicha contraseña. Para cada una existe su respectiva pantalla. Y los procedimientos a seguir son:

- Presionar la tecla seleccionadora para acceder al cuadro de texto
- Introducir la contraseña numérica de 5 dígitos de longitud

- Ingresada la contraseña se presiona la tecla que indique sobre ella OK, lo que permitirá salir del cuadro de texto.
- Finalmente se presiona la tecla que indique sobre ella OK, para pasar a la siguiente pantalla. Y se repiten los pasos anteriores en las siguientes dos pantallas, la de ingreso del nuevo password y la de su confirmación.
- Si se está en la pantalla de confirmación del nuevo password, al presionar la tecla OK, se mostrará la pantalla de uso de mensajería (ver apartado 7.2.2) y posteriormente se confirmará el cambio de contraseña.

Figura 7.10 Pantalla de verificación del password

Dentro de las restricciones en el password se tienen:

- El password es un dato numérico
- Se debe respetar la longitud del password de 5 dígitos, de lo contrario cada vez que se quiera aceptar el password habrá una pantalla de información de longitud errónea.
- El nuevo password junto a su confirmación deben ser iguales de lo contrario no se aceptará el nuevo password
- Y el más importante de todas las restricciones, la contraseña no debe ser olvidada debido a que la mayoría de las funciones solicitan verificación del password para validar la acción, figura 7.10, y sin ella ninguna acción se realizará. En caso de olvido, la aplicación SADD solo permite 5 intentos simultáneos de ingreso de contraseña en caso de introducirlo erróneamente. Después de los 5 intentos aparecerá una pantalla de

bloqueo de la aplicación SADD, figura7.11, y no podrá ser utilizado nuevamente a menos que la aplicación sea eliminada y nuevamente cargada al teléfono móvil por parte de sus creadores, además la rutina desarrollada por el microcontrolador, ubicado en el interior del vehículo, debe ser reseteado. Para así, sincronizar el password en ambos dispositivos nuevamente.

Figura 7.11 Pantalla de bloqueo de la aplicación SADD

7.2.4 Leer Estados

Esta función permite al usuario solicitar al modem, para que éste le envíe al teléfono móvil el estado de los dispositivos en el interior del vehículo. Esta solicitud es en forma forzada, ya que si el sistema estuviera activo se recibirían los estados en forma automática cada vez que conmutarán los estados de los dispositivos.

Para acceder a esta función es necesario seleccionarla en el menú, marcándola con las teclas de desplazamiento y accediendo a ella por medio de la tecla de selección que indique sobre ella SELECR o la tecla seleccionadora. Posteriormente, se verificará el password y al introducirlo la contraseña correctamente, se tendrá en pantalla una nota informativa tal como la figura 7.12 para que en pocos segundos se tenga un indicador de mensaje recibido, un mensaje de texto que contendrá la información concerniente al estado de los dispositivos.

Figura 7.12 Solicitud de estados de dispositivos confirmada

7.2.4 Sistema

Esta función activa y desactiva el sistema dentro del vehículo. Para seleccionarla se debe marcar Sistema en el menú por medio de las teclas de desplazamiento y acceder a ella usando la tecla que indique sobre ella SELECR o la tecla seleccionadora. Mostrándose la pantalla que se observa en la figura 7.13.

Figura 7.13 Pantalla de la función Sistema

Para seleccionar Activar o Desactivar el sistema se debe marcar y presionar la tecla seleccionadora, y luego presionar la tecla que indica sobre ella SELECR, se escribe correctamente la contraseña en la pantalla de Verificación de Password y finalmente se notificará que la acción se ha realizado, figura 7.14.

Figura 7.14 Acciones realizadas en la función Sistema

Si se activa, se solicita al microcontrolador que esté revisando continuamente el estado de los dispositivos, y cada vez que exista una variación en dichos estados, éste deberá enviar una notificación de cambio de estados al teléfono móvil por medio de SMS. En cambio, si se desactiva se solicita al microcontrolador que interrumpa la lectura de los estados de los dispositivos.

Es de prestar atención que cada vez que se active el sistema, aparecerá junto a la palabra Sistema un dibujo de un vehículo rojo que indica que el sistema está activo, figura 7.15, pero si se desactiva dicho indicador desaparecerá.

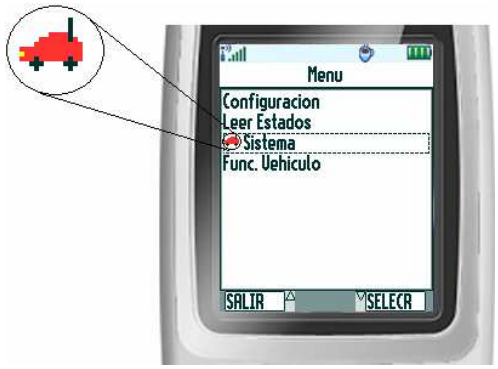


Figura 7.15 Sistema activó

7.2.4 Funcionamiento del Vehículo

Esta función solicita al microcontrolador, bloquear y desbloquear el funcionamiento del vehículo, debido a la obstrucción y permiso del pasó de corriente en los inyectores del vehículo. Para ingresar a esta función, se debe

cerciorar antes que el icono de sistema activado esté presente, de lo contrario no se puede utilizar, pero si está presente se selecciona la función por medio de las tecla que indica sobre ella SELECR o por medio de la tecla seleccionadora.

Figura 7.16 Pantalla de la función: Funcionamiento del Vehículo

Para seleccionar Bloquear o Desbloquear el sistema se debe marcar y presionar la tecla seleccionadora, y luego presionar la tecla que indica sobre ella SELECR, se escribe correctamente la contraseña en la pantalla de Verificación de Password y finalmente se notificará que la acción se ha realizado, figura 7.17

Figura 7.17 Acciones realizadas en la función Func. Vehículos

Cada vez que se realice el bloqueo del funcionamiento del vehículo aparecerá junto a la palabra Func. Vehículo un indicador, tal como se muestra en la figura 7.18. Y cuando se desbloquee el funcionamiento del vehículo, desaparecerá el indicador.

Figura 7.18 Funcionamiento de vehículo bloqueado

Es importante recalcar que para poder salir de la aplicación SADD, es necesario que no haya algún indicador presente en el menú, es decir que el sistema no esté activo y mucho menos que el funcionamiento del vehículo esté bloqueado.

Capítulo 8:
JUSTIFICACIÓN DE LA APLICACIÓN SADD

8. JUSTIFICACIÓN DE LA APLICACIÓN SADD

8.1 Descripción del capítulo

La situación social en estos días a llevado a las personas a encontrar nuevas formas de comunicarse, y a la vez, tener cierto modo de vida en las que ellas puedan sentirse seguras, es por ello, la tecnología se encuentra en constante, o abrupto, movimiento para poder suplir dichas necesidades. Uno de los principales problemas que atraviesa la sociedad hoy en día es el robo de vehículos, no solo en nuestra sociedad sino en muchas mas alrededor del mundo, es un problema que va creciendo diariamente y por ello el desarrollo de nuevas tecnologías se hace indispensable para poder respaldar la seguridad de personas y de dichos bienes personales. El presente capítulo justifica un sistema de control de dispositivos analógicos utilizando tecnología celular GSM, la cual podría tener fines de seguridad, si así se dispone en el futuro. Es por eso que a continuación se presenta una reseña histórica y estadística del robo de automotores en El Salvador en el pasado año 2004.

8.2 Tecnología GSM aplicada a seguridad de automóviles

Uno de los principales problemas que afronta la sociedad hoy en día es el robo de automóviles, es por eso que en algunas sociedades se ha comenzado a implementar el rastreo de los vehículos para poder garantizar a los usuarios su seguridad y el de sus bienes, tecnologías de control a distancia utilizando la tecnología celular GSM se ha comenzado a expandir para poder aplicarse a otras áreas en las que los sistemas de comunicación pueden ser una respuesta a las necesidades de la sociedad. El robo de vehículos en nuestro país desgraciadamente ha estado incrementando con los meses, es por eso que una aplicación como el control de

dispositivos a distancia puede ayudar a reducir dicha cantidad de robos que se dan en nuestro país. En los apartados siguientes se muestra un control estadísticas de robos de automóviles, elaborado por empresas aseguradoras que llevan en sus bases de datos información que ayuda a tener una perspectiva de la situación.

8.3 Estadística de robos de automotores asegurados (año 2003–2004)

La Asociación Salvadoreña de Empresas de Seguros a través de su *Sistema Informático Especializado* ha recopilado información sobre el comportamiento estadístico de robos y hurto de automotores asegurados, correspondiente al período comprendido del primero de enero al treinta y uno de diciembre de 2004.

Los datos obtenidos, poco varían con respecto al comportamiento observado durante el año 2003. Al finalizar el 2003 se totalizaban 717 casos de robos; y al finalizar el año 2004 se totalizaban 689 automóviles robados. Lo cual indica que la variación del año 2003 al 2004 es de una disminución en el 4.1%. Dicha disminución del cuatro por ciento no refleja directamente si ha sido menor la cantidad de indemnizaciones pagadas por el sector asegurador; debido a que las cantidades pagadas a los clientes pueden variar de automóvil en automóvil y no son directamente proporcionales al número de casos de robo y hurto.

Cabe mencionar que a Junio de 2004 se habían reportado 343 casos de robo y hurto. Para el segundo semestre se reportaron 346 casos. Lo cual indica un balance que refleja que el robo de automóviles no esta ligado directamente a épocas o estaciones del año.

El tipo de vehículo preferido por las bandas roba carros sigue siendo el Pick up. Al finalizar el año 2003 se habían registrado 240 casos de robos de este tipo de automotor (Incluyendo los Pick up 4x4), liderando la lista global. Al finalizar el año 2004 se reportaron 201 casos de robo de Pickups (Incluyendo los del tipo 4x4). Siguen

estando los automotores de tipo Sedan y las Motocicletas entre los objetivos favoritos de las bandas roba carros.

Tipo de vehículos mas robados al finalizar el **2003**

Marca	Cantidad	Porcentaje
PICK UP	213	29.71%
MOTOCICLETA	119	16.60%
SEDAN	157	21.90%
MICROBUS	62	8.65%
RUSTICO	70	9.76%
PANEL	27	3.77%
PICK UP 4X4	27	3.77%
CAMIONETA	13	1.81%
CAMION	13	1.81%
CABEZAL	4	0.56%
FURGON	1	0.14%
OTROS	9	1.26%
MONTACARGA	1	0.14%
CISTERNA	1	0.14%
* TOTAL	717	100.00%

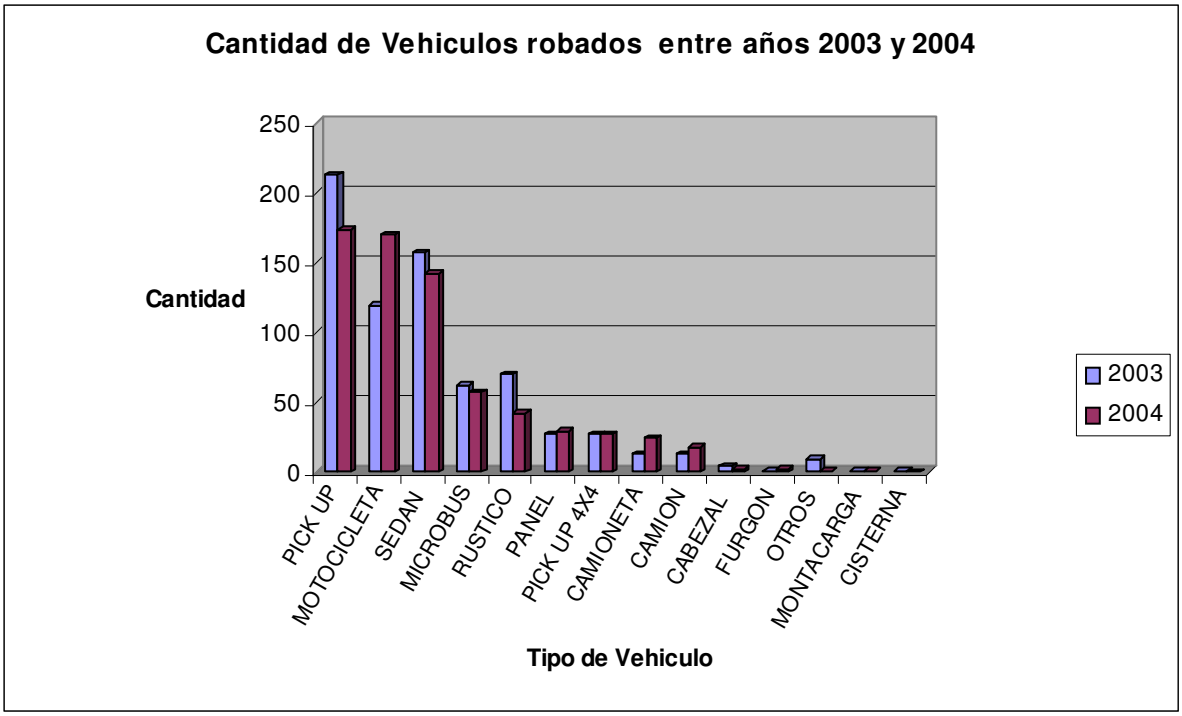
Tipo de vehículos mas robados al finalizar el **2004**

Marca	Cantidad	Porcentaje
PICK UP	174	25.33%
MOTOCICLETA	170	24.75%
SEDAN	142	20.67%
MICROBUS	57	8.30%
RUSTICO	42	6.11%
PANEL	29	4.22%
PICK UP 4X4	27	3.93%
CAMIONETA	24	3.49%
CAMION	18	2.62%
CABEZAL	2	0.29%
FURGON	2	0.29%
AUTOBUS	1	0.15%
MONTACARGA	1	0.15%
* TOTAL	689	100.00%

** Tabla del 2004 ordenada de mayor incidencia a menor incidencia. Tabla del 2003 ordenada a manera de comparar con el año 2003*

Fuente: Compañías de seguros salvadoreñas

A continuación se muestra la representación gráfica de las tablas anteriores referentes al robo de vehículos entre los años 2004 y 2004, clasificados por modelos de automóviles.



Fuente: Compañías de seguros salvadoreñas

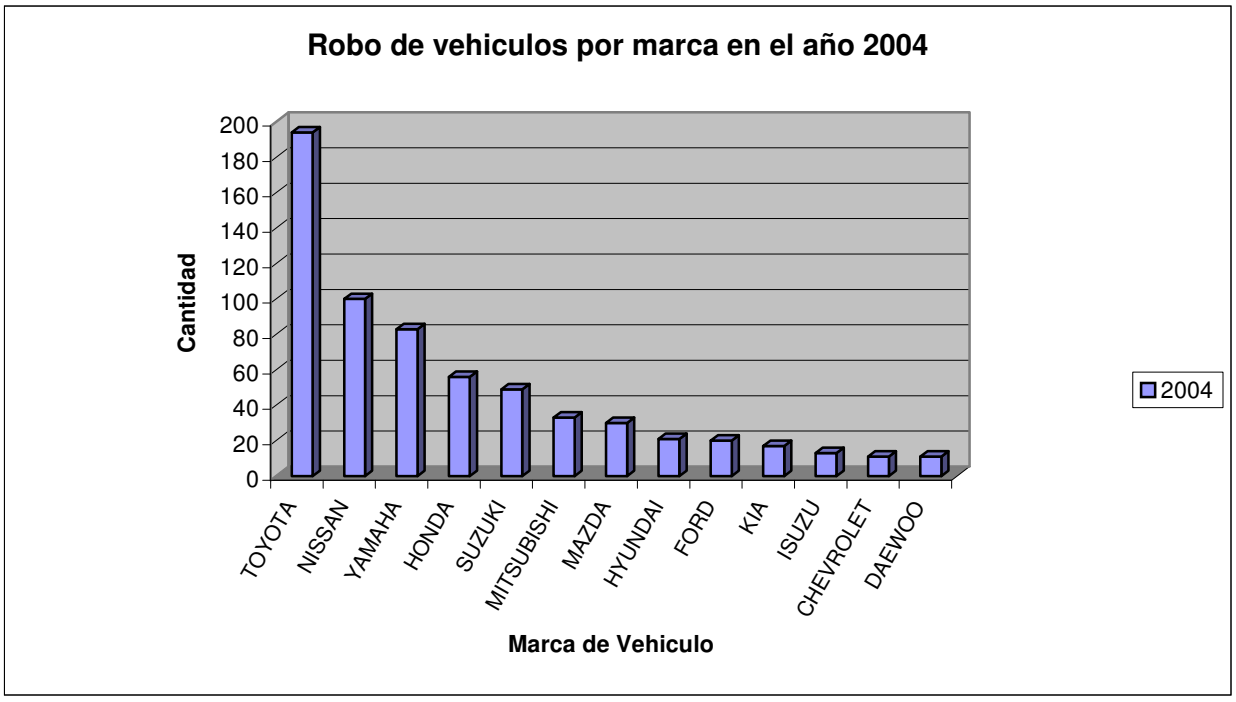
Figura 8.1 Vehículos robados (2003-2004)

En lo que respecta al robo de automotores por Marca; Toyota repite el patrón presentado a Diciembre de 2003. Y sigue estando en el primer lugar de la lista de robos. Al finalizar el 2003 se reportaron 315 casos de robos de automotores de dicha marca, a Junio de 2004 se tenían 106 casos de Robo y Hurto de carros Toyota, aumentando a 194 casos en Diciembre de 2004.

Marca	Cantidad
TOYOTA	194
NISSAN	100
YAMAHA	83
HONDA	56
SUZUKI	49
MITSUBISHI	33
MAZDA	30
HYUNDAI	21
FORD	20
KIA	17
ISUZU	13

CHEVROLET	11
DAEWOO	11

En todo el país a Noviembre de 2004 la PNC reporta 1099 casos de robo y hurto de automotores marca Toyota. (Incluyendo vehículos no asegurados) A continuación se presenta la gráfica del robo de vehículos clasificado por marca de vehículo concerniente al año 2004.



Fuente: Compañías de seguros salvadoreñas

Figura 8.2 Vehículos robados por marca

Siendo también a nivel nacional Toyota, la marca con mayor número de casos reportados. El alto índice en automotores de la marca Toyota radica en que es el automóvil más común y de mayor circulación en el país.

En lo que respecta a Modelos de automotores al finalizar 2004, los modelos mas robados son el Pickup Toyota Hilux, el Microbús Toyota Hiace y la Motocicleta Yamaha Turismo. Como se puede observar en la siguiente tabla, los mencionados modelos también fueron los favoritos del año 2003.

AÑO 2003

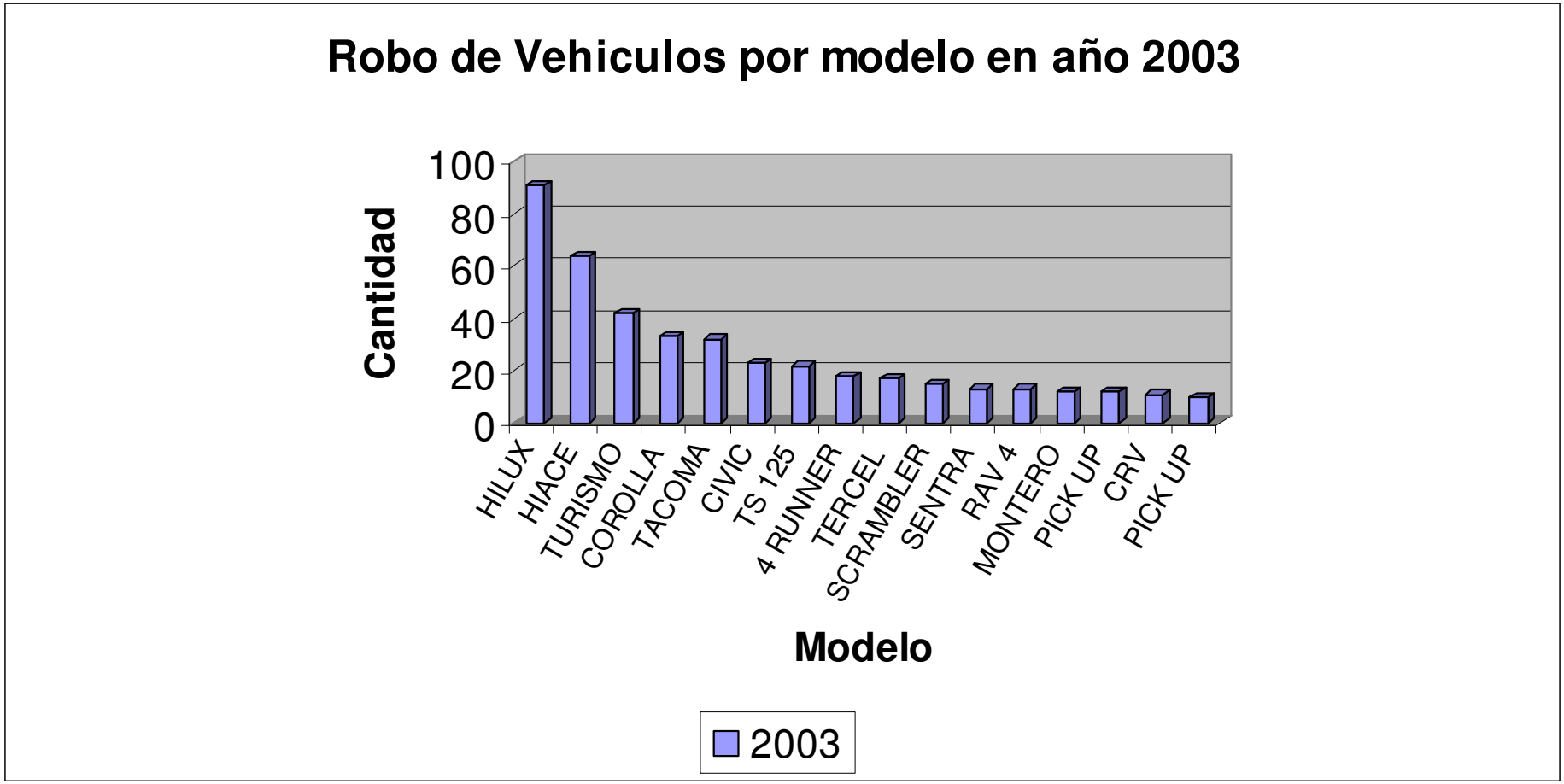
Marca	Modelo	Cantidad
TOYOTA	HILUX	91
TOYOTA	HIACE	64
YAMAHA	TURISMO	42
TOYOTA	COROLLA	33
TOYOTA	TACOMA	32
HONDA	CIVIC	23
SUZUKI	TS 125	22
TOYOTA	4 RUNNER	18
TOYOTA	TERCEL	17
YAMAHA	SCRAMBLER	15
NISSAN	SENTRA	13
TOYOTA	RAV 4	13
MITSUBISHI	MONTERO	12
TOYOTA	PICK UP	12
HONDA	CRV	11
NISSAN	PICK UP	10

AÑO 2004

Marca	Modelo	Cantidad
TOYOTA	HILUX	65
TOYOTA	HIACE	49
YAMAHA	TURISMO	47
NISSAN	FRONTIER	28
NISSAN	SENTRA	21
SUZUKI	TS 185	20
HONDA	CIVIC	18
HONDA	XL 125	18
TOYOTA	COROLLA	14
TOYOTA	4 RUNNER	13
FORD	EXPLORER	11
SUZUKI	TF 125	11
TOYOTA	TACOMA	11
YAMAHA	SCRAMBLER	11
MAZDA	B2900	10
MITSUBISHI	MONTERO	10
MAZDA	B2500	8
MITSUBISHI	L 200	8
NISSAN	PATHFINDER	8

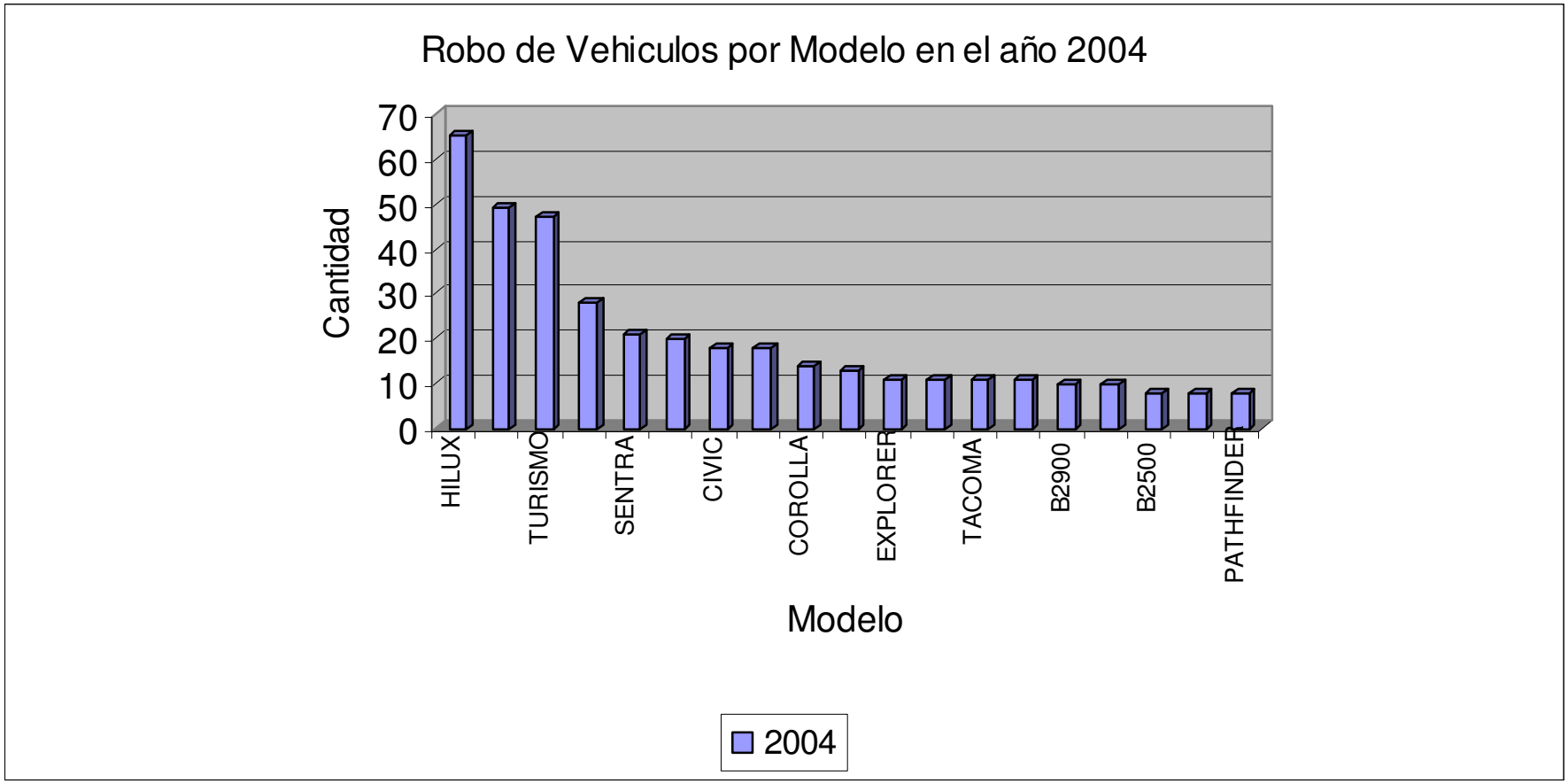
Fuente: Compañías de seguros salvadoreñas

En lo que respecta al año de fabricación del carro la PNC declaró que los automotores favoritos son los que radican entre los años 1995 y 2004 debido a que estos los pueden vender afuera de las fronteras una vez les han modificado el chasis, y los modelos de carros viejos son robados con la intención de desmantelarlos venderlos como piezas en talleres. Añadió la PNC, que la mayoría de automotores son robados entre lunes y viernes pero la mayor incidencia de robos se da entre los días lunes y miércoles. Además ha declarado que a Noviembre de 2004 se habían desmantelado 12 bandas roba carros, haciendo un total de 149 capturas. Por lo que a nivel nacional se habían recuperado 1300 vehículos robados de los cuales 100 pertenecen al sector de vehículos asegurados. A continuación se presenta la grafica comparativa entre los años 2003 y 2004 respecto al robo de vehículos clasificados modelo entre dichos años.



Fuente: Compañías de seguros salvadoreñas

Figura 8.3 Vehículos robados por modelo (2003)



Fuente: Compañías de seguros salvadoreñas

Figura 8.4 Vehículos robados por modelo (2004)

Según el estudio realizado por once compañías aseguradoras en el país el robo de autos en el año 2004, deja en primer lugar a los pick up, como el tipo de automóvil mas robado, le siguen las motocicletas, después el tipo sedan y posteriormente los microbuses.

Las compañías involucradas en el estudio son:

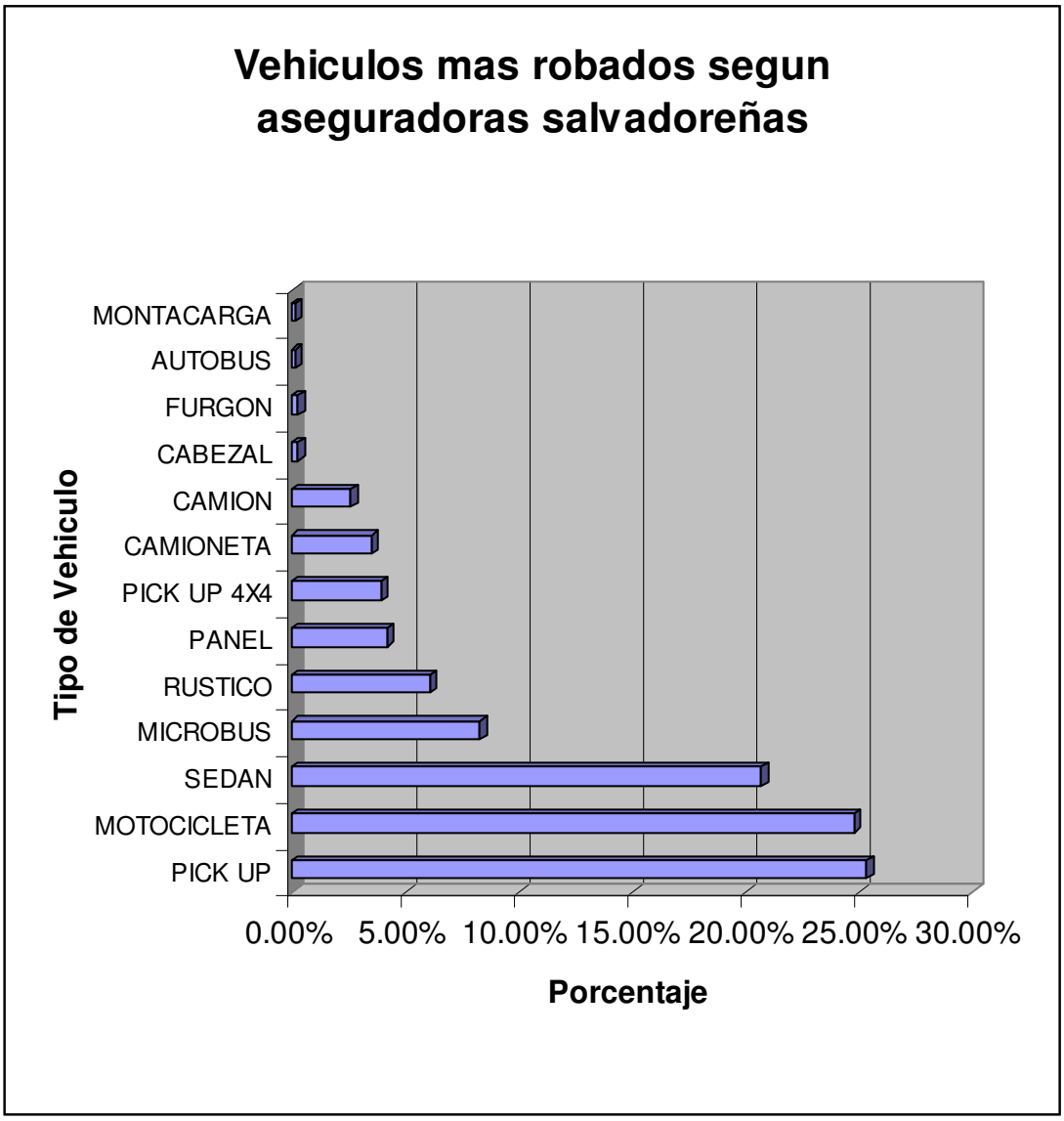
- ACSA
- AIG Unión y Desarrollo
- Asegurado Popular
- ASESUISA
- Compañía Anglosalvadoreña
- Compañía General de Seguros
- Internacional de Seguros
- La Central de Seguros
- La Centroamericana
- Pacífico
- SISA

Dichas compañías reportaron la siguiente estadística correspondiente al año 2004, referente al robo de vehículos, ordenados desde el tipo de vehículo mas robado al menos robado.

Aseguradoras

Marca	Cantidad	Porcentaje
PICK UP	174	25.33%
MOTOCICLETA	170	24.75%
SEDAN	142	20.67%
MICROBUS	57	8.30%
RUSTICO	42	6.11%
PANEL	29	4.22%
PICK UP 4X4	27	3.93%
CAMIONETA	24	3.49%
CAMION	18	2.62%
CABEZAL	2	0.29%

FURGON	2	0.29%
AUTOBUS	1	0.15%
MONTACARGA	1	0.15%
* TOTAL	689	100.29%



Fuente: Compañías de seguros salvadoreñas

Figura 8.5 Tipo de vehículo robados

El problema del robo de vehículos se ha venido dando continuamente en la región Centroamérica es por eso que las compañías aseguradoras realizan este tipo de estudios, para poder tener una perspectiva más amplia de la gravedad de la situación, y para que empresas encargadas al desarrollo de tecnológicas puedan tener bases estadísticas para el desarrollo de estas, que puedan al menos reducir la cantidad de robos de vehículos. El proyecto que se presenta en este documento tiene como base este tipo de tecnología que podría ayudar a minimizar este tipo de problema que se da día a día en la región y con ello conseguir el desarrollo de mejores y nuevas aplicaciones basadas en la red de telefonía celular GSM.

CONCLUSIONES

- Al desarrollar una aplicación empleando el WMA, para que el teléfono móvil se pueda comunicar hacia otro dispositivo vía SMS es necesario programar el dispositivo en modo servidor en donde el servidor debe ser el número de teléfono del usuario
- Para lograr descargar un midlet al teléfono móvil por cualquier otro método que no sea por OTA (over the air), es necesario percatarse que la función Java Apps Loader del teléfono móvil esté activado de lo contrario será imposible descargar la aplicación.
- Para lograr la comunicación entre el Modem GSM y el circuito controlador es necesario que ambos dispositivos trabajen bajo el estándar RS-232 y con los mismos parámetros de comunicación, además utilizando el control de flujo por medio de Hardware, si no se cumplen estas condiciones nunca se logrará tener comunicación entre los dos dispositivos.
- Debido a que el Modem GSM no es un dispositivo programable sino un dispositivo que responde o ejecuta acciones en función de los comandos que recibe, por lo tanto es necesario el uso de un sistema que provea los comandos necesarios y reconozca las respuestas y notificaciones que el Modem genera para lograr la activación/desactivación y lectura del estado de los dispositivos.
- En la búsqueda de la optimización del Servicio de Mensajes Cortos, se estableció un tipo de sintaxis para poder enviar la mayor cantidad de información útil posible, en un solo mensaje SMS.
- No se debe esperar el mismo comportamiento de las aplicaciones Java en todos los teléfonos móviles, ya que éste posee ciertas variaciones de acuerdo a las marcas de teléfonos, debido a las diferencias en los sistemas operativos.
- Para un mejor desempeño de los circuitos de comunicación serial es necesario el tener la recepción y la transmisión de caracteres independientes, es decir que existan circuitos independientes que controlen de forma eficaz cada una de las tareas dejando siempre un medio de comunicación entre las dos etapas esto

para lograr que no se congestionen los buffer de recepción cuando los circuitos están ocupados realizando alguna otra tarea.

- Para lograr minimizar al máximo la posibilidad de recepción de caracteres basura en el USART de los Microcontroladores y así tener un funcionamiento apropiado es necesario deshabilitar el envío de caracteres de Echo del Modem hacia el DTE, es decir que se debe evitar el reenvío de datos como normalmente se hace en la comunicación entre un DCE y un DTE.

RECOMENDACIONES

- En el desarrollo del proyecto se utilizó un inversor de voltaje (12Vdc a 120Vac), se empleó como una forma de alimentar las fuentes que se necesitaban para la energización de los dispositivos, más por el hecho de necesitar una fuente de -12V para la alimentación del driver para el RS-232 y al no existir en el vehículo tal potencial, se decidió utilizar este dispositivo por ser una solución accesible y rápida, sin embargo, es de reconocer que existen muchas formas diferentes en las que se podría lograr esta misma aplicación. También es importante mencionar que el dejar conectado directamente la batería del automóvil a este inversor de voltaje hará que tarde o temprano la carga de la batería se agote, por lo tanto, se sugiere utilizar una segunda batería de 12Vdc con la suficiente corriente para alimentar el inversor, y a la vez, disponer de un circuito que realice la carga de dicha batería, ya sea por medio de un segundo alternado, aunque, esto hará elevar los costos de la implementación.
- Es recomendable utilizar un modelo de MODEM GSM cuya característica de voltaje de alimentación sea de 12Vdc, y de esta forma simplificar un poco la instalación y la circuitería asociada a la alimentación de este dispositivo.
- Una desventaja asociada a la aplicación en general, es que existen formas en las cuales el programa desarrollado para el móvil puede perder sincronía con el circuito microcontrolador, es decir que los estados del sistema presentes en el circuito microcontrolador sean diferentes a los reflejados en la aplicación SADD dentro del teléfono móvil, esto puede ser provocado por problemas de transmisión y recepción de los SMS tanto en el modem como en el teléfono móvil. Por lo tanto, se sugiere adicionar a la aplicación en el teléfono móvil, una función de sincronización, en donde, se solicite al modem que le envíe el estado del sistema actual presente en el microcontrolador, y al recibir el teléfono móvil el SMS solicitado, permita a la aplicación sincronizarse en base al contenido de dicho mensaje.

BIBLIOGRAFÍA

[1] **"GSM World"**

Fecha de visita: 13 de junio de 2005

http://www.gsmworld.com/about/history/index.shtml

[2] **"GSM System Survey"**

Ericcson. StudentText. EN/LZT 123 3321 R3B. 1998

Páginas: 10-17

[3] **"Java a Tope: J2ME (Java 2 Micro Edition)".** Edición Electrónica

Autores: Sergio Gálvez Rojas y Lucas Ortega Díaz

Dpto. de Lenguajes y Ciencia de la Computación. E.T.S de Ingeniería Informática

Fecha de visita: 2 de febrero de 2005

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

[4] **"Developing Wireless Applications Using MIDP 2.0, WMA and MMA"**

Bill Day. Sun Microsystem, Inc

Fecha de visita: 10 de junio de 2005

http://www.javaoneonline.mentorware.net/mw/subsystems/2000/system/docs/wireless_applications.pdf

[5] **"Estudio Monográfico sobre GSM y su Evolución hacia 3G"**

Autores: Carlos Caballero y Otto Varela

Universidad Don Bosco. Trabajo de graduación 2001

Páginas: 146-149

[6] "**Next Generation of Mobile Terminals**"

Autor: Jorunn Kaasin
Norwegian University of Science and Technology
Fecha de visita: 10 de junio de 2005
Páginas: 15-17
http://www.item.ntnu.no/lab/nettint1/activities/hovedoppgaver/2002/JorunnKaasin/diplom_siste.pdf

[7] "**Wireless Messaging API (WMA)"** para Java 2 Micro Edition
JSR 120 Expert Group
Siemens AG
Fecha de visita: 3 de Julio de 2005
http://www.jcp.org/aboutJava/communityprocess/final/jsr120/

[8] **"V300, V400, V500 and V600 Handset Series"**
Technical Manual. General Developer Edition. Motorola
Fecha de visita: 17 de marzo de 2005
Páginas: 82-90
http://www.motocoder.com

[9] **Sitio oficial de Sun**
http://java.sun.com

[10] **Sitio oficial de Motorola**
http://motorola.com

GLOSARIO

A

AMS	Application Management Software
API	Application Program Interface
Applet	Programa Java que se ejecuta en un navegador web
AUC	Authentication Center
AWT	Abstract Window Toolkit

B

BSC	Base Station Controller
BTS	BaseTransceiver Controller
BlueTooth	Dispositivo portable para mecanismos de radio móvil a corta distancia

C

CEPT	Conference European Post and Telecommunications
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration

D

DCE	Data Communication Equipment
DCS	Digital Cellular System
DECT	Digital Enhanced Cordless Telecommunication
DTE	Data Terminal Equipment
DTI	Digital Technology International

E

EEC	End-Entity Certificate
EIR	Equipment Identity Register
ETSI	European Technical Standard Institu

F

FCC	Federal Communication Commission
FIFO	First In, First Out

G

GUI	Graphic User Interface
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communication

H

Handover	es el cambio de una llamada continua a un canal o celda diferente
HLR	Home Location Register
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol

I

IDE	Integrated Development Environment
ITA	Interim Type Approval
ITU	International Telecommunication Union
IP	Internet Protocol
ISDN	Integrated Services Digital Network

J

J2EE	Java 2 Platform, Enterprise Edition
J2ME	Java 2 Platform, Micro Edition
J2ME	Java 2 Platform, Standard Edition
JAD	Java Application Descriptor
JAL	Java Application Loader
JAR	Java Archive. Usado por aplicaciones J2ME para compresión y empaquetamiento.
JDBC	Java Data Base Connectivity
JNI	Java Native Interface
JSR 120	Java Specifications Request 120. Definido como una colección de APIs opcionales que proveen acceso estándar a recursos de comunicación inalámbrica.
JVM	Java Virtual Machina

K

Kbps	Kilo bits por segundo
KVM	Kilo Virtual Machine

M

ME	Mobile Equipment
MIDlet	es una aplicación Java realizada con el perfil MIDP sobre la configuración CLDC
MIDP	Mobile Information Device Profile
MMA	Multimedia API
MMS	Multimedia Messaging Service
MoU	Memorandum of Understanding
MS	Mobile Station
MSC	Mobile service Switching Center
MT	Mobile Termination

N

NMC	Network Managment Center
NMT	Nordic Mobile Telephone

O

OMC	Operation and Maintenance Center
OTA	Over The Air

P

PCN	Personal Communication System
PCS	Personal Cellular System
PDA	Personal Digital Assistant

R

RAM	Random Access Memory
RMI	Remote Method Invocation
RMS	Record Management System
ROM	Read Only Memory
Roaming	Capacidad que poseen los teléfonos móviles de registrarse a una red distinta de la que proviene

	RTOS	Real Time Operating System
S		
	SADD	Sistema de Activación y Desactivación de Disp.
	SACCH	Show Associated Control Channel
	SDCCH	Stand-Alone Dedicated Control Channel
	SDK	Software Development Kit
	SIM	Subscriber Identity Module
	SMS	Short Message Service
	SMSC	Short Message Service Center
	S.O	Sistema Operativo
	SU	Subscriber Unit
T		
	TACS	Telephone Advanced Cellular System
	TCH	Traffic Channel
	TDMA	Time Division Multiple Access
	TTL	Transistor-Transistor Logic
U		
	UI	User Interface
	UMTS	Universal Mobile Telephone System
	URL	Uniform Resource Locator
	USART	Universal Synchronous Asynchronous Receiver Transmitter
V		
	VLR	Visitor Location Register
	VM	Virtual Machine
W		
	WAP	Wireless Application Protocol
	WMA	Wireless Messaging API
	WML	Wireless Markup Language

ANEXO

Anexo A: Adapador GSM SMS [7]

A.1.0 Estructura de Mensaje SMS GSM

Los mensajes SMS GSM están definidos en el estándar GSM 03.40. El mensaje consiste en un encabezado y un campo llamado TP-Usuario-Datos. El campo TP-Usuario-Dato lleva la carga útil del mensaje corto y una información de encabezado adicional que no es parte del encabezado. Esta información adicional esta contenida en un campo llamado Usuario-Datos-Encabezado. La presencia de la información adicional en este campo esta indicado por un campo separa que es parte del encabezado.

A.1.1 Longitud de Mensaje

La longitud máxima de un protocolo de mensaje depende de la codificación y si hay encabezado opcionales presentes en el campo TP-Usuario-Datos. Si la información de encabezado opcional especifica un número de puerto entonces la carga útil que encaja en el protocolo de mensaje SMS será más pequeña.

Típicamente, el mensaje es mostrado al usuario final. Sin embargo, esta aplicación JAVA soporta el uso de números de puerto para especificar una aplicación JAVA como el objetivo del mensaje.

El mensaje que la aplicación JAVA envía puede ser demasiado extenso para encajar en un solo protocolo de mensaje SMS. En este caso, en la implementación se debe utilizar la característica de concatenación especificada en las secciones 9.2.3.24.1 y 9.2.3.24.8 del estándar GSM 03.40.

Esta característica puede ser usada para dividir la carga útil del mensaje dado a la aplicación JAVA en múltiples protocolos de mensaje SMS y pasar la carga útil totalmente reensamblada a la aplicación vía API.

A.1.2 Concatenación de la Carga Útil del Mensaje

El estándar GSM 03.40 especifica dos mecanismos para la concatenación, especificados en las secciones 9.2.3.24.8 y 9.2.3.24.1. Estos difieren en la longitud del número de referencia. Para mensaje que son enviados, la implementación puede utilizar ambos mecanismos. Para mensajes recibidos, las implementaciones deben aceptar mensajes con ambos mecanismos.

Nota: Dependiendo de que mecanismo sea usado para mandar mensajes, la longitud máxima de carga útil de un solo mensaje de protocolo SMS difiere por un carácter por byte.

Para que la concatenación resulte, sin importar que mecanismo sea usado por la implementación, las aplicaciones son recomendadas a asumir una longitud de número de referencia de 16 bits, cuado se estime cuantos mensajes de protocolo SMS son necesarios para enviar un mensaje dado. Las longitudes en la tabla A-1 están calculadas asumiendo una longitud de número de referencia de 16 bits.

Las implementaciones en este API, deben soportar al menos tres mensajes de protocolo SMS para ser recibidos y concatenados juntos. Similarmente, para enviar, los mensajes que pueden ser enviados con más de tres mensajes de protocolo SMS deben ser soportados. Dependiendo de la implementación, estos límites pueden ser mayores. Sin embargo, las aplicaciones son notificadas para enviar los mensajes que lleven más de tres mensajes de protocolo, a menos que tengan una razón para asumir que el destinatario será capaz de manejar un número mayor.

El método de número de segmentos (MessageConnection.numberOfSegmentes) permite a la aplicación verificar cuantos mensajes de protocolo SMS usará un mensaje dado cuando es enviado.

Tabla A-1: Número de mensajes de protocolo SMS necesitados para diferentes longitudes de carga útil.

Codificación de Encabezados Opcionales	*Sin número de puerto (mensaje a ser mostrado a usuario final)*		*Con número de puerto (mensaje objetivo de una aplicación)*	
	Longitud	Mensajes SMS	Longitud	Mensajes SMS
GSM 7-bits Alfabeto	0-160 caract.	1	0-152 caract.	1
	161-304 caract.	2	153-290 caract.	2
	305-456 caract.	3	291-435 caract.	3
8 bits Datos Binarios	0-140 bytes	1	0-133 bytes	1
	41-266 bytes	2	134-254 bytes	2
	267-399 bytes	3	255-381 bytes	3
USC-2 Alfabeto	0-70 caract.	1	0-66 caract.	1
	71-132 caract.	2	67-126 caract.	2
	133-198 caract.	3	127-189 caract.	3

La tabla A-1 asume para el alfabeto GSM 7-bits que solamente caracteres que puedan ser codificados con una sola septeta son utilizados. Si un carácter que se codifica en dos septetas (usando el código de escape para tabla de extensión) es utilizado, cuenta como dos caracteres en el cálculo de esta longitud.

Nota: Los valores de la tabla A-1 incluye una concatenación de encabezado en todos los mensajes, cuando el mensaje no puede ser enviado en un solo mensaje de protocolo SMS.

Tabla de Mapeo de Caracteres

GSM 7-bit	UCS-2	Character name
0x00	0x0040	COMMERCIAL AT
0x01	0x00a3	POUND SIGN
0x02	0x0024	DOLLAR SIGN
0x03	0x00a5	YEN SIGN
0x04	0x00e8	LATIN SMALL LETTER E WITH GRAVE
0x05	0x00e9	LATIN SMALL LETTER E WITH ACUTE
0x06	0x00f9	LATIN SMALL LETTER U WITH GRAVE

GSM 7-bit	UCS-2	Character name
0x07	0x00ec	LATIN SMALL LETTER I WITH GRAVE
0x08	0x00f2	LATIN SMALL LETTER O WITH GRAVE
0x09	0x00c7	LATIN CAPITAL LETTER C WITH CEDILLA
0x0a	0x000a	control: line feed
0x0b	0x00d8	LATIN CAPITAL LETTER O WITH STROKE
0x0c	0x00f8	LATIN SMALL LETTER O WITH STROKE
0x0d	0x000d	control: carriage return
0x0e	0x00c5	LATIN CAPITAL LETTER A WITH RING ABOVE
0x0f	0x00e5	LATIN SMALL LETTER A WITH RING ABOVE
0x10	0x0394	GREEK CAPITAL LETTER DELTA
0x11	0x005f	LOW LINE
0x12	0x03a6	GREEK CAPITAL LETTER PHI
0x13	0x0393	GREEK CAPITAL LETTER GAMMA
0x14	0x039b	GREEK CAPITAL LETTER LAMDA
0x15	0x03a9	GREEK CAPITAL LETTER OMEGA
0x16	0x03a0	GREEK CAPITAL LETTER PI
0x17	0x03a8	GREEK CAPITAL LETTER PSI
0x18	0x03a3	GREEK CAPITAL LETTER SIGMA
0x19	0x0398	GREEK CAPITAL LETTER THETA
0x1a	0x039e	GREEK CAPITAL LETTER XI
0x1b	xxx	escape to extension table
0x1c	0x00c6	LATIN CAPITAL LETTER AE
0x1d	0x00e6	LATIN SMALL LETTER AE
0x1e	0x00df	LATIN SMALL LETTER SHARP S
0x1f	0x00c9	LATIN CAPITAL LETTER E WITH ACUTE
0x20	0x0020	SPACE
0x21	0x0021	EXCLAMATION MARK
0x22	0x0022	QUOTATION MARK
0x23	0x0023	NUMBER SIGN
0x24	0x00a4	CURRENCY SIGN
0x25	0x0025	PERCENT SIGN
0x26	0x0026	AMPERSAND
0x27	0x0027	APOSTROPHE
0x28	0x0028	LEFT PARENTHESIS
0x29	0x0029	RIGHT PARENTHESIS
0x2a	0x002a	ASTERISK
0x2b	0x002b	PLUS SIGN
0x2c	0x002c	COMMA
0x2d	0x002d	HYPHEN-MINUS
0x2e	0x002e	FULL STOP
0x2f	0x002f	SOLIDUS
0x30	0x0030	DIGIT ZERO
0x31	0x0031	DIGIT ONE
0x32	0x0032	DIGIT TWO
0x33	0x0033	DIGIT THREE
0x34	0x0034	DIGIT FOUR
0x35	0x0035	DIGIT FIVE
0x36	0x0036	DIGIT SIX
0x37	0x0037	DIGIT SEVEN
0x38	0x0038	DIGIT EIGHT

GSM 7-bit	UCS-2	Character name
0x39	0x0039	DIGIT NINE
0x3a	0x003a	COLON
0x3b	0x003b	SEMICOLON
0x3c	0x003c	LESS-THAN SIGN
0x3d	0x003d	EQUALS SIGN
0x3e	0x003e	GREATER-THAN SIGN
0x3f	0x003f	QUESTION MARK
0x40	0x00a1	INVERTED EXCLAMATION MARK
0x41	0x0041	LATIN CAPITAL LETTER A
0x42	0x0042	LATIN CAPITAL LETTER B
0x43	0x0043	LATIN CAPITAL LETTER C
0x44	0x0044	LATIN CAPITAL LETTER D
0x45	0x0045	LATIN CAPITAL LETTER E
0x46	0x0046	LATIN CAPITAL LETTER F
0x47	0x0047	LATIN CAPITAL LETTER G
0x48	0x0048	LATIN CAPITAL LETTER H
0x49	0x0049	LATIN CAPITAL LETTER I
0x4a	0x004a	LATIN CAPITAL LETTER J
0x4b	0x004b	LATIN CAPITAL LETTER K
0x4c	0x004c	LATIN CAPITAL LETTER L
0x4d	0x004d	LATIN CAPITAL LETTER M
0x4e	0x004e	LATIN CAPITAL LETTER N
0x4f	0x004f	LATIN CAPITAL LETTER O
0x50	0x0050	LATIN CAPITAL LETTER P
0x51	0x0051	LATIN CAPITAL LETTER Q
0x52	0x0052	LATIN CAPITAL LETTER R
0x53	0x0053	LATIN CAPITAL LETTER S
0x54	0x0054	LATIN CAPITAL LETTER T
0x55	0x0055	LATIN CAPITAL LETTER U
0x56	0x0056	LATIN CAPITAL LETTER V
0x57	0x0057	LATIN CAPITAL LETTER W
0x58	0x0058	LATIN CAPITAL LETTER X
0x59	0x0059	LATIN CAPITAL LETTER Y
0x5a	0x005a	LATIN CAPITAL LETTER Z
0x5b	0x00c4	LATIN CAPITAL LETTER A WITH DIARESIS
0x5c	0x00d6	LATIN CAPITAL LETTER O WITH DIARESIS
0x5d	0x00d1	LATIN CAPITAL LETTER N WITH TILDE
0x5e	0x00dc	LATIN CAPITAL LETTER U WITH DIARESIS
0x5f	0x00a7	SECTION SIGN
0x60	0x00bf	INVERTED QUESTION MARK
0x61	0x0061	LATIN SMALL LETTER A
0x62	0x0062	LATIN SMALL LETTER B
0x63	0x0063	LATIN SMALL LETTER C
0x64	0x0064	LATIN SMALL LETTER D
0x65	0x0065	LATIN SMALL LETTER E
0x66	0x0066	LATIN SMALL LETTER F
0x67	0x0067	LATIN SMALL LETTER G
0x68	0x0068	LATIN SMALL LETTER H
0x69	0x0069	LATIN SMALL LETTER I
0x6a	0x006a	LATIN SMALL LETTER J

GSM 7-bit	UCS-2	Character name
0x6b	0x006b	LATIN SMALL LETTER K
0x6c	0x006c	LATIN SMALL LETTER L
0x6d	0x006d	LATIN SMALL LETTER M
0x6e	0x006e	LATIN SMALL LETTER N
0x6f	0x006f	LATIN SMALL LETTER O
0x70	0x0070	LATIN SMALL LETTER P
0x71	0x0071	LATIN SMALL LETTER Q
0x72	0x0072	LATIN SMALL LETTER R
0x73	0x0073	LATIN SMALL LETTER S
0x74	0x0074	LATIN SMALL LETTER T
0x75	0x0075	LATIN SMALL LETTER U
0x76	0x0076	LATIN SMALL LETTER V
0x77	0x0077	LATIN SMALL LETTER W
0x78	0x0078	LATIN SMALL LETTER X
0x79	0x0079	LATIN SMALL LETTER Y
0x7a	0x007a	LATIN SMALL LETTER Z
0x7b	0x00e4	LATIN SMALL LETTER A WITH DIARESIS
0x7c	0x00f6	LATIN SMALL LETTER O WITH DIARESIS
0x7d	0x00f1	LATIN SMALL LETTER N WITH TILDE
0x7e	0x00fc	LATIN SMALL LETTER U WITH DIARESIS
0x7f	0x00e0	LATIN SMALL LETTER A WITH GRAVE
0x1b 0x14	0x005e	CIRCUMFLEX ACCENT
0x1b 0x28	0x007b	LEFT CURLY BRACKET
0x1b 0x29	0x007d	RIGHT CURLY BRACKET
0x1b 0x2f	0x005c	REVERSE SOLIDUS
0x1b 0x3c	0x005b	LEFT SQUARE BRACKET
0x1b 0x3d	0x007e	TILDE
0x1b 0x3e	0x005d	RIGHT SQUARE BRACKET
0x1b 0x40	0x007c	VERTICAL LINE
0x1b 0x65	0x20ac	EURO SIGN

Los caracteres GSM de 7 bits que usan el código de escape para una combinación de dos septetos están representadas en esta tabla con las representaciones hexadecimales de los dos septetos separadamente, En los mensajes codificados, los septetos están codificados juntos sin alineación extra a los octetos limites.

A.2.0 Direccionamiento de Mensajes

La sintaxis de conexión de cadena URL que especifica la dirección están descritas en la tabla A-2

Tabla A-2 Cadena de conexión para direcciones de mensaje

String	Definition
smsurl	::== "sms://" address_part
address_part	::== foreign_host_address | local_host_address
local_host_address	::== port_number_part
port_number_part	::== ":" digits
foreign_host_address	::== msisdn | msisdn port_number_part
msisdn	::== "+" digits | digits
digit	::== "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
digits	::== digit | digit digits

Ejemplos de cadenas de conexión URL validas son:

sms://+358401234567

sms://+358401234567:6578

sms://:3381

Cuando este adaptador es usado en el método Connector.open( ) es transferido a URL con esta sintaxis, debe retornar una instancia implementando la interface javax.wireless.messaging.MessageConnecion.

A.2.1 Especificando direcciones de destino

En esta cadena de conexión URL, la parte MSISDN identifica el número de teléfono destino y el número de puerto de los números de direcciones de la aplicación que se encuentran especificadas en la especificación en el GSM 3.40 SMS (secciones 9.2.3.24.3 y 9.2.3.24.4). El mismo mecanismo es usado, por ejemplo para mensajes WAP WDP.

Cuando el número de puerto este presente en la dirección el TP-Usuario-Datos, del SMS debe contener un Usuario-Datos-Encabezado con el elemento de información de esquema de direccionamiento de puerto de la aplicación.

Cuando la dirección de destino no contiene un número de puerto, el TP-Usuario-Datos no debe contener el encabezado de direccionamiento de puerto de la aplicación. Las aplicaciones Java no pueden recibir este tipo de mensaje, pero será manejado en

el dispositivo destino de manera normal, por ejemplo, los mensajes de texto, serán presentados al usuario final.

A.2.2 Conexión de Modo Servidor y Modo Cliente

Los mensajes pueden ser enviados usando este API, a través de tipos de conexiones cliente o servidor. Cuando un mensaje identificando él número de puerto es enviado desde un tipo de MessageConnection servidor, el número de puerto que se origina en el mensaje es establecido en el número de puerto del MessageConnection. Esto permite al destinatario enviar una respuesta a los mensajes que serán recibidos por esta MessageConnection.

Sin embargo, cuando un tipo de conexión de mensaje cliente es utilizado para enviar un mensaje con número de puerto, el número de puerto originado es establecido al valor de implementación específica y cualquier posible mensaje recibido a este número de puerto no son entregados a la conexión de mensaje.

Solo el modo servidor puede ser utilizado para recepción de mensajes. Cualquier mensaje que se espera obtener una respuesta del dispositivo destino debe utilizar el apropiado modo servidor.

A.2.3 Manejo de mensajes recibidos

Cuando un SMS es recibido por una aplicación, estos son removidos de la memoria SIM/ME donde estos son almacenados.

Si la información de mensaje debe ser almacenada persistentemente, entonces la aplicación es responsable de guardarlos. Por ejemplo, la aplicación podría salvar la información del mensaje utilizando la facilidad RMS del MIDP API u otro tipo de mecanismo disponible.

El protocolo GSM SMS no garantiza el ordenamiento cuando múltiples mensajes son enviados. Cuando un mensaje extenso es separado en varias secciones GSM SMS como se especifica anteriormente, el ordenamiento se maneje correctamente cuando

son concatenados automáticamente a un solo mensaje. Si la aplicación envía múltiples mensajes al mismo destinatario, estos pudiesen no ser entregados en el orden correcto. La aplicación debe estar programada para poder resolver este problema correctamente. Sin embargo aun cuando el orden pueda cambiar durante la entrega en la red, la implementación debe garantizar que los mensajes sean entregados a la aplicación en el mismo orden como fueron recibidos por la implementación en el terminal destino.

A.3.0 Dirección Centro de Servicio de Mensajes Cortos

Las aplicaciones podrían necesitan obtener la dirección SMSC(Centro de Servicio de Mensajes Cortos) para decidir que número de destino utilizar, por ejemplo la aplicación puede necesitar hacer esto porque esta utilizando números de servicio para aplicaciones servidores, los cuales podrían no estar consistentes en todas las redes y SMSCs.

La dirección de SMSC utilizada para mandar el mensaje debe ser habilitada utilizando System.getProperty con el nombre de propiedad descrita en la tabla A-32.

Tabla A-3 Nombre de propiedad y descripción para direcciones SMSC

Property name	Description

`wireless.messaging.sms.smsc`	The address of the SMS expressed using the syntax expressed by the msisdn item of thefollowing BNF definition:
	`msisdn ::== "+" digits | digits`
	`digit ::== "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |"8" | "9"`
	`digits ::== digit | digit digits`

A.4.0 Uso de números de puerto

La aplicación receptora en un dispositivo está identificado con el número de puerto incluido en el mensaje. Cuando ser abre un MessageConnection modo servidor, la aplicación específica el número de puerto que se usará para la recepción de mensajes.

La primera aplicación en localizar un número de puerto dado, lo obtendrá. Si otras aplicaciones tratan de localizar el mismo número de puerto mientras este es utilizado por la primera aplicación se desplegara un error IOEXception cuando intenten

abrir el MessageConnection. La misma regla se aplica si un número puerto es utilizado por un sistema de aplicación en el dispositivo. En este caso, las aplicaciones Java no serán capaces de utilizar ese número de puerto.

Como se especifica en el estándar GSM 03.40, los números de puertos son divididos en rangos. El IANA(Autoridad de Números Asignados para Internet) controla uno de los rangos. Si un autor de aplicación desea asegurar que una aplicación siempre pueda utilizar un número de puerto especifico, entonces puede ser registrado como IANA. De otra manera, el autor puede escoger elegir aleatoriamente un número del rango disponible y esperar que el número no este siendo utilizado por otra aplicación que pueda estar instalada en el mismo dispositivo. Esto es exactamente de la misma forma que los números de puerto son comúnmente utilizados en TCP y UDP en Internet.

A.5.0 Tipos de Mensaje

Los mensajes SMS pueden ser enviados usando los tipos de mensaje TextMessage o el BinaryMessage del API. La codificación utilizada en el protocolo SMS están definidos en el estándar GSM 03.38 (Parte 4 SMS Esquema de Codificación de Datos)

Cuando la aplicación utiliza el tipo de mensaje de texto. El TP-Datos-Codificación-Esquema en el SMS debe indicar el alfabeto de 7-bit o el UCS-2. El TP-Usuario-Dato debe ser codificado apropiadamente utilizando el alfabeto seleccionado. El Alfabeto de 7 bits debe ser utilizado para codificar si la cadena que es dada por la aplicación solamente contiene caracteres que están presentes en el alfabeto de 7 bits GSM. Si la cadena dada por la aplicación contiene al menos un carácter que no este presente en el alfabeto de 7 bits GSM, la codificación UCS-2 debe ser utilizado.

Cuando la aplicación utiliza un tipo de mensaje binario, el TP-Dato-Codificación-Esquema en el SMS debe indicar datos de 8 bits.

La aplicación es responsable de asegurar que la carga útil del mensaje encaje en un mensaje SMS cuando se codifique como esta definido en esta especificación. Si la

aplicación intenta enviar un mensaje con una carga útil demasiada extensa, el método MessageConnection.send( ) desplegara un IllegalArgumentException y el mensaje no será enviado. Esta especificación contiene la información que aplicaciones necesitan para determinar la máxima carga útil para el tipo de mensaje que estén intentando enviar.

Todos los mensajes enviados vía esta API deben ser enviados como mensajes clase 1 Especificación GSM 3.40 SMS sección -.2.3.9 “Tp-Protocolo-Identificador”

A.6.0 Restricciones en números de puerto por mensajes SMS

Por razones de seguridad, a las aplicaciones Java no se les permite mandar mensajes a los números de puerto listados en la tabla A-4. Las implementaciones deben desplegar un SecurityException en el método MessageConnection.send ( ) si una aplicación intenta enviar un mensaje a cualquiera de estos número de puerto.

Tabla A-4 Números de puerto restringidos para Mensajes SMS

Port number	Description
2805	WAP WTA secure connection-less session service
2923	WAP WTA secure session service
2948	WAP Push connectionless session service (client side)
2949	WAP Push secure connectionless session service (client side)
5502	Service Card reader
5503	Internet access configuration reader
5508	Dynamic Menu Control Protocol
5511	Message Access Protocol
5512	Simple Email Notification
9200	WAP connectionless session service
9201	WAP session service
9202	WAP secure connectionless session service
9203	WAP secure session service
9207	WAP vCal Secure
49996	SyncML OTA configuration
49999	WAP OTA configuration

Anexo B: Hoja técnica del V600

MOTOROLA
intelligence everywhere™

Motorola V600

Developer Reference Sheet

Technical Specifications

Band/Frequency	GSM 850/900/1800/1900 GPRS
Region	Global
Technology	WAP 2.0, J2ME, SMS, EMS, MMS, AOL/OICQ IM
Connectivity	CE Bus, Bluetooth
Dimensions	49 x 89 x 24
Weight	90 g
Display	Internal: 176 x 220 External: 96 x 32
Operating System	Motorola
Chipset	i250S1

Key Features

- Quad band
- Integrated digital camera (VGA quality)
- Sleek, compact design
- Metal cover with end user changeable covers
- Video clip playback (10 sec)
- 5 MB of end user memory
- External CLI display (96 x 32, with blue font)
- Situational lighting (front and side)
- Games (preloaded and downloadable)
- PIM functionality with Picture Caller ID
- Downloadable themes (ringtones, wallpaper, screensaver)
- Voice memo
- MIDI speaker
- WAP 2.0
- Email clients: POP3, SMTP, IMAP4

J2ME™ Information

CLDC v1.0 and MIDP v2.0 compliant	
Maximum MIDlet suite size	100kb
Heap size	800kb
Maximum record store size	64kb
MIDlet storage available	Up to 5Mb
Interface connections	HTTP, Socket UDP, Serial port
Maximum number of sockets	4
Supported image formats	.PNG, .JPEG
Double buffering	Supported
Encoding schemes	ISO8859_1, ISO10646
Input methods	Multitap, iTAP
Additional API's	JSR 120, JSR 135 Phonebook
Audio	MIDI, WAV, AMR MP3

Related Information

Motorola Developer Information:
Developer Resources at
http://www.motocoder.com/

Tools:
CodeWarrior® Wireless Studio v7.0
J2ME™ SDK version v4.0
Motorola Messaging Suite v1.1

Documentation;
Creating Media for the Motorola V300, V400, V500 and V600 Handsets

References:
J2ME™ specifications:
http://www.java.sun.com/j2me
MIDP v2.0 specifications:
http://www.java.sun.com/products/midp
CLDC v1.0 specifications:
http://www.java.sun.com/products/cldc
WAP forum: http://www.wap.org
EMS standards: http://www.3GPP.org

Anexo C: Costo Económico del Proyecto

El costo total del proyecto viene dado por todos los costos involucrados para la realización de éste, varios factores en conjunto ayudaron a la elaboración del todo el proyecto. Estos factores son:

a) Costos de materiales y dispositivos
b) Costo de horas hombre trabajadas
c) Costo de instalación
d) Costo al usuario.

Dichos factores conformarán el costo total del proyecto, a continuación se detallan cada uno de los factores y su costo individual, además de una breve descripción de éstos, y lo que implica en el desarrollo del proyecto.

Costos de materiales y dispositivos

El costo de los materiales se define como todos los materiales utilizados para elaborar el proyecto, tanto como dispositivos ya diseñados como dispositivos que se diseñaron exclusivamente para el proyecto, tales como tarjeras controladoras y otros. A continuación se presenta la lista de materiales y componentes utilizados con su respectivo precio individual y al final la suma de todos los precios.

Lista de Componentes y dispositivos utilizados

Dispositivo	Cantidad	Valor
MODEM GSM	1	$ 270.00
Microcontrolador PIC	2	$ 3.00
Cable Terminal Móvil	1	$ 25.00
Terminal móvil	1	$ 200.00
Tarjeta programadora de PIC	1	$ 20.00
Tarjeta Controladora	1	$ 20.00
Inversor de voltaje	1	$ 40.00
	Valor Total	**$ 578.00 dólares**

Costo de horas hombre trabajadas

El costo de horas hombres trabajadas, se calcula estableciendo las horas diarias trabajadas para la elaboración del proyecto, además, multiplicadas por la cantidad de días en las que se trabajó en él, posteriormente se dispone a calcular el valor de cada hora, la cual será multiplicada por el número de horas totales trabajadas y eso da como resultado el costo de horas hombre trabajadas para el desarrollo y elaboración del proyecto.

-Calculo de horas trabajadas

Se han trabajado un promedio de una hora y media diaria en un total de 2OO días, aproximadamente, desde que se aprobó el trabajo de graduación, por lo que al final si se multiplica el valor de horas trabajas promedio diarias por el número aproximado de días trabajados se tiene un total de 200 horas trabajas

1.5 horas X 200 días = 300 horas

-Calculo del valor de hora de trabajo.

Para el calculo del valor de hora trabajada, se toma un sueldo base de $800 dólares, lo cual es un valor aproximado del sueldo de un programador y desarrollador de aplicaciones de este tipo, el número de horas trabajadas por cada empleado según la ley salvadoreña es de 44 horas semanales, en un mes típico constituido de 4 semanas, se tiene un total de 176 horas mensuales y dividiendo el sueldo base entre el total de número de horas trabajas mensualmente se tiene el valor de la hora trabajada.

$800 dólares / 176 horas mensuales = $4.54 dólares por cada hora trabajada

-Calculo de costo total de horas hombres trabajadas

Una vez se tiene el valor de horas trabajadas y el valor de cada una de estas horas, se multiplican los valores y se obtiene el valor del costo total aproximado de

horas hombre trabajadas para la realización del proyecto por cada integrante del grupo de proyecto.

300 horas X $4.54 dólares por hora = $1362.00 dólares

El costo total de horas hombre trabajas por cada integrante es de **$1362.00 dólares.**

Ahora el costo total de horas hombre trabajadas por el grupo de trabajo es, el costo total de un integrante multiplicado por tres, debido a que el grupo está comprendido por tres integrantes dando como resultado un valor de **$4086 dólares**

Costo de Instalación

Para el desarrollo del proyecto se requirió la instalación del equipo en el vehículo para esto se tuvo que adaptar el vehículo para poder agregar dichos componentes y poder hacerlo funcional en conjunto, el costo de los cambios hechos al vehículo para poder interconectarlo los demás dispositivos constituye el costo de instalación total del proyecto, este costo es aproximadamente **$30 dólares** aproximadamente.

Costo al usuario

Para calcular el costo al usuario, se toman un número promedio de mensajes cortos utilizados diariamente por el usuario para recibir notificaciones del estado del vehículo, y para habilitar y deshabilitar el sistema, dicho número de mensajes cortos se multiplican por el número de días de un mes, y posteriormente el valor total de mensajes cortos mensuales se multiplican por el valor promedio de un mensaje corto en el mercado, lo que nos da como resultado el costo de utilización del proyecto para el usuario.

10 SMS diarios (aproximado) X 31 días = 310 mensajes cortos mensuales
310 SMS mensuales X $0.03 dólares = **9.3 dólares mensuales**