ESCUELA POLITÉCNICA DEL EJÉRCITO

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y AUTOMATIZACIÓN Y CONTROL

PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA

IMPLEMENTACIÓN DE UN PROTOTIPO PARA GESTIÓN DE PRODUCTOS Y PEDIDOS EN DISTRIBUIDORAS UTILIZANDO SMS DE TECNOLOGÍA GSM

.

CRISTIAN GONZALO KAROLYS TOVAR
DIEGO FRANCISCO NIAMA BONIFAZ

SANGOLQUÍ - ECUADOR
2009

C E R T I F I C A C I Ó N

Certificamos que el presente proyecto de grado titulado ” IMPLEMENTACIÓN DE UN PROTOTIPO PARA GESTIÓN DE PRODUCTOS Y PEDIDOS EN DISTRIBUIDORAS UTILIZANDO SMS DE TECNOLOGÍA GSM” fue realizado en su totalidad por los señores Cristian Gonzalo Karolys Tovar y Diego Francisco Niama Bonifaz, bajo nuestra dirección.

Ing. Gonzalo Olmedo
DIRECTOR

Ing. Julio Larco
CODIRECTOR

R E S U M E N

Este proyecto constituye el diseño e implementación de un prototipo registrador de pedidos utilizando mensajes SMS con tecnología GSM, que reemplace de forma económica y ágil a los actuales equipos registradores de pedido.

El prototipo consta de una aplicación desarrollada en NetBeans, que se instala en un dispositivo móvil, que puede ser un smartphone o un celular. Está aplicación permite registrar pedidos de clientes, para luego enviarlos por medio de mensajes SMS a un modem GSM. Adicionalmente, la aplicación almacena pedidos en el dispositivo móvil con fines de seguridad para prevenir posibles fallos en el equipo receptor de los mensajes SMS o para almacenar los pedidos que no puedan ser enviados por falta de cobertura de señal de la operadora.

El prototipo también consta de un modem GSM ubicado en la Distribuidora, el cual recibe el pedido realizado en el dispositivo móvil, para luego procesar el mismo y así determinar el mensaje SMS de respuesta al dispositivo móvil.

En un HMI desarrolla en Visual Basic, se realiza el procesamiento del mensaje SMS, que consiste en verificar en una base de datos si existe o no producto disponible, si el código de cliente es correcto, si el dispositivo que envía el pedido por medio de un mensaje SMS está autorizado por la Distribuidora. También genera la impresión de la factura correspondiente, siempre que todos los datos verificados en el procesamiento del mensaje SMS sean los correctos, de esta manera se agiliza el proceso de facturación, ya que es automático.

D E D I C A T O R I A

A mis padres que lucharon incansablemente

para darme una educación *y* han sido mi guía y apoyo

todos los días de mi vida.

Cristian G. Karolys T.

A G R A D E C IM IE N T O

A m is padres, por todo el apoyo incondicional que m e han entregado a lo largo de m i vida, que su esfuerzo, su paciencia y su dedicación m e permitieron llegar a culm inar m is estudios.

A m i novia Cristina que fue un increíble apoyo no solo en este proyecto sino a lo largo de m i vida convirtiéndose en uno de los pilares fundam entales de m i vida.

A Diego por ser no solo un com pañero sino un am igo y un apoyo.

A los ingenieros Gonzalo Olm edo y Julio Larco por abrirnos las puertas y guiarnos en la consecución de este proyecto y en nuestra form ación, y a todos aquellos m aestros que no fueron guiando día a día para ser las m ejores personas y profecionales.

Cristian G. Karolys T.

D E D I C A T O R IA

A m is padres César y Elvia,

gracias a su fé en m í,

he logrado alcanzar todas m is m etas.

A m is herm anos Sebastian, Karina y Esteban,

a m is tías Sarita y Fabiola,

quienes han sido el apoyo m ás im portante de m i vida.

Diego F. Niam a B.

AGRADECIMIENTO

Agradezco a Dios y la Dolorosa por haberme dado la oportunidad de haber crecido en el seno de una familia en la que siempre estuvo presente el cariño de mis padres y hermanos, y principalmente los valores que me han servido de guía durante mi vida.

A mis Padres, por haberme dado todo su apoyo, por su sacrificio al alejarse del amor de toda la familia para poder brindarnos mejores oportunidades a mis hermanos y a mí. Gracias a ellos soy el hombre en el que me he convertido.

A mis hermanos, Sebastián, Karina, y Esteban, por siempre ser mi apoyo, por sus consejos, por ser las personas que siempre me supieron dar fuerza para seguir adelante en ausencia de mis padres.

A mis amigos, en especial Andrés, Carlos y Cristóbal, que siempre me dieron palabras de aliento para alcanzar cualquier meta propuesta. A Cristian, por haber compartido conmigo todos estos años de estudio.

A los ingenieros Gonzalo Olmedo y Julio Larco, por habernos servido de guía para la ejecución de este proyecto y a todos aquellos profesores que a más de enseñarme alguna materia, me enseñaron a ser mejor persona.

Diego F Niama B.

PRÓLOGO

La entrega ágil de los productos de una distribuidora a sus diferentes centros de expendio ha generado mejoras a los sistemas de preventa; las mismas que han resultado ser ineficientes por la demora con la que llega la orden de pedido a las distribuidoras, produciendo un uso ineficiente de recursos. El uso de mecanismos modernos en nuestro medio para solucionar este tipo de problema son muy costosos por la tecnología importada que se utiliza, y por la utilización de servicios de comunicación; siendo necesario buscar mecanismos tecnológicos económicos que permitan solucionar esta necesidad por lo que se ha realizado este proyecto.

Para un mejor entendimiento se presenta una introducción al sistema de GSM (CAPÍTULO 1) el cual va a permitir un mejor entendimiento de la manera en la que se transmiten los datos y de esta poder describir el funcionamiento de los SMS en la tecnología GSM (CAPÍTULO 2).

Para el manejo de modem GSM se realiza una explicación de los comandos AT utilizados (CAPÍTULO 3).

Para un mejor entendimiento de la plataforma de programación de la aplicación del dispositivo móvil, se describe las principales características funcionales de NetBeans y J2ME (CAPÍTULO 4).

La explicación del diseño y funcionamiento del prototipo receptor (modem) tanto en hardware como en software se puede encontrar en los CAPÍTULO 5 y CAPÍTULO 6.

Los principales controles utilizados para el manejo de la base de datos, comunicación, e impresión de facturas y funcionamiento de la HMI desarrollada en Visual Basic e instalada en la PC, se detalla en el CAPÍTULO 7.

El desarrollo y el funcionamiento de la HMI desarrollada en NetBeans, así como las principales operaciones que realiza la aplicación se puede encontrar en el CAPÍTULO 8.

Para observar y entender el funcionamiento del sistema se realizaron varias pruebas las cuales se describen en el CAPÍTULO 9, así como los resultados de las mismas.

Finalmente como culminación del proyecto se llego a las conclusiones y recomendaciones descritas en el CAPÍTULO 10.

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

G L O S A R IO

A M S: A pplication M anagem ent System

A P I: A pplication Program m ing Interface

B S C : B ase Station C ontroller

B S S: B ase Station Subsystem

B T S: B ase Transceiver Station

C C G : C onsejo de C ooperación del G olfo

C D C : C onnected D evice C onfiguration

C L D C : C onnected Lim ited D evice C onfiguration

C V M : C om pact V irtual M achine

D C E: D ata circuit-term inating equipm ent

D T E: D ata term inal equipm ent

D T M F: D ual T one M ultifrequency

E E P R O M : E lectrically E rasable Program m able R ead-O nly M em ory

E P R O M : E rasable Program m able R ead-O nly M em ory

F D M A: F requency D ivision M ultiple A ccess.

G M S K : G aussian M inim um Shift K eying

G P R S: G eneral packet radio service

G S M : G roupe S peciale M obile

G U I: G raphical user interface

H M I: H um an M achine Interface

J2E E: J ava 2 E nterprise E dition

J2M E: J ava 2 M icro E dition

J2S E: J ava 2 S tantard E dition

J A D : J ava A pplication D escriptor

J A R : J ava A rchive

J N I: J ava N ative Interface

J V M : J ava V irtual M achine

J V M D I: JavaTM V irtual M achine D ebug Interface.

M S C : M obile S ervices S w itching C enter

M T : S hort M essage M obile Term inated P oint-to-P oint

N S S: N etw ork and S w itching S ubsystem

P D A : P ersonal D igital A ssistant

PDU: Protocol Data Unit

RAM: Random Access Memory

RMI: Remote Method Invocation

RMS: Record Management System

ROM: Read Only Memory

RSS: Radio SubSystem

SIM: Subscriber Identity Module

SM MO: ShortMessageMobile Originated Point-to-Point.

SM-AL: Short Message Aplication Layer

SM-LL: Short Message Lower Layers

SM-RL: Short Message Relay Layer

SMS: Short Message Service

SM-TL: Short Message Transfer Layer

TDMA: Time Division Multiple Access

TPDU: Transfer Protocol Data Units

UD: User Data

UDH: User Data Header

UDHI: User Data Header Indicator

UDHL: User Data Header Length

UDL: User Data Length

UIT: Unión Internacional de Telecomunicaciones

UMTS: Universal Mobile Telecommunications System

USART: Universal synchronous asynchronous receiver transmitter

USB: Universal Serial Bus

USI: Interfaz Serial Universal

W-CDMA: Wideband Code Division Multiple Access

WMA: Wireless Messaging API.

CAPÍTULO 1

SISTEMA GSM

1.1. INTRODUCCIÓN

A lo largo de la evolución de las telecomunicaciones celulares, los diferentes sistemas se han desarrollado sin el beneficio de especificaciones normalizadas. Esto presenta muchos problemas directamente relacionados con la compatibilidad, especialmente con el desarrollo de la tecnología de radio digital. La norma GSM se destina a hacer frente a estos problemas [1].

1.2. DEFINICIÓN DE GSM

El Sistema Global para las Comunicaciones Móviles (GSM, proviene de "*Groupe Speciale Mobile*") es un sistema estándar, completamente definido, para la comunicación mediante teléfonos móviles que incorporan tecnología digital. Por ser digital cualquier cliente de GSM puede conectarse a través de su teléfono con su ordenador y puede hacer, enviar y recibir mensajes por e-mail, faxes, navegar por Internet, acceso seguro a la red informática de una compañía (LAN/Intranet), así como utilizar otras funciones digitales de transmisión de datos, incluyendo el Servicio de Mensajes Cortos (SMS) o mensajes de texto.

GSM se considera, por su velocidad de transmisión y otras características, un estándar de segunda generación (2G).

1.3. ALCANCE MUNDIAL Y PORCENTAJE DE USO

Según el promotor de GSM, la Asociación GSM (GSMA o *GSM Association)*, este estándar es el más extendido en el mundo, con un 82% de los terminales mundiales en uso [2].

GSM cuenta con más de 3.000 millones de usuarios en 212 países distintos, siendo el estándar predominante en Europa, América del Sur, Asia y Oceanía, y con gran extensión en América del Norte [3].

La ubicuidad del estándar GSM ha sido una ventaja tanto para consumidores (beneficiados por la capacidad de itinerancia[1] y la facilidad de cambio de operador sin cambiar de terminal, simplemente cambiando la tarjeta SIM) como para los operadores de red (que pueden elegir entre múltiples proveedores de sistemas GSM, al ser un estándar abierto que no necesita pago de licencias).

En GSM se implementó por primera vez el servicio de mensajes cortos de texto (SMS), que posteriormente fue extendido a otros estándares. Además, en GSM se define un único número de emergencias a nivel mundial, el 112, que facilita que los viajeros de cualquier parte del mundo puedan comunicar situaciones de emergencia sin necesidad de conocer un número local.

1.4. ARQUITECTURA DE LA RED GSM

En la Figura. 1.1. Arquitectura de la red GSM se muestra de manera resumida la arquitectura de la red GSM. Esta arquitectura es más compleja y dispone de más elementos que los presentados en esta figura. El objetivo de este proyecto es describir el servicio SMS a nivel de aplicación, sin entrar en demasiados detalles de la red subyacente. La arquitectura GSM está constituida por tres partes:

1. **Subsistema Radio** (**RSS**, ***Radio SubSystem***). Cubre la comunicación entre las estaciones móviles (**MS**) y las estaciones base (**BS**). El interfaz radio entre ellas se denomina Um.

2. **Subsistema de estaciones base (BSS, *Base Station Subsystem*).** Esta constituido por los siguientes elementos:

   (a) BTS (*Base Transceiver Station*): emisor, receptor y antena. Procesa los canales de radio (Interfaz Um).

[1] Itinerancia: capacidad de un dispositivo para moverse de una zona de cobertura a otra

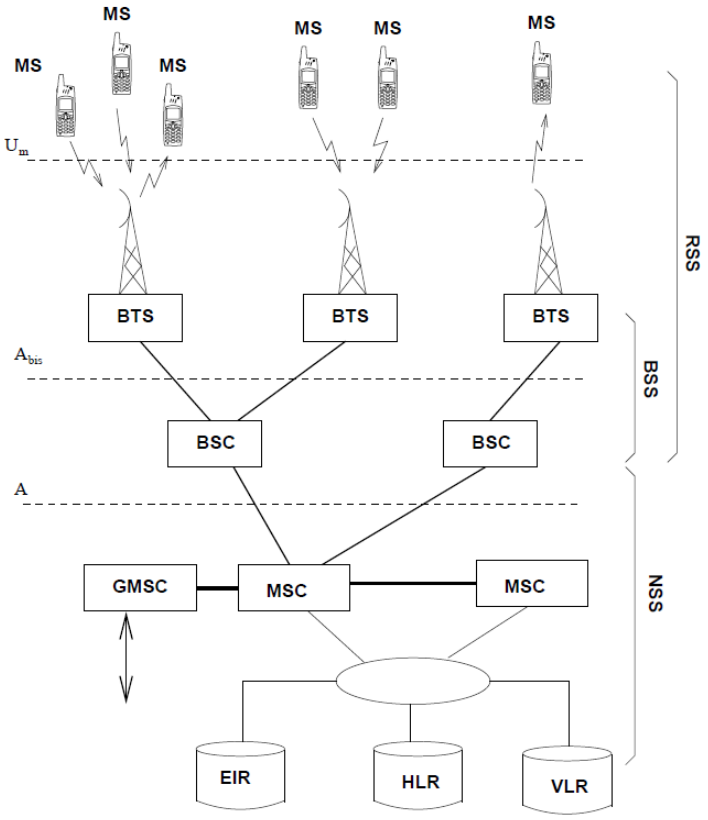


Figura. 1.1. Arquitectura de la red GSM.

(b) BSC (*Base Station Controller*): Handover, control de las BTS, mapeo de canales de radio sobre los canales terrestres. Por un lado se comunica con las BTS a través de la interfaz Abis, con canales de 16 kbps y por otro lado se comunica con los MSC a través de la interfaz A, con canales de 64 kbps.

Este subsistema hace de interfaz entre la estación móvil y la parte de red.

3. Subsistema de red y conmutación (NSS, Network and Switching Subsystem)

Conmutación, gestión de la movilidad, interconexión con otras redes y control del sistema. Esta es la parte más compleja, siendo sus elementos principales los siguientes:

(a) MSC (*Mobile Services Switching Center*): centro de conmutación, entre otras muchas funciones.

(b) Bases de datos:

i. HLR (*Home Location Register*)

ii. VLR (*Visitor Location Register*)

iii. EIR (*Equipment Identity Registry*)

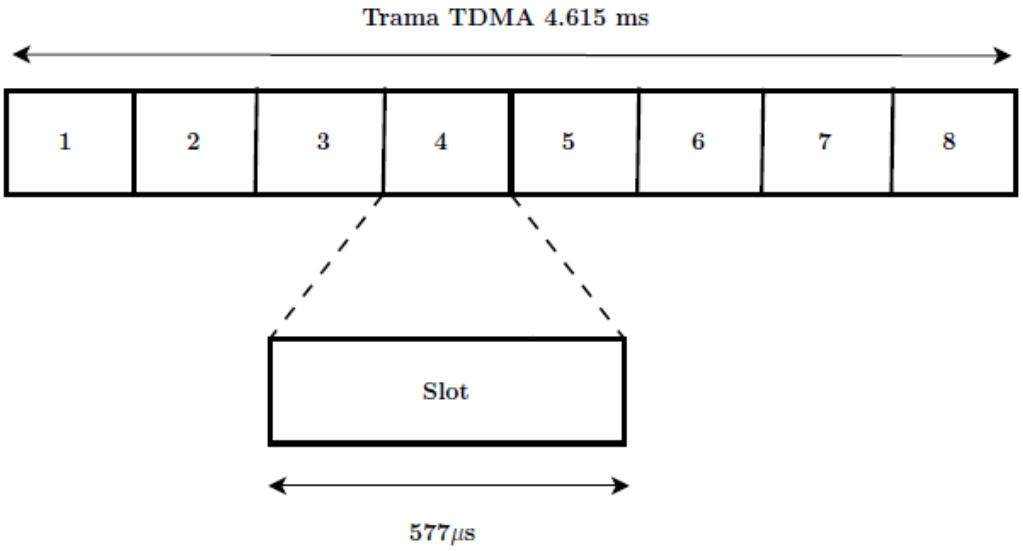


Figura. 1.2. Trama TDMA y slots.

1.5. INTERFAZ DE RADIO Um

Para la transmisión de bits entre la BTS y una MS se utilizan canales físicos, caracterizados por un slot y una portadora. Dentro de cada portadora se multiplexan en el tiempo 8 ranuras, formando una trama TDMA como la que se muestra en la

Figura. 1.2. A un nivel superior, los canales físicos se dividen en:

- Canales de tráfico: llevan la voz y/o los datos.
- Canales de control: señalización y señales de control.

Los canales de tráfico pueden ser de 2, 4,4, 8 ó 9,6 kbps. Para el servicio SMS se utilizan canales de control [4].

1.6. CARACTERÍSTICAS TÉCNICAS

Las bandas de frecuencia designadas por la UIT (*International Telecommunication Union*) para la operación de GSM en la mayor parte del mundo, incluyendo Europa, Medio Oriente, África y gran parte de Asia, son de 900 MHz y 1800 MHz. Algunos países en

America, incluyendo Ecuador, usan las bandas de 850 MHz y 1900 MHz. Las principales características del GSM son:

- Utiliza dos bandas de 25 MHz para la transmisión y recepción de información empleando FDD2.
- GSM 850 utiliza la banda de 824-849 MHz para el uplink y la banda de 869-894 MHz para el *downlink*.
- El acceso al medio se realiza mediante TDMA/FDMA (*Time Division Multiple Access/Frequency Division Multiple Access*).
- Tiene 124 canales, y cada canal puede dar servicio a 8 o 16 usuarios a la vez.
- Ancho de banda del canal 200 Khz.
- La modulación empleada es GMSK (*Gaussian Minimum Shift Keying*).
- La velocidad máxima del canal de radio es 270.833 kbps.
- Duración de un bit de 3,692 msec.
- La longitud de una trama es de 4,615 msec, y la longitud de un slot de tiempo 577 µsec.
- Codificación de la voz: RPE-LPT 13 kbps (*Regular Pulse Excitation-Long Term Prediction*).
- Potencia de salida de 20 mW a 20W.

1.7. SERVICIOS

1.7.1. Servicios básicos

Hay dos tipos básicos de servicios que se pueden ofrecer a través de GSM: telefonía (tele-servicios) y datos (servicios portadores). Los servicios de telefonía son principalmente servicios de voz que proveen a los suscriptores la capacidad total (incluyendo el equipo terminal necesario) para comunicarse con otros suscriptores. Los servicios de datos proveen la capacidad necesaria para transmitir datos entre dos puntos de acceso, creando una interfaz de red. Además de la telefonía normal y llamadas de emergencia, GSM también soporta los siguientes servicios de suscriptor:

- DTMF (*Dual-tone multifrequency*)
- Fax
- SMS (*Short Message Service*)
- Buzón de voz

1.7.2. Servicios suplementarios

GSM soporta un amplio conjunto de servicios suplementarios que pueden complementar a los servicios de telefonía y datos. A continuación se listan algunos servicios suplementarios:

- Transferencia de llamada
- Llamada en espera
- Conferencia
- Bloqueo llamadas salientes
- Bloqueo llamadas entrantes
- Facturación detallada
- Marcación abreviada
- Identificación de llamadas [5].

1.8. BENEFICIOS QUE OFRECE EL SISTEMA GSM

1.8.1. Beneficios para el usuario

Si analizamos a los involucrados en este sistema vemos que tanto usuarios como operadores son beneficiados. Entre los beneficios al usuario tenemos:

- **Cobertura:** ya que se encuentra en más de 210 países del mundo. Este beneficio permite al usuario permanecer accesible por medio del mismo dispositivo móvil dentro de su país y en muchos otros más.
- **Selección:** debido a la gran cantidad de usuarios que GSM tiene los fabricantes de equipos se han visto en la obligación de presentar en el mercado una gran cantidad y variedad de modelos lo que le da al usuario la posibilidad de escoger.
- **Calidad de voz**: GSM presenta un servicio de transmisión de voz a una calidad muy alta.
- **Flexibilidad**: La opción de cambiar de equipo sin pasar por las molestias de configurar este nuevo es posible gracias a GSM quien trabaja con una tarjeta conocida como Módulo de Identidad del Abonado (SIM), la cual no solo nos ahorra el trabajo de configuración de nuestro dispositivo nuevo, adicionalmente nos da beneficios como es

el no perder servicios de suscripción personalizados tales como mensajería. Además, gracias a esta tarjeta se le facilita al usuario el poder cambiar de operador GSM y mantener el mismo teléfono; la flexibilidad de la tarjeta SIM hace que las redes de datos basadas en GSM, tales como las GPRS, sean atractivas para diversas aplicaciones de datos.

- **Servicios innovadores:** gracias a las características de GSM esta ha sido pionera en la presentación de servicios como el de mensajes cortos (SMS) que soportan mensajes de texto y contenidos tales como ringtones. Otro servicio muy importante es el de capacidad de roaming que permite a los usuarios gozar de los servicios dentro y fuera de su área local o país.
- **Movilidad:** Hasta hace muy poco tiempo, el concepto de movilidad estaba asociado exclusivamente al terminal telefónico, pero actualmente se relaciona al usuario que utiliza sus servicios.
- **Movilidad del terminal:** El usuario dispone de un teléfono asociado a un número de la red y puede utilizarlo en cualquier lugar con cobertura.
- **Movilidad personal:** El usuario, y no el terminal, está asociado a un número de teléfono de la red que, de forma inteligente, le sigue en sus desplazamientos, pudiendo hacer uso de él desde cualquier teléfono, sea éste fijo o móvil.

1.8.2. Beneficios para el operador

Analizando el sistema desde la perspectiva del operador observamos muchos beneficios, entre los cuales podemos resaltar:

- **Economías de escala:** Debido a la gran acogida que ha tenido este sistema en el mundo la demanda de equipos terminales e infraestructura es también presentada en gran escala, lo que hace de este un mercado muy interesante para proveedores y desarrolladores de aplicaciones, y gracias a esta producción en gran volumen, los costos se reducen, permitiendo así a los operadores de GSM el establecer precios para sus servicios a un nivel más competitivo.
- **Cobertura:** a causa de la cantidad de países en los que este sistema opera, las empresas proveedoras de este servicio pueden ofrecer a sus clientes la posibilidad de usar su mismo equipo y número celular dentro y fuera del país, opción que es muy atractiva

para personas de negocios mismos que generan una gran parte de los ingresos a las operadoras, y así logran tener una mayor competitividad y aumentan sus ingresos.

- **Flexibilidad:** por la gran atención que ha prestado la comunidad GSM a las normas se ha logrado asegurar la interoperabilidad entre dispositivos e infraestructura de diversas marcas, lo que permite que el operador no este esclavizado a un solo proveedor tecnológico, adicionalmente tanto equipos como infraestructura se presentan para las bandas de frecuencia más populares, entre ellas las de 850 y 1900 MHz, permitiendo al operador seleccionar el que mejor se ajuste a sus necesidades de espectro y de mercado.
- **Eficiencia:** al dividir los canales en slots de tiempo permite a la operadora trabajar con mas llamadas, lo que representa mas usuarios que generan mas ganancias para la empresa, así se está utilizando los recursos de una forma mucho mejor que los sistemas anteriormente presentados, y si adicionalmente se utiliza optimizaciones tales como el Codec Adaptativo a Múltiples Velocidades (AMR), proveen un incremento adicional de casi tres veces más llamadas de voz simultáneas que la tecnología GSM básica.
- **Capacidad de actualizarse:** GSM es estructurado de modo tal que cada paso subsiguiente aprovecha el paso anterior logrando así que un equipo nuevo sea compatible con uno antiguo, lo que preserva tanto las inversiones como los clientes a lo largo de la migración haciéndola de una forma ligera y no traumante. Las normas que rigen la capacidad de actualización y la interoperabilidad de GSM están coordinadas y respaldadas por organizaciones internacionales clave tales como el Proyecto de Asociación para la Tercera Generación (3GPP), 3G Américas entre otros [6].

CAPÍTULO 2

ESTUDIO DE TRANSMISIÓN Y RECEPCIÓN DE MENSAJERÍA SMS EN BASE A TECNOLOGÍA GSM.

2.1. SERVICIO SMS

El servicio de mensajes cortos o SMS (*Short Message Service*) es un servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos (también conocidos como mensajes de texto, o más coloquialmente, textos o mensajitos) entre teléfonos móviles, teléfonos fijos y otros dispositivos de mano. SMS fue diseñado originariamente como parte del estándar de telefonía móvil digital GSM, pero en la actualidad está disponible en una amplia variedad de redes, incluyendo las redes 3G.

Un mensaje SMS es una cadena alfanumérica de hasta 160 caracteres de 7 bits, y cuyo encapsulado incluye una serie de parámetros. En principio, se emplean para enviar y recibir mensajes de texto normal, pero existen extensiones del protocolo básico que permiten incluir otros tipos de contenido, dar formato a los mensajes o encadenar varios mensajes de texto para permitir mayor longitud (formatos de SMS con imagen de Nokia, tonos IMY de Ericsson, estándar EMS para dar formato al texto e incluir imágenes y sonidos de pequeño tamaño).

El servicio SMS permite transferir un mensaje de texto entre una estación móvil (MS) y otra entidad (SME) a través de un centro de servicio (SC) Figura. 2.1.

El servicio final ofrecido es una comunicación extremo-extremo entre la estación móvil (MS) y la entidad (SME). La entidad puede ser otra estación móvil o puede estar situado en una red fija. En el caso de envío de un mensaje entre dos móviles, ambas partes son estaciones móviles. Cuando se envía un mensaje para solicitar algún tipo de servicio (o realizar alguna votación, sobre todo en los concursos de la TV, que ahora están tan de

moda), un extremo es una estación móvil y la otra es un servidor que atiende las peticiones (o anota los votos).

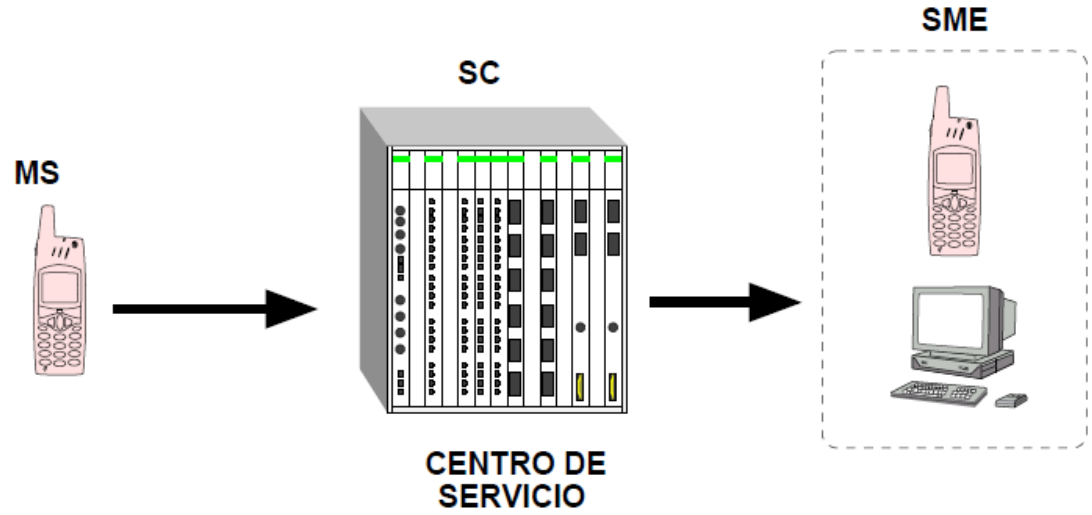


Figura. 2.1. Servicio SMS.

El servicio SMS se divide en dos servicios Básicos Figura. 2.2

1. **SM MT (Short Message Mobile Terminated Point-to-Point).** Servicio de entrega de un mensaje desde el **SC** hasta una **MS**, obteniéndose un informe sobre lo ocurrido.

2. **SM MO (Short Message Mobile Originated Point-to-Point).** Servicio de envío de un mensaje desde una **MS** hasta un **SC**, obteniéndose un informe sobre lo ocurrido

2.2. ARQUITECTURA DE RED

La estructura básica de la red para el servicio SMS se muestra en la Figura. 2.3. Las entidades involucradas son las siguientes:

- **MS**: Estación móvil.
- **MSC**: Centro de conmutación.
- **SMS-GMSC**: **MSC** pasarela para el servicio de mensajes cortos (Servicio SM MT).
- **SMS-IWMSC**: **MSC** de interconexión entre PLMN y el SC (Servicio SM MO).
- **SC**: Centro de Servicio.

- **HLR, VLR**

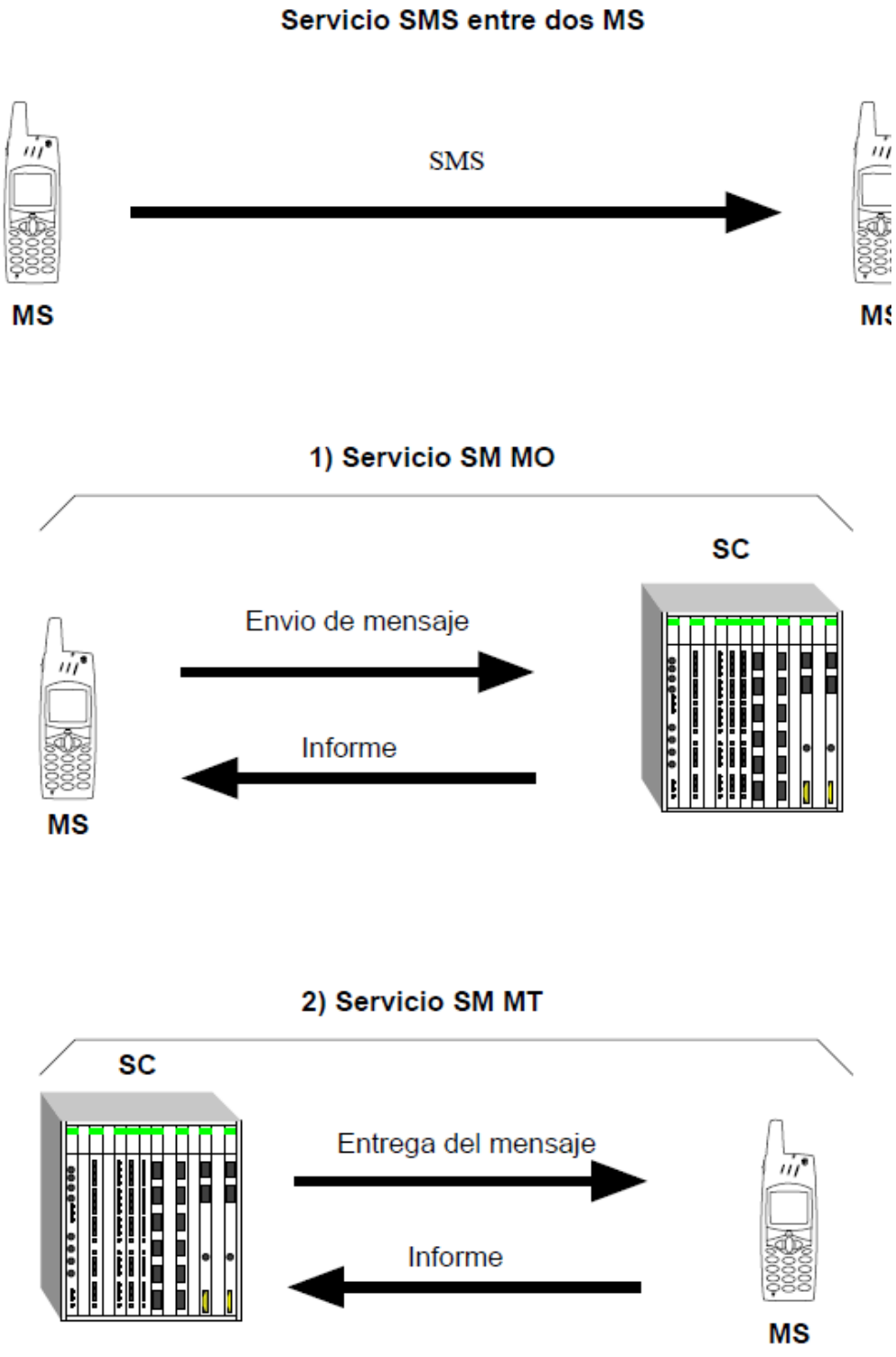


Figura. 2.2. Servicios básicos SM MT y SM MO

Para la descripción detallada de la arquitectura, se utiliza un modelo de capas, en el que cada capa o nivel proporciona un servicio a la capa superior, y este servicio se implementa mediante el protocolo correspondiente. La arquitectura se divide en 4 capas (Figura. 2.4):

- **SM-AL (Short Message Aplication Layer): Nivel de aplicación**.
- **SM-TL (Short Message Transfer Layer): Nivel de transferencia**. Servicio de transferencia de un mensaje corto entre una **MS** y un **SC** (en ambos sentidos) y obtención de los correspondientes informes sobre el resultado de la transmisión. Este servicio hace abstracción de los detalles internos de la red, permitiendo que el nivel de aplicación pueda intercambiar mensajes.
- **SM-RL (Short Message Relay Layer): Nivel de repetición**. Proporciona un servicio al nivel de transferencia que le permite enviar TPDU (Transfer Protocol Data Units) a su entidad gemela.
- **SM-LL (Short Message Lower Layers): Niveles inferiores.**

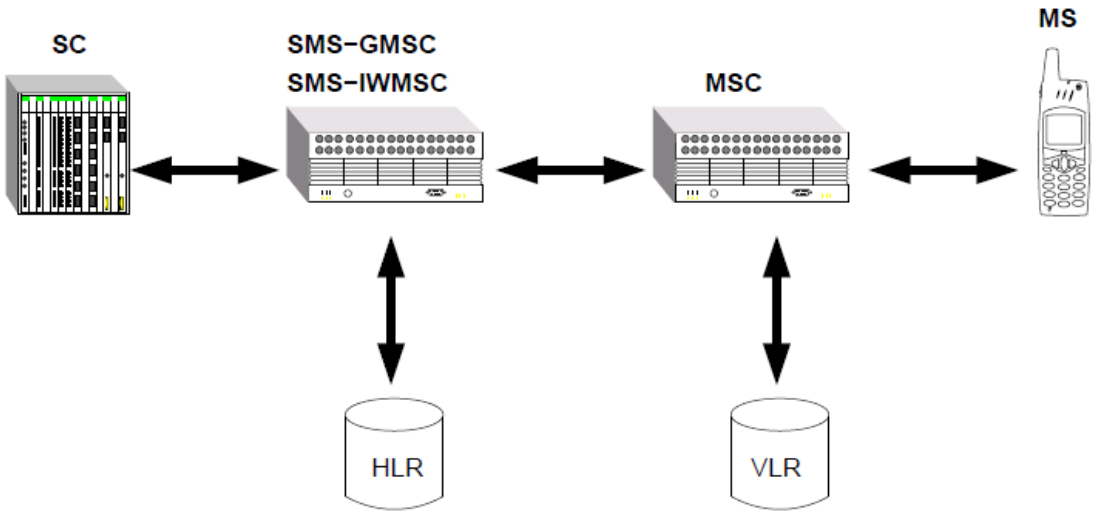


Figura. 2.3. Estructura básica de la red para la transferencia de mensajes cortos.

2.3. NIVEL SM-TL Y PROTOCOLO SM-TP

Cada capa proporciona los servicios a la capa superior utilizando un protocolo. Se definen los protocolos **SM-TP** y **SM-RP**, que se corresponden con las capas **SM-RL** y

SM-TL. El nivel de interés de este trabajo es el **SM-TL**, que es el que se usará para enviar y recibir SMS.

El servicio proporcionado por la **capa SM-TL** permite al nivel de aplicación enviar mensajes a su entidad gemela, recibir mensajes de ella así como obtener informes sobre el estado de transmisiones anteriores. Se utilizan las siguientes 6 PDUs (Figura. 2.5):

- **SMS-DELIVER**: Transmitir un mensaje desde el **SC** al **MS.**
- **SMS-DELIVER-REPORT**: Error en la entrega (si lo ha habido).
- **SMS-SUBMIT**: Trasmitir un mensaje corto desde el **MS** al **SC.**
- **SMS-SUBMIT-REPORT**: Error en la transmisión (Si lo ha habido).
- **SMS-STATUS-REPORT**: Transmitir un informe de estado desde el **SC** al **MS.**
- **SMS-COMMAND**: Transmitir un comando desde el **MS** al **SC.**

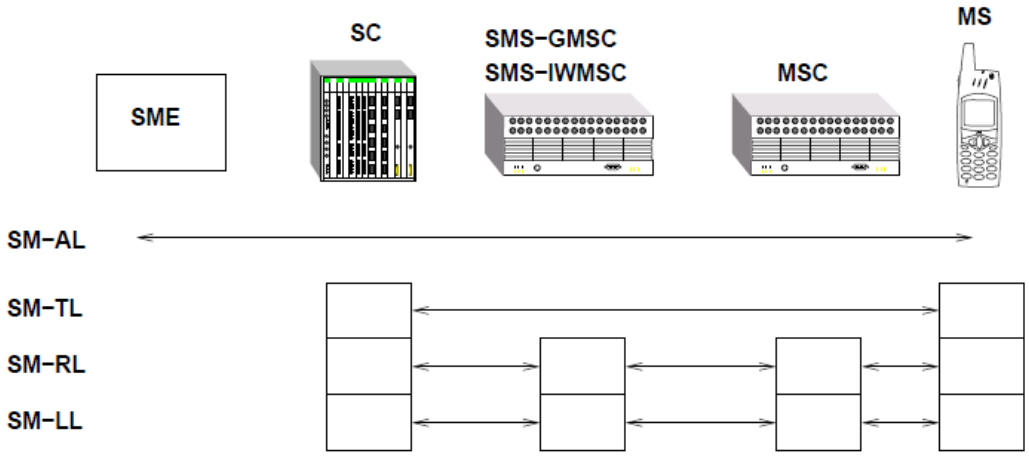


Figura. 2.4. Niveles y servicios para el envío de mensajes cortos

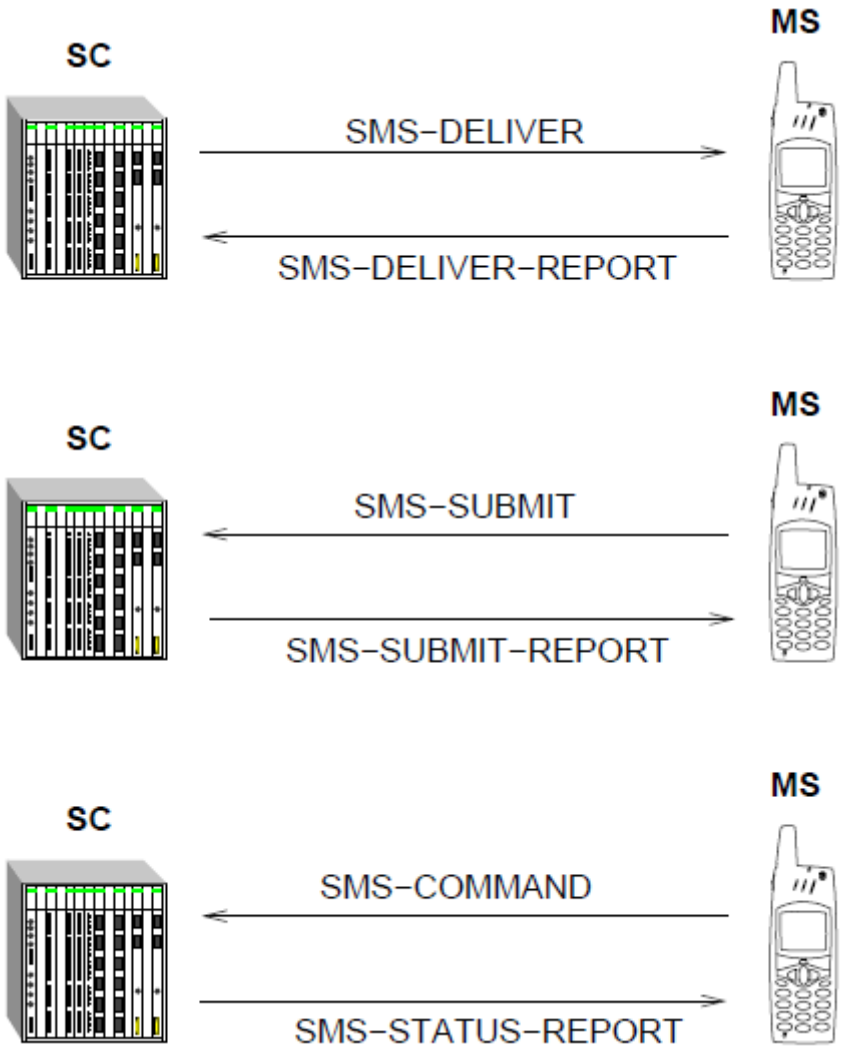


Figura. 2.5. Las 6 PDUs del SM-TP.

2.3.1. Sms-submit

La estructura de la PDU **SMS-SUBMIT** se muestra en la Figura 2.6. Los campos que la componen son los siguientes:

- **SCA**: Número de teléfono del Centro de Servicio (SC). La estructura detallada se muestra en la Figura. 2.. Consta de los siguientes campos:
  - **Longitud**: Número de dígitos del teléfono del SC.
  - **Tipo de número**: Indica si se trata de un número nacional o internacional:
    - _ **81h**: Nacional
    - _ **91h**: Internacional
  - **Dígitos BCD**: Número de teléfono del SC, en dígitos BCD

- **PDU-TYPE**: Contiene información sobre el **tipo de PDU**
  - **RP**: Existe camino de respuesta. RP=0 en tramas de tipo SMS-SUBMIT
  - **UDHI**: Indica si el campo UD contiene sólo el mensaje corto (UDHI=0) o si existe una cabecera antes del mensaje corto (UDHI=1)
  - **SRR**: Informe de estado no solicitado (SRR=0) o sí solicitado (SRR=1)
  - **VPF**: Indica si el campo **VP** está o no presente
  - **RD**: Rechazar o no duplicados
  - **MTI**: Tipo de mensaje:

BIT 1	BIT 0	Descripción
0	0	SMS-DELIVER
0	0	SMS-DELIVER-REPORT
0	1	SMS-SUBMIT
0	1	SMS-SUBMIT-REPORT
1	0	SMS-STATUS-REPORT
1	0	SMS-COMMAND
1	1	Reservado

Tabla. 2.1. Valores del campo MTI.

- **MR**: Parámetro para identificar el mensaje
- **DA**: Dirección del SME destino (número de teléfono)
- **PID**: Identificación del protocolo de la capa superior
- **DCS**: Identificación del tipo de codificación dentro de los datos de usuario
- **VP**: Periodo de validez del mensaje
- **UDL**: Longitud del campo **UD**
- **UD**: Datos de usuario

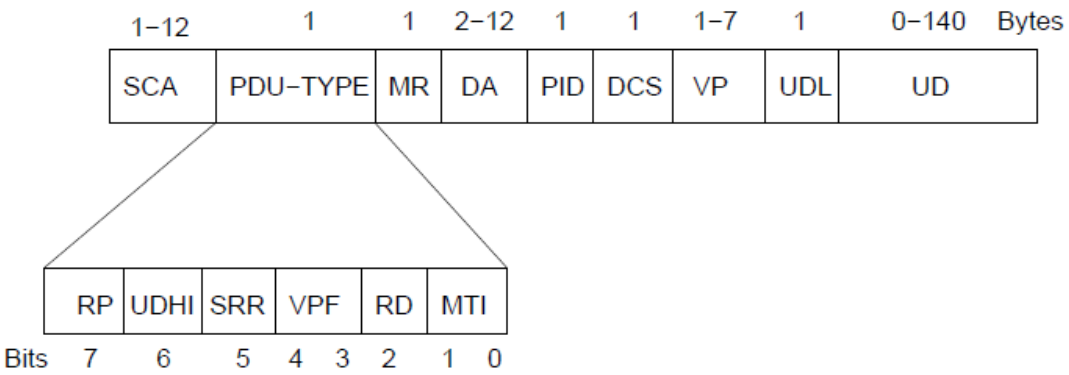


Figura. 2.6. Trama SMS-SUBMIT

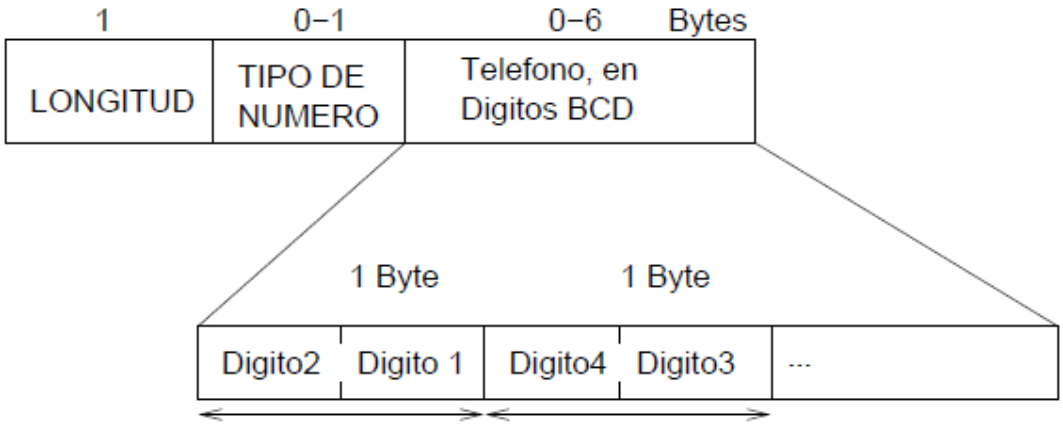


Figura. 2.7. Detalle del campo SCA

2.3.2. Sms-deliver

Esta trama, transmitida desde el **SC** hasta el **MS**, tiene una estructura similar a **SMS-SUBMIT** y se muestra en la Figura. 2.. Los nuevos campos que aparecen son los siguientes:

OA: Dirección del SME que envía el mensaje

SCTS: Marca de tiempo de cuando el centro de servicio recibió el mensaje

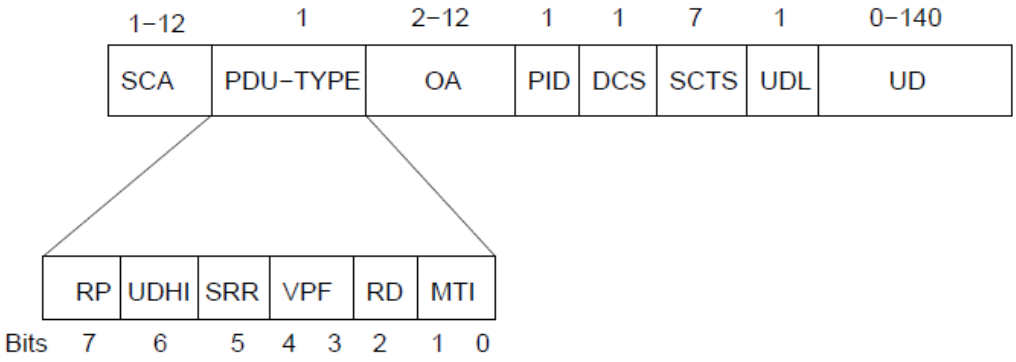


Figura. 2.8. Trama SMS-DELIVER

CAPÍTULO 3

ACCESO A LOS SERVICIOS SMS

3.1. MODEM GSM

El Modem GSM, se trata de un Modem similar a los contenidos en nuestros computadores, pero inalámbricos. Estos Modems tienen la capacidad de utilizar la red GSM para establecer comunicaciones de: Voz, Datos y SMS.

Para su funcionalidad un Modem GSM necesita de algún operador, que le proporcioné un número por contrato o tarjeta, y un número de Centro de Mensajes, para la recepción/envío de mensajes SMS.

Para poder ofrecer estos servicios es necesario diseñar software y hardware que pueda acceder a los servicios SMS. Esto se puede conseguir de varias maneras:

- Algunos teléfonos se pueden conectar directamente a un PC y mediante un software propietario se puede acceder a los datos de móvil (agenda, tarjeta SIM ...), así como enviar y recibir mensajes SMS.
- Utilización de un Modem GSM.

La conexión de un Modem GSM a un Sistema Digital se puede observar en la Figura 3.1.

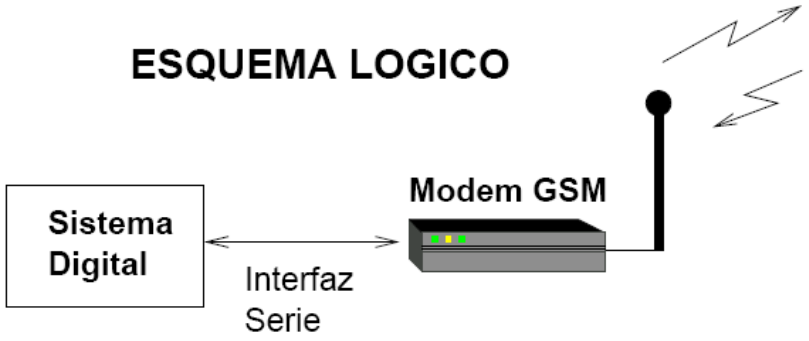


Figura. 3.1. Esquema lógico de conexión de un Modem GSM

Existen dos tipos de Modems GSM, según la aplicación en la que se encuentran implementados:

- *Modems para circuito impreso.* Son Modems de tamaño reducido y perfectamente apantallados que están diseñados para ser incorporados dentro de un circuito impreso y que permiten desarrollar un hardware específico que no dependa de una computadora.
- *Modems para PC.* Tienen un tamaño también bastante reducido, y disponen de un conector serial o USB para conectarse al PC. Son muy útiles para permitir que desde cualquier ordenador de una intranet se puedan enviar mensajes SMS.

3.2. INTERFAZ CON MODEMS: COMANDOS AT

La comunicación con los Modems se realiza a través de una línea serie, y dependiendo del Modem, se pueden usar los niveles definidos por la norma RS232 (Modems para PC) niveles TTL (Modems para Circuito impreso).

El estándar para controlar los Modems se basa en los comandos AT HAYES, o más comúnmente conocidos como comandos AT. El Modem, antes de realizar una conexión con otro Modem, se encuentra en modo comando. En este modo podemos configurar y controlar el Modem utilizando los comandos AT. Una vez establecida la conexión con un Modem remoto, se pasa del modo comando al modo conexión, por lo que la información que le llega al Modem por la línea seria no es interpretada como comandos AT sino como información a transmitir. Una vez terminada la conexión el Modem vuelve al modo comando.

Los comandos AT con cadenas ASCII que comienzan por los caracteres AT y terminan con un retorno de carro. Cada vez que el Modem recibe un comando, lo procesa y devuelve un resultado, que normalmente es una cadena ASCII salvo que hayamos indicado lo contrario. Al estar la comunicación en ASCII, pondremos utilizar un terminal de comunicaciones desde un ordenador para acceder al Modem, bien para configurarlo, bien para hacer pruebas o bien para establecer una comunicación con otro Modem. A continuación se listan algunos comandos AT básicos:

- ATA: responder a una llamada entrante.
- ATD: llamar a un número de teléfono.
- ATE: encendido (1) o apagado (0) de eco de comandos.
- ATF: seleccionar modo de conexión.
- ATH: colgar o descolgar.
- ATL: volumen del altavoz.
- ATM: control del altavoz.
- ATZ: reset del Modem.

Más información sobre los comandos AT se puede encontrar en [4].

3.3. INTERFAZ CON MODEMS GSM

Los Modems GSM no sólo se comportan de forma muy parecida a un modem normal, permitiendo el intercambio de datos con otro Modem y utilizándose los comandos AT originales, sino que incluyen muchas más características. Son como pequeños teléfonos móviles, que incluyen su propia tarjeta SIM para poder funcionar y por tanto permiten gestionar la base de datos de teléfonos, la lista de los mensajes SMS recibidos, enviar mensajes SMS, configurar diversos parámetros, etc.

Para tener acceso a todos esos servicios, y dado que los comandos AT estaban muy extendidos y muy estandarizados, se ha realizado una ampliación, añadiéndose nuevos comandos. Estos nuevos comandos comienzan por las letras AT+, y se denominan comandos AT+.

3.3.1. Comandos AT+

En este apartado se listan los comandos AT+ implementados en el Modem GSM para el presente proyecto. Más información sobre los comandos AT se puede encontrar en [4].

- AT+CMGF (Message Format). Este comando selecciona el formato de entrada y salida para los mensajes SMS, a ser utilizado por el teléfono. El formato para este comando es el siguiente:

  AT+CMGF=< modo >

  Donde < modo > puede ser 0 para el modo PDU o 1 para el modo texto.

  La Figura 3.2 muestra los tipos de formato que permite manejar el Modem utilizado para la aplicación.

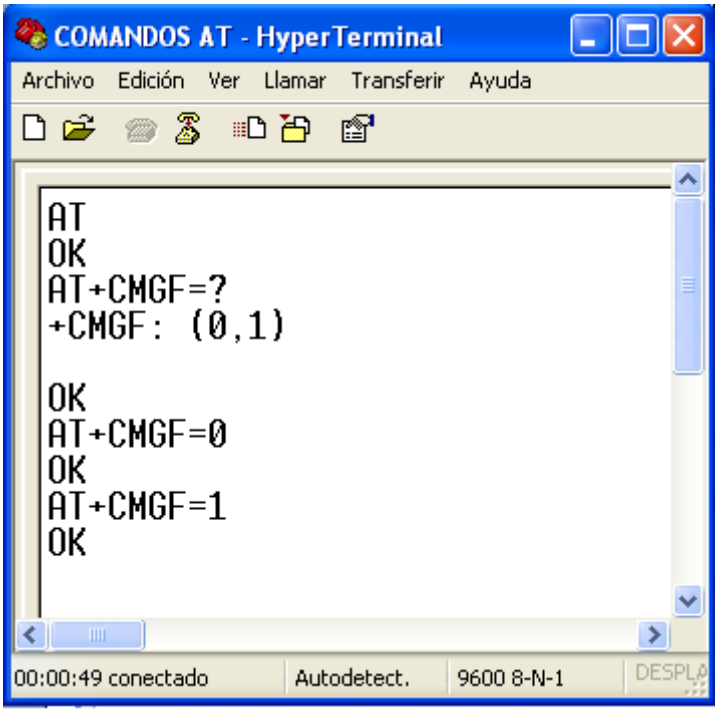


Figura. 3.2. Tipos de formato que permite el Modem utilizado

- AT+CMGS (Send Message). Este comando permite enviar un mensaje SMS desde un equipo terminal a otro, su formato depende del modo en el cual se esté trabajando:

  Para enviar un mensaje en modo texto (AT+CMGF=1), primero se especifica el número de teléfono, seguido de un caracter de retorno carro <CR>. El modem responde enviando el caracter '>' que indica que se puede escribir el mensaje que se quiere

enviar. Para delimitar el mensaje hay que ingresar el caracter <Ctrl+Z> (código ASCII 26).

AT+CMGS=< número ><CR>

Para enviar un mensaje en modo PDU (AT+CMGF=0), se especifica la longitud de la trama PDU en octetos seguido de un caracter de retorno de carro <CR>. El modem responde enviando el caracter '>' y a continuación se coloca la trama PDU seguida del caracter <Ctrl+Z>.

La Figura 3.3 muestra el envió de un SMS en el Modem utilizado para el modo texto y pdu respectivamente.

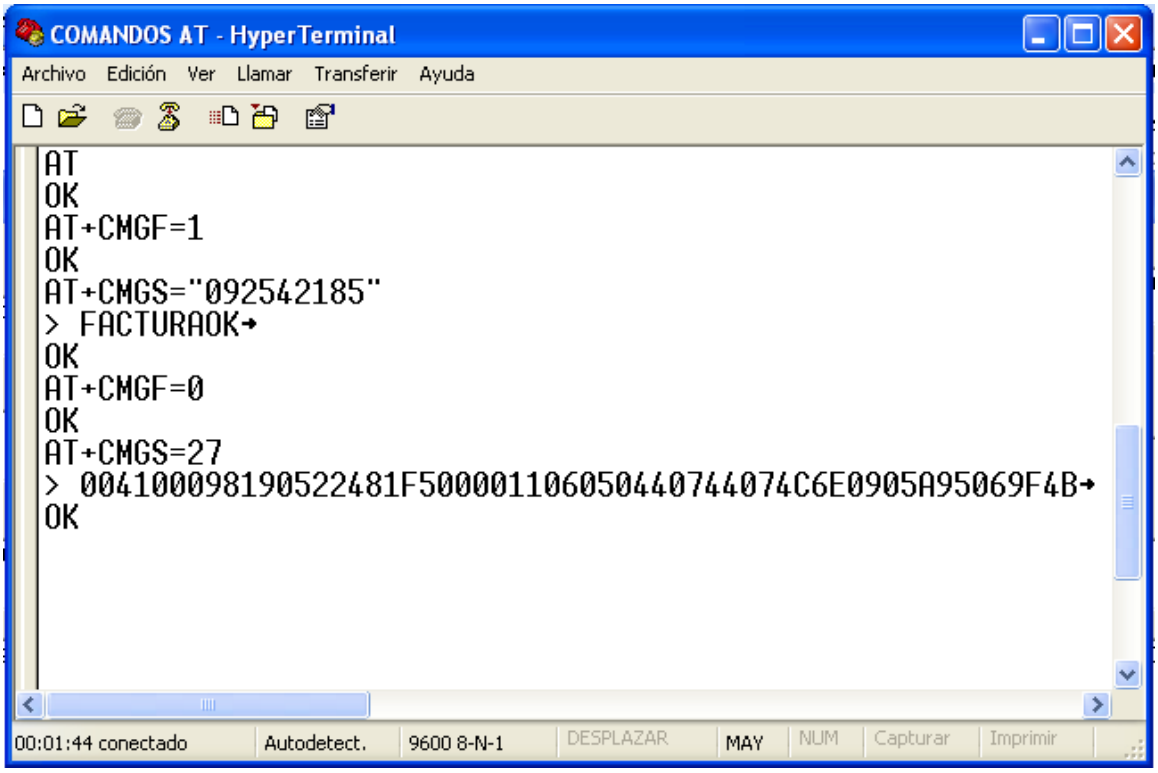


Figura. 3.3. Envío de un SMS para los modos texto y PDU

- AT +CNMI (New Message Indications to DTE). Selecciona el procedimiento o la forma en que la se receptará nuevos mensajes de la red al DTE.

  Sintaxis: AT + CNMI = [<modo> [, <mt> [, <bm > [, <ds> [, <bfr>]]]]]

  <modo>:

  0 Todas las indicaciones del buffer en el adaptador de datos.

  1 No hay indicaciones cuando el enlace DTE-DCE es reservado (on-line data mode).

Todas las indicaciones del buffer en el adaptador de datos cuando el enlace DTE-DCE es reservado.

<mt>:

0 No hay indicaciones de mensajes recibidos al DTE.

1 Las indicaciones de mensajes recibidos son ruteadas al DTE.

2 Los mensajes recibidos son ruteados directamente al DTE usando el código +CMT (Excepto los mensajes de clase 2 indicando +CMTI).

3 Los mensajes de clase 3 son recibidos directamente en el DTE indicando el código +CMT

Y los de otras clases se indican con el código +CMTI.

<bm>:

0 No se rutean mensajes de difusión de celular al DTE.

2 Nuevos mensajes de difusión de celular son ruteados directamente al DTE indicando el código +CMB.

<ds>:

0 Reportes de estado no son ruteados al DTE.

1 Reportes de estado son ruteados al DTE indicando con el código +CDS

<bfr>:

0 Las indicaciones del buffer del adaptador de datos son limpiadas en el DTE cuando <modo> 1 o 2 es introducido.

1 Las indicaciones del buffer del adaptador de datos son borradas en el DTE cuando <modo> 1 o 2 es introducido.

La Figura. 3.4 muestra la manera en la que directamente los mensajes recibidos en el Modem se rutean directamente a la PC por medio del Hyper Terminal, para los modos PDU y texto respectivamente.

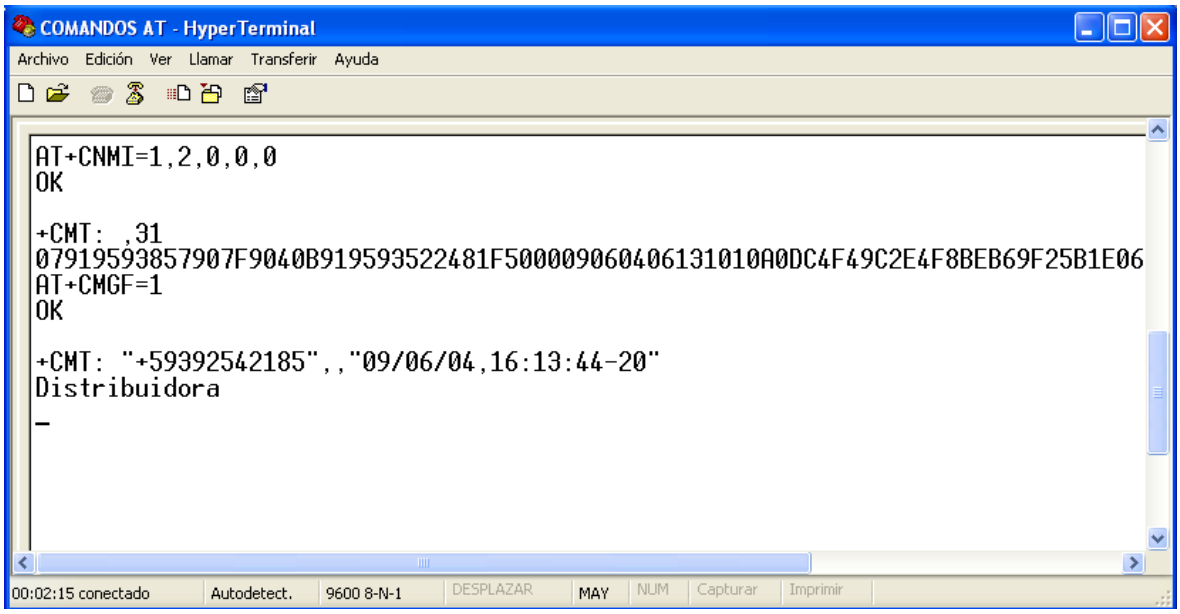


Figura. 3.4. Ejecución del comando CNMI

CAPÍTULO 4

JAVA 2 MICRO EDITION

4.1. INTODUCCIÓN

La plataforma **Java Micro Edition**, o anteriormente **Java 2 Micro Edition** (**J2ME**), es una especificación de un subconjunto de la plataforma Java orientada a proveer una colección certificada de APIs de desarrollo de software para dispositivos con recursos restringidos. Está orientado a productos de consumo como PDAs, teléfonos móviles o electrodomésticos.

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.

J2ME contiene una mínima parte de las APIs de Java. Esto es debido a que la edición estándar de APIs de Java ocupa 20 Mb, 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". J2EE es un superconjunto 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 ha mencionado, contiene varias limitaciones con respecto a J2SE.

4.2. ENTORNO DE EJECUCIÓN

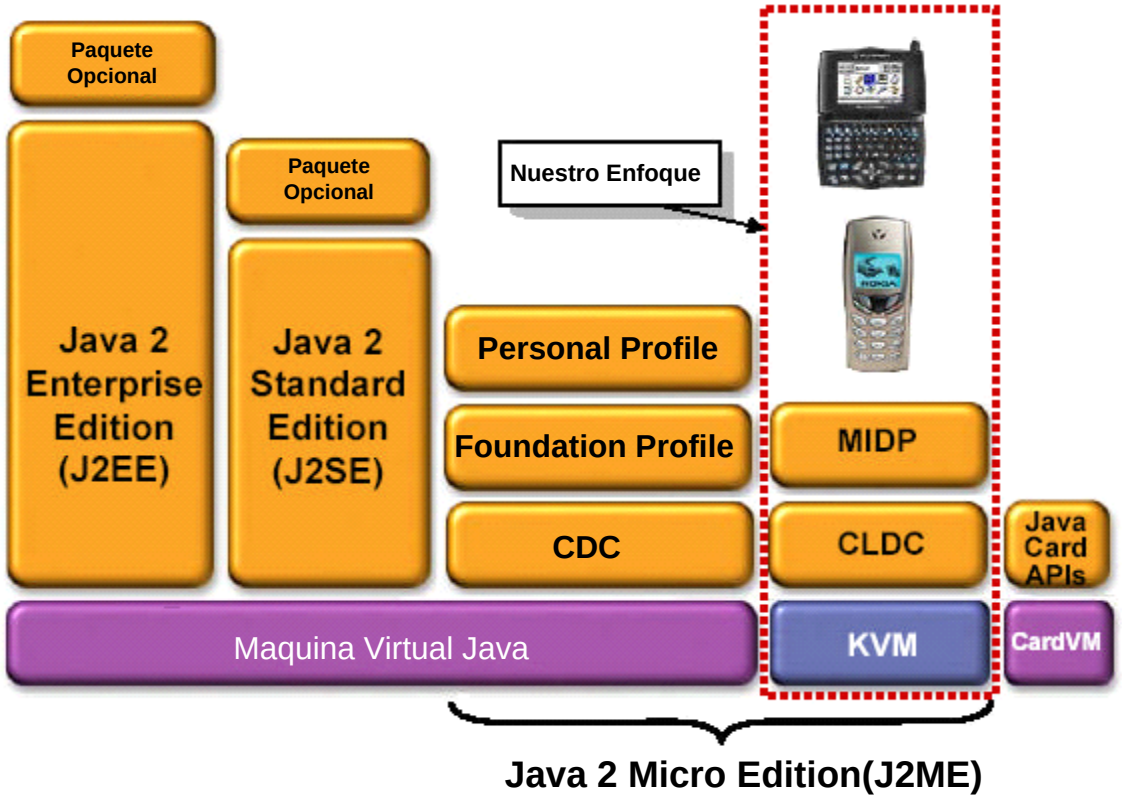


Figura. 4.1. Arquitectura de la plataforma Java 2 de Sun

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

a) Máquina virtual (KVM).

b) Configuración (una configuración es el conjunto mínimo de APIs Java que permiten desarrollar aplicaciones para un grupo de dispositivos CLDC O CDC).

c) Perfil (bibliotecas Java de clases específicas orientadas a implementar funcionalidades de más alto nivel para familias específicas de dispositivos).

d) Paquetes Opcionales.

Cada una de las configuraciones mencionadas anteriormente tiene características propias.

Como consecuencia, cada una requiere su propia máquina virtual. La VM (Virtual Machine) de la configuración CLDC se denomina KVM y la de la configuración CDC se denomina CVM.

4.2.1. Máquina virtual

Una máquina virtual Java 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 y observar las reglas de seguridad y corrección de código definidas para el lenguaje Java. J2ME define dos máquinas virtuales con diferentes características que son:

- **KVM**

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. 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 (*Java Native Interface*) 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 queramos utilizar grupos de hilos utilizaremos los objetos *Colección* para almacenar cada hilo en 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.
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

- **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), Interfaz de depuración de la Máquina Virtual (JVMDI).

4.2.2. Configuraciones

Una configuración es un conjunto mínimo de APIs básicas que permiten desarrollar aplicaciones destinadas a un amplio rango de dispositivos

- **CDC**

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

- **CLDC**

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. 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).

4.2.3. Perfiles

Un perfil es un conjunto de APIs que dotan a una configuración de funcionalidad específica.

Existen unos perfiles que se construyen sobre la configuración CDC y otros sobre la CLDC. Para la configuración CDC tenemos los siguientes perfiles:

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

Y para la configuración CLDC tenemos los siguientes:

- PDA Profile.
- Mobile Information Device Profile (MIDP). Las aplicaciones que se realizan utilizando MIDP reciben el nombre de MIDlets.

También existe una gran cantidad de paquetes opcionales que amplían los perfiles; éstos incluyen Wireless Messaging API*, Mobile Media API*, J2ME RMI Optional Package* y el paquete opcional JDBC para CDC Foundation Profile*, al igual que otros que aún están en el proceso de especificación, tal como J2ME Web Services

4.3. MIDLET

Las aplicaciones J2ME desarrolladas bajo la especificación MIDP, se denominan MIDLets. Las clases de un MIDLet, son almacenadas en *bytecodes* java, dentro de un fichero *.class*. Estas clases, deben ser verificadas antes de su "puesta en marcha", para garantizar que no realizan ninguna operación no permitida. Este preverificación, se debe hacer debido a las limitaciones de la máquina virtual usada en estos dispositivos. Esta máquina virtual se denomina KVM. Para mantener esta máquina virtual lo más sencilla y

pequeña posible, se elimina esta verificación, y se realiza antes de la entrada en producción. La preverificación se realiza después de la compilación, y el resultado es una nueva clase, lista para ser puesta en producción.

Los MIDLets, son empaquetados en ficheros ".jar". Se requiere alguna información extra, para la puesta en marcha de las aplicaciones. Esta información se almacena en el fichero de "manifiesto", que va incluido en el fichero ".jar" y en un fichero descriptor, con extensión ".jad". Un fichero ".jar" típico, por tanto, se compondrá de:

- Clases del MIDLet
- Clases de soporte
- Recursos (imágenes, sonidos...)
- Manifiesto (fichero ".mf")
- Descriptor (fichero ".jad")

Un fichero ".jar" puede contener varios MIDLets. Esta colección de MIDLets, se suele llamar "MIDLet Suite". Esta unión de varios MIDLets en una distribución, permite compartir recursos (imágenes, sonidos...), y por tanto optimizar los recursos del dispositivo.

4.3.1. Descripción del MIDlet

Todos los MIDLets, deben heredar de la clase *javax.microedition.midlet.MIDlet*, contenida en el API MIDP estándar. Esta clase define varios métodos, de los cuales destacaremos los siguientes:

- startApp() – Lanza el MIDLet
- pauseApp() – Para el MIDLet
- destroyApp() – Detruye el MIDLet

4.3.2. Estados de un MIDlet

Un *MIDlet* durante su ejecución pasa por 3 estados diferentes. Como ya hemos visto 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. 4.2 nos muestra el diagrama de estados de un *MIDlet* en ejecución:

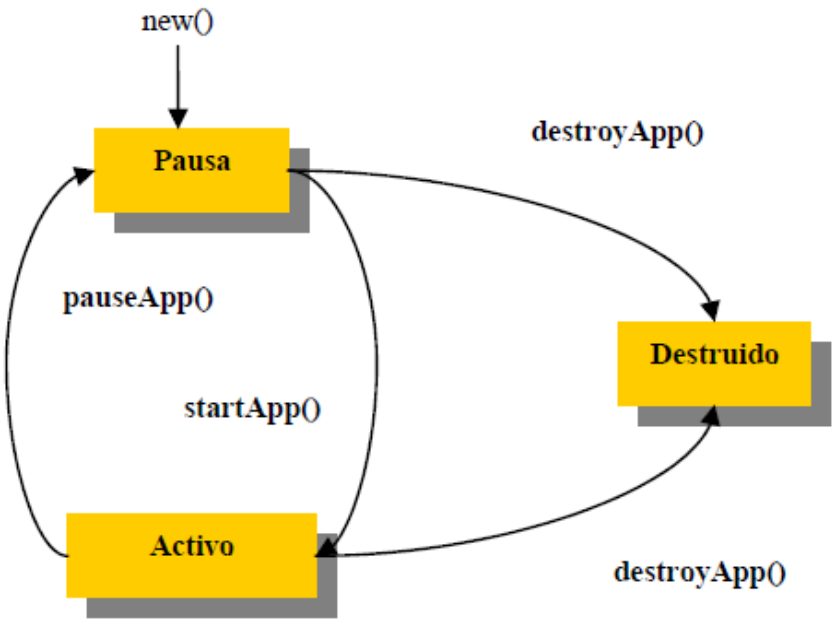


Figura. 4.2. Estados de un MIDlet.

Como vemos 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 vamos a ver por los estados que pasa un *MIDlet* durante una ejecución típica y cuáles son las acciones que realiza 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 [7].

4.4. RECORD MANAGEMENT SYSTEM

Un dispositivo móvil (al menos por ahora) no dispone de disco duro donde almacenar información permanentemente. J2ME resuelve el problema mediante el RMS (Record Management System).

El RMS (Record Management System) es un pequeño sistema de bases de datos que permite añadir información en una memoria no volátil del celular o dispositivo móvil. En una base de datos RMS, el elemento básico es el registro (record).

Un registro es la unidad de información más pequeña que puede ser almacenada. Los registros son almacenados en un recordStore que puede visualizarse como una colección de registros.

Cuando se almacena un registro en el recordStore, a éste se le asigna un identificador que identifica unívocamente al registro. Para poder utilizar RMS se debe importar el paquete javax.microedition.rms. Algunas operaciones básicas con registros son las siguientes:

- *Abrir y cerrar un recordStore*. Para abrir un recordStore se utiliza el método openRecordStore(), mientras que para cerrar el recordStore se utiliza el método RecordStore.closeRecordStore().
- *Añadir registros*. Una vez abierto el recordStore se pueden añadir registros con el método addRecord().

- *Leer registros*. Para leer registros se utiliza el método getRecord(), el cual permite acceder al registro deseado, siempre que se conozca su identificador.
- *Borrar registros*. El borrado de registros se realiza con el método deleteRecord().
- *Recorrer registros*. Se hace uso del objeto RecordEnumeration para recorrer todos los registros almacenados en la base de datos. Para crear una enumeración se utiliza el método enumerateRecords().

Una descripción mas detallada de estos métodos y del Record Management System se encuentra en [8].

4.5. WIRELESS MESSAGING API

WMA es una extensión de las especificaciones de CLDC y MIDP para el envío, la recepción y la gestión de SMS desde MIDLets. El API está compuesto de interfaces ubicadas bajo el paquete javax.wireless.messaging. Estas interfaces con sus respectivos métodos se muestran en la siguiente Tabla. Los javax.wireless.messaging.MessageConection pueden funcionar de dos modos:

- **En modo cliente**. Sólo sirve para enviar SMS a un destinatario. El modo cliente se lo especifica de la siguiente manera:

```
MessageConnection con=(MessageConnection)Connector.open ("sms://+593925421 85");
MessageConnection con=(MessageConnection)Connector.open ("sms://+
593925421 85:16500");
```

En ambos casos el SMS sería enviado al teléfono 092542185 de Ecuador (por el prefijo +593), pero en el primer caso, el SMS sería tratado por la aplicación que por defecto tiene instalada el teléfono, mientras que, en el segundo caso, el SMS sería tratado por la aplicación que esté escuchando en el puerto 16.500.

- **En modo servidor**. Sirve para recibir y tratar los SMS que son dirigidos hacia él. En este modo también se pueden enviar SMS. El modo servidor se especifica de la siguiente manera:

Interfaz	Descripción	Métodos
Message	Interfaz base, de la que derivan TextMessage y BinaryMessage	getAddress() getTimestamp() setAddress()
BinaryMessage	Subinterfaz de Message que proporciona métodos para fijar y detectar el payload binario	getPayloadData() setPayloadData()
TextMessage	Subinterfaz de Message que proporciona métodos para fijar y detectar el payload de texto	getPayloadText() setPayloadText()
MessageConnection	Interfaz a través de la cual se realiza el envío y la recepción de mensajes	newMessage() receive() send() setMessageListener() numberOfSegments()
MessageListener	Define la interfaz listener para implementar la notificación asíncrona de objetos Message	notifyIncomingMessage()

Tabla. 4.1. Tabla Clases en javax.wireless.messaging [5].

4.6. COMPONENTES APLICACIONES J2ME

Una aplicación J2ME está formada por un archivo JAR que es el que contiene a la aplicación en sí y un archivo JAD (*Java Archive Descriptor*) que contiene diversa información sobre la aplicación.

El gestor de aplicaciones o AMS (*Application Management System*) es el *software* encargado de gestionar los *MIDlets*. 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.

Cuando un *MIDlet* comienza su ejecución, está en el estado "Activo" pero, ¿qué ocurre si durante su ejecución recibimos 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 terminemos de trabajar con el *MIDlet* y salgamos de él, éste pasaría al estado de "Destruido" dónde sería eliminado de la memoria del dispositivo. Cuándo decimos que el *MIDlet* pasa al estado "Destruido" y es eliminado de memoria, nos referimos a la memoria volátil del dispositivo que es usada para la ejecución de aplicaciones. Una vez finalizada la ejecución del *MIDlet* podemos volver a invocarlo las veces que queramos ya que éste permanece en la zona de memoria persistente hasta el momento que deseemos desinstalarlo.

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 [9].

4.7. INTRODUCCIÓN A NETBEANS

NetBeans es un entorno de desarrollo integrado (IDE por sus siglas en inglés). Esto quiere decir que integra todas las herramientas que necesitamos para poder desarrollar. Originalmente la programación en Java era algo complicada porque Java cuenta con una enorme cantidad de librearías y funciones que era preciso aprenderse de memoria, viendo esto muchas compañías construyeron diferentes entornos de programación para facilitar la tarea del programador. Entre los más populares surgió Eclipse que reinó como el único y más importante IDE de Java durante varios años. Sun Microsystems desarrollo su propio IDE, que tenía la ventaja de que fue creado por las mismas personas que crearon Java años antes, este IDE fue NetBeans y después de varios años de desarrollo ha llegado a ser tan útil y poderoso como Eclipse o quizás un poco más.

Por otro lado, NetBeans Platform es un Framework con una amplia variedad de APIs que resuelven gran cantidad de problemas con los que nos encontramos a la hora de

construir una aplicación. Él es el corazón sobre el cual se construye, entre otras aplicaciones, NetBeansIDE.

NetBeans Platform hace fuerte hincapié sobre la construcción del software de forma modular, módulo sobre módulo, y es ahí precisamente donde mayor provecho podremos sacar de esta plataforma, ya que nos ofrece implementados los mecanismos de descubrimiento de nuevos módulos (y de actualizaciones de los existentes) desde repositorios remotos, resolución de dependencias, activación/desactivación de módulos en caliente, comunicación entre los mismos, etc. permitiéndonos preocuparnos por la lógica y rápidamente desplegar nuestras aplicaciones, pudiendo ir extendiendo su funcionalidad a medida que pasa el tiempo.

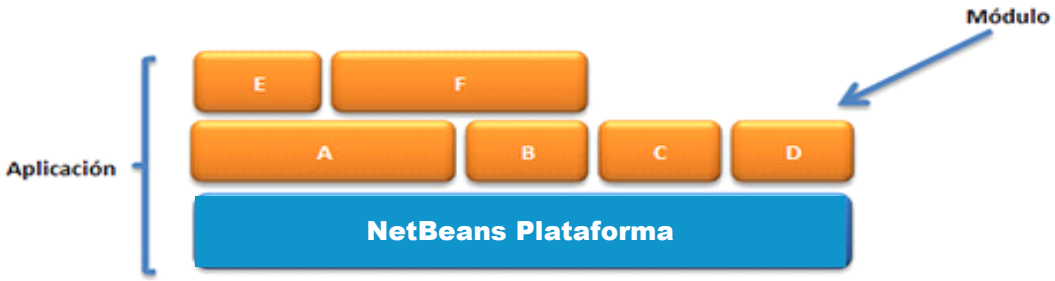


Figura. 4.3. Aplicación Modular Netbeans

Una gran ventaja de la construcción modular es que podemos crear una aplicación conformada por X cantidad de módulos diferentes, cada uno responsable de llevar a cabo determinadas responsabilidades, y según el rol de la persona que la va a utilizarla solo se carga en la aplicación los módulos que permiten cumplir con su tarea, permitiéndonos tener un abanico de aplicaciones sin tener que programar una sola línea de código adicional. Por ejemplo se indica en la Figura. 4.4:

Otras características que hacen interesante la elección de NetBeans Platform como plataforma para nuestra aplicación son las siguientes:

- Los proyectos desarrollados no dejan de ser multiplataforma, y poseen lanzadores (launchers) para cada plataforma.

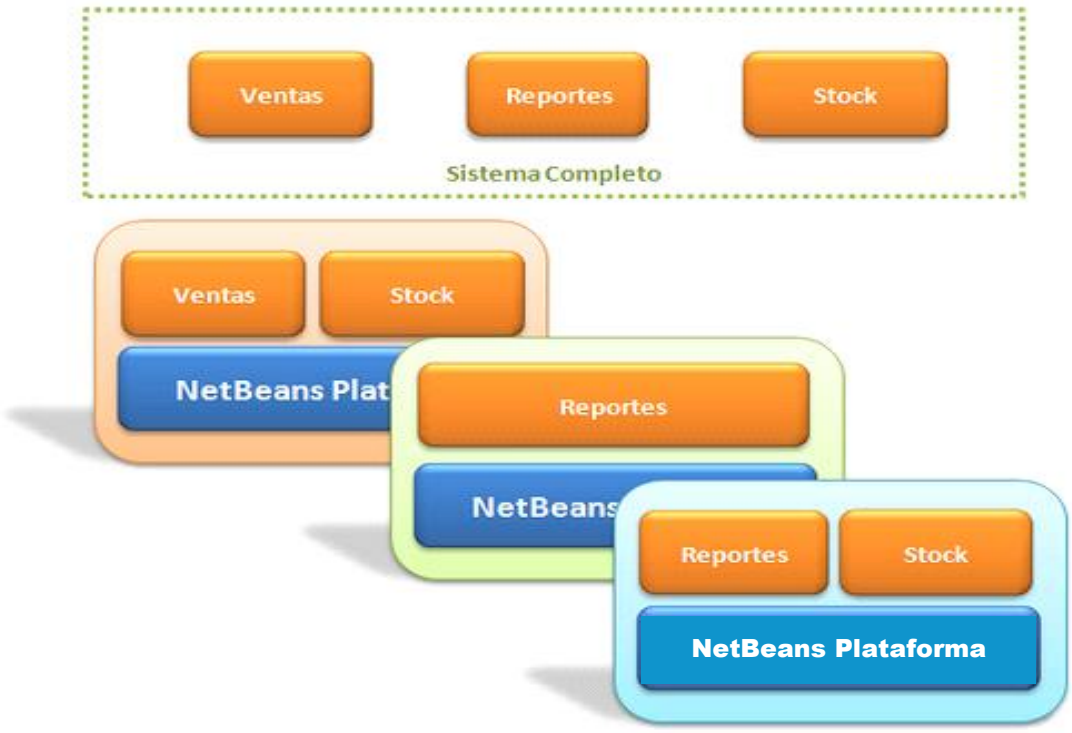


Figura. 4.4. Ejemplo de Aplicación Modular Netbeans

- Sistema de ventanas práctico para desarrollar las interfaces de usuario
- Sistema de ficheros virtual en el cual se van montan los diferentes módulos con el cual se van adaptando automáticamente los menús, barra de herramientas, menús contextuales, etc. de la aplicación
- Su licencia nos permite construir tanto aplicaciones open source como comerciales
- Compatibilidad con Java Web Start
- No es obligatorio que una aplicación deba tener interfaz de usuario grafica (GUI), ya que la plataforma permite dejar de lado la misma y seguir disfrutando del resto de los beneficio, por ejemplo la actualización de módulos desde un repositorio remoto.
- Soporte completo para desarrollar desde NetBeans IDE, por lo que no necesitaremos otra herramienta adicional para el desarrollo

4.7.1. Objetos

Objetos generados por el wizard del Netbeans:

- **Displayables**
    - **Alert**

Una alerta es una pantalla que muestra los datos al usuario y espera durante un cierto período de tiempo antes de proceder a mostrar la siguiente. Una descripción puede contener una cadena de texto y una imagen.

El uso de Alerta es informar al usuario acerca de errores y otras condiciones excepcionales.

- **Text Box**

  La clase TextBox es una pantalla que permite al usuario introducir y editar texto.

- **Form**

  Un formulario es una pantalla que contiene una mezcla arbitraria de elementos: imágenes, campos de texto de sólo lectura, campos de texto editable, editar los campos de fecha, medidores, grupos de elección, y los temas personalizados. En general, cualquier subclase de la clase puede ser la partida que figura dentro de un formulario. La aplicación se encarga de diseño, transversal, y el desplazamiento.

- **Login Screen**

  La pantalla de inicio de sesión personalizado, constituye un valioso componente de interfaz de usuario estándar con elementos tales como nombre de usuario de campo, Contraseña de campo y botón de acceso. Puede utilizar este componente personalizado para crear la interfaz de usuario para acceder a funciones de protección tales como GSM bancario

- **Splash Screen**

  Splash Screen se utilizan para mejorar el aspecto de una aplicación. Normalmente, se utiliza una pantalla de bienvenida cuando se inicia el programa, o para mostrar un logotipo o marca de información. Se ofrece a los usuarios la primera impresión de su aplicación.

- **Commands**

  Todos los componentes de comandos tienen la misma funcionalidad y comportamiento. El Comando de la clase es un concepto que engloba la información semántica de una acción. El comportamiento que se activa el comando no es encapsulado en este objeto. Esto significa que sólo el componente contiene información acerca de un "comando", y no la acción que se produce cuando el comando está activado. La acción que se produce se define en el CommandListener mostrar asociados con el. Comando objetos se presentan en la interfaz de usuario. Cómo se presentan pueden depender de la información semántica contenida en el comando. Aquí tenemos algunos de los comandos:

- Back Command
- Exit Command
- Ok Command

- **Items**
  - **Choice Group**

    Un ChoiceGroup es seleccionar un grupo de elementos destinados a ser introducidos en un formulario. El grupo se puede crear con una modalidad que requiere de una única elección que se hizo o que permite múltiples opciones. La aplicación se encarga de proporcionar la representación gráfica de estos modos, y facilitará los gráficos visualmente diferentes para los distintos modos de transporte. Por ejemplo, podría usar "botones" para el único modo de elección y "casillas" para el modo de selección múltiple.

  - **Text Field**

    Un campo de texto editable es un componente de texto que se puede poner en un formulario. Se puede dar un trozo de texto que se utiliza como valor inicial

  - **Table Item**

    El componente TableItem permite crear rápidamente tablas que constan de una o más columnas, cada uno con una cabecera y un área de datos que se repite para cada registro. El cuadro puede ser más grande que una pantalla en ambas direcciones, los usuarios pueden utilizar un cursor para moverse en todas direcciones.

  - **String Item**

    Un elemento que puede contener una cadena. Un StringItem es sólo visual el usuario no puede editar el contenido. Tanto la etiqueta y el contenido textual de un StringItem pueden ser modificados por la solicitud. La representación visual de la etiqueta puede diferir de la de los contenidos textuales

- **Flujo**

Principalmente responsable de la aplicación lógica:

  - **If**

    "If" dirige el flujo basado en una determinada condición.

  - **Switch**

    Switch" es similar a la componente "If" pero tiene que especificar los casos mediante la asignación de "Cambiar el asunto" componentes [10].

CAPÍTULO 5

MICROCONTROLADOR Y MODEM GSM

5.1. MICROCONTROLADOR

El microcontrolador es un circuito integrado de muy alta escala de integración que contiene las partes funcionales de un computador:

- CPU (Central Processor Unit o Unidad de Procesamiento Central)
- Memorias volátiles (RAM), para datos
- Memorias no volátiles (ROM, PROM, EPROM) para escribir el programa
- Líneas de entrada y salida para comunicarse con el mundo exterior.
- Algunos periféricos (comunicación serial, temporizador, convertidor A/D, etc.)

Es decir el microcontrolador es un computador integrado en un solo chip. Integrar todos estos elementos en un solo circuito integrado ha significado desarrollar aplicaciones importantes en la industria al economizar materiales, tiempo y espacio.

5.1.1. Aplicaciones del microcontrolador

Las aplicaciones de un microcontrolador son tan inmensas que el límite es la propia imaginación del usuario. Estos microcontroladores están en el auto, en el televisor, en el teléfono, en una impresora, en un horno de microondas, en un transbordador espacial, en un juguete, etc. Algunas fuentes estiman que en una casa típica de EE.UU. se tiene alrededor de 250 microcontroladores.

Los siguientes son algunos campos en los que los microcontroladores tienen gran uso:

- En la industria del automóvil: Control de motor, alarmas, regulador del servofreno, dosificador, etc.
- En la industria de los electrodomésticos: control de calefacciones, lavadoras, cocinas eléctricas, etc.

- En informática: como controlador de periféricos. Por ejemplo para controlar impresoras, plotters, cámaras, scanners terminales, unidades de disco, teclados, comunicaciones (modems), etc.
- En la industria de imagen y sonido: tratamiento de la imagen y sonido, control de los motores de arrastre del giradiscos, magnetófono, video, etc.

En la industria, en general se utilizan en:

- Regulación: todas las familias de microcontroladores incorporan en alguna de sus versiones conversores A/D y D/A, para la regulación de la velocidad de las máquinas, de niveles, de temperatura, etc.
- Automatismos: La enorme cantidad de líneas de entrada y salidas, y su inmunidad al ruido le hacen muy valioso para el control secuencial de procesos. Por ejemplo control de máquinas, herramientas, apertura y cierre automático de puertas según condiciones, plantas empaquetadoras, aparatos de maniobra de ascensores, etc.
- Robótica: para control de los motores y captura de señales de los diferentes sensores, fabricación de controladores robóticos para sistemas automáticos, etc.

Instrumentos portátiles compactos:

- Radio paginador numérico (beeper)
- Planímetro electrónico
- Nivelímetro digital
- Identificador-probador de circuitos integrados
- Tacómetro digital
- Panel frontal de un osciloscopio
- Controlador de display LCD
- Analizador de espectros, etc.

Dispositivos autónomos:

- Fotocopiadoras
- Máquinas de escribir
- Selector, Codificador decodificador de TV

- Localizador de peces
- Teléfonos de tarjeta
- Teléfonos celulares
- Cerraduras electrónicas
- Sistemas de seguridad

Se emplea también en medicina, en aplicaciones militares, edificios inteligentes, etc.

5.1.2. Principales fabricantes

Por lo general los fabricantes de microprocesadores lo son de microcontroladores. Los fabricantes de microcontroladores son más de 50, podemos mencionar a:

- Atmel
- Motorola
- Intel
- Microchip
- NEC
- Hitachi
- Mitsibishi
- Philips
- Matsushita
- Toshiba
- AT&T
- Zilog
- Siemens
- National Semiconductor
- etc.**[11].**

5.1.3. Selección del microcontrolador

Sin duda la elección del microcontrolador dependerá de la tarea o proyecto que se tiene en mente pues los fabricantes como se mencionó anteriormente son más de 50, estos tienen muchos modelos enfocados a tareas específicas. Esta selección deberá ir de la mano

con factores económicos óptimos así como de la idea del controlador incrustado (*embedded controller*), el cual es un controlador dedicado a una sola tarea incorporado al sistema que gobierna.

Antes de seleccionar un microcontrolador es imprescindible analizar los requisitos de la aplicación:

- **Procesamiento de datos**: Cuando se desea realizar cálculos complejos en un tiempo limitado, se debe seleccionar un microcontrolador suficientemente rápido para ello.
  Por otro lado, habrá que tener en cuenta la precisión de los datos a manejar: si no es suficiente con un microcontrolador de 8 bits, puede ser necesario acudir a microcontroladores de 16 ó 32 bits.

- **Entrada/Salida**: Se debe identificar la cantidad y tipo de señales a controlar. Una vez realizado este análisis puede ser necesario añadir periféricos externos o cambiar a otro microcontrolador más adecuado a ese sistema.

- **Consumo**: algunos productos que incorporan microcontroladores están alimentados con baterías, pero al ser alimentado a través de la PC no influye el consumo.

- **Memoria**: para detectar las necesidades de memoria de una aplicación debemos saber la cantidad y el tipo de memoria necesaria para esto se debe tener una versión preliminar (pseudo-código) de la aplicación y escoger el microcontrolador apropiado. En este caso al manejar datos sumamente extensos es necesario tomar en cuenta un microcontrolador con gran memoria

- **Ancho de palabra**: el criterio de diseño debe ser seleccionar el microcontrolador de menor ancho de palabra que satisfaga los requerimientos de la aplicación. Usar un microcontrolador de 4 bits supondrá reducir los costos, mientras que uno de 8 bits puede ser el más adecuado si el ancho de los datos es de un byte. Los microcontroladores de 16 y 32 bits, debido a su elevado costo, deben reservarse para aplicaciones que requieran altas prestaciones (Entrada/Salida grande o espacio de direccionamiento muy elevado).

Hay que tomar en cuenta que se necesita un microcontrolador que soporte comunicación serial adicional para la comunicación con la PC y con el otro microcontrolador.

5.2. MICROCONTROLADOR ATMEL AVR

ATMEL fábrica los microcontroladores de la familia AVR, esta nueva tecnología proporciona todos los beneficios habituales de arquitectura RISC y memoria flash reprogramable eléctricamente. La característica que los identifica a estos microcontroladores de ATMEL, es la memoria flash y eeprom que incorpora. AVR compite con varias familias de microcontroladores bien establecidas en el mercado, tales como 8051 de Intel, 68HC11 de Motorola y la familia PIC de Microchip. La firma también produce y vende varios subproductos de la popular familia 8051 con la diferencia de que están basados en la memoria flash.

El diseño AVR de ATMEL difiere de los demás microcontroladores de 8 bits por tener mayor cantidad de registros (32) y un conjunto ortogonal de instrucciones. AVR es mucho más moderna que su competencia. Por ejemplo, los 8051, 6805 y los PIC, se los arreglan con un único acumulador, los 658HC11 y 68HC12 tienen simplemente 2. Esto hace que la arquitectura AVR sea más fácil de programar a nivel de lenguaje ensamblador y que sea fácil de optimizar con un compilador. El gran conjunto de registros disminuye la dependencia respecto a la memoria, lo cual mejora la velocidad y disminuye las necesidades de almacenamiento de datos. Además casi todas las instrucciones se ejecutan en 1 ó 2 ciclos de reloj versus 5-10 ciclos de reloj para los chips 8051, 6805, 68HC11 y PIC.

Adicionalmente, ATMEL también proporciona en línea el entorno software (AVR estudio) que permite editar, ensamblar y simular el código fuente, (la explicación del Avr Studio 4.0, se explicará más adelante). Una vez ensamblado y depurado el código fuente del programa, se transferirá el código máquina a la memoria flash del microcontrolador para esto se debe disponer de otro entorno de desarrollo para programar en forma serial o paralelo la memoria flash.

Las familias AVR rápidamente han crecido en el mercado y se dispone de las siguientes categorías:

- **TINY AVR**: son microcontroladores de propósito general con memoria flash hasta 2 kbytes y 128 bytes de memorias SRAM y EEPROM.

  AVR: Microcontroladores de propósito general con 8 kbytes de memoria flash y 512 bytes de memoria SRAM y EEPROM.

- **Mega AVR** Memoria flash hasta 256 kbytes, 4 kbytes de memoria EEPROM y SRAM Los tipos de encapsulado del microcontrolador del ATmega presenta desde 28 pines hasta 100 pines en la forma de DIP, TQFP y MLF y su voltaje de alimentación está en el rango de 1.8 a 5.5 voltios. Se presenta en la Figura. 5.3 sus características principales

Product	Flash (KB)	EEPROM (Bytes)	RAM (Bytes)	I/O	SPI	USART	USI	TWI	PWM	On-Chip Debug		10-bit ADC	LCD
										JTAG	debugWire		
megaAVR													
ATmega 48	4	256	512	23	1	1	–	1	5	–	Y	8	–
ATmega8	8	512	1K	23	1	1	–	1	3	–	–	8	–
ATmega88	8	512	1K	23	1	1	–	1	5	–	Y	8	–
ATmega8515	8	512	512	35	1	1	–	–	3	–	–	–	–
ATmega8535	8	512	512	32	1	1	–	1	4	–	–	8	–
ATmega16	16	512	1K	32	1	1	–	1	4	Y	–	8	–
ATmega162	16	512	1K	35	1	2	–	–	6	Y	–	–	–
ATmega168	16	512	1K	23	1	1	–	1	5	–	Y	8	–
ATmega32	32	1K	2K	32	1	1	–	1	4	Y	–	8	–
ATmega64	64	2K	4K	53	1	2	–	1	8	Y	–	8	–
ATmega128	128	4K	4K	53	1	2	–	1	8	Y	–	8	–
ATmega256	256	4K	8K	53	1	2	–	1	16	–	Y	8	–
LCD AVR													
ATmega169	16	512	1K	53	1	1	Y	–	4	Y	–	8	Y
ATmega329	32	1K	2K	53	1	1	Y	–	4	Y	–	8	Y

Figura. 5.1. Características del microcontrolador ATmega AVR

5.2.1. Microcontrolador ATmega16

Resumiendo las características principales se puede mencionar, que el ATmega16 es un microcontrolador de 8K bytes de flash programable con capacidad de lectura y escritura, con 512 bytes de EEPROM, 1kbyte de SRAM, también cuenta con 32 I/O de propósito general, un USART Serial programable, interface SPI master/esclavo.

Generador del reloj. Usualmente un cristal de cuarzo de frecuencias que genera una señal oscilatoria entre 1 y 40 MHz, o también resonadores o circuitos RC. De alto rendimiento y baja potencia AVR ® de 8 bits Microcontrolador

- **Arquitectura RISC avanzada**
  - 131 Instrucciones Potentes.
  - La mayoría Ejecuta en solo ciclo de reloj.
  - 32 x 8 Registros de propósito general.
  - Rendimiento hasta 16 MIPS a 16 MHz.
- **Programa no volátil de datos y recuerdos**
  - De 16K Bytes En Sistema de auto-programable Flash.
  - Resistencia: 10.000 **Write/Erase** Ciclos.
    - 512 Bytes EEPROM.
  - Resistencia: 100.000 **Write/Erase** Ciclos.
    - 1K Byte SRAM Interior
  - Programación de bloqueo de Software Security
- **JTAG (IEEE Std. Compatible 1149,1) Interfaz**
  - Capacidades de exploración de Fronteras Según la Norma JTAG.
  - -Amplia El chip de depuración de Apoyo.
  - La programación de Flash, EEPROM, fusibles, y de bloqueo de bits a través de la interfaz JTAG.
- **Características Periféricas**
  - Dos de 8-bits temporizador / Contadores con Prescalers separado y comparar los modos.
  - Un temporizador de 16-bit / Contador con Prescaler separado, comparar el modo de captura y Modo.
  - Cuatro Canales PWM.
  - 8-channel, 10-bits ADC

    - Byte orientadas a dos cables de interfaz serial.
  - Programable serie USART.
  - Maestro / Esclavo de interfaz en serie SPI.
  - Temporizador programable de vigilancia por separado con el chip oscilador.
- **Características especiales Microcontrolador.**
  - Power-On Reset y programable de detección de Brown.

- Oscilador RC Calibrada Interna.
- Interrupción de Exteriores y del Interior Fuentes.
- Seis modos: Inactivo, ADC Reducción de ruido, de ahorro de energía, reducción de potencia, modo de espera y extensión de espera

- **I / O y encapsulado**
  - 32 programable líneas de E / S
  - 40-pines PDIP, 44-plomo TQFP, y 44-almohadilla FML
- **Los voltajes de operación**
  - 2,7 - 5.5V para ATmega16L
  - 4.5 - 5.5V para ATmega16
- **Grados de Velocidad**
  - 0 - 8 MHz para ATmega16L
  - 0 - 16 MHz para ATmega16
- **Consumo de energía a 1 MHz, 3V, y 25 ° C**
  - Activa: 1,1 mA
  - El modo de espera: 0,35 mA
  - Power-Down Mode: <1 µA

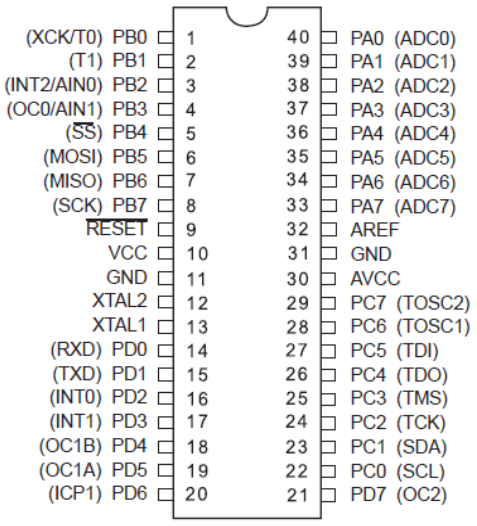


Figura. 5.1. Distribución de pines ATmega16

5.2.2. ATtiny2313

El ATtiny2313 es un microcontrolador Atmel de características muy interesantes; barato, con la versión de 20-pines encapsulado DIP y oscilador interno. Estas

características permiten a los conductores construir muy compacto, sin grandes dificultades en el tiempo para montar en placa. El ATtiny2313 tiene las siguientes características:

- **General:**
  - 32 Registros de 8 bits de propósito general
  - Hasta 20 MIPS a 20 MHz de reloj
  - 2K En Bytes de Flash-Sistema de auto programable (10.000 ciclos de escritura / borrado)
  - 128 Bytes En-Sistema programable EEPROM (10.000 ciclos de escritura / borrado) 128 Bytes de SRAM interna
- **Características de los periféricos**
  - Un temporizador / Contador con 8 bits Prescaler
  - Un temporizador / Contador de 16-bit con Prescaler
  - Cuatro canales de PWM
  - Una analogía comparativa en propia chipOn chip comparador analógico
  - Temporizador programable con oscilador de vigilancia en el chip USI - Interfaz
  - Serial Universal Full Duplex USART
- **Características del chip especial**
  - Chip DebugWIRE Depuración
  - En el sistema programable a través de la SPI Puerto Interno y Externo
  - Interrupciones de baja potencia de inactividad, el poder y el modo de Círculo de poder en Restablecer
  - Circuito de detección programable Brown-out Oscilador interno Calibrable.
- **I / O y encapsulamiento**
  - 18 pines de E / S programables
  - 20 pines PDIP, SOIC, 20 pines, QFN / FML, 20 pastillas
- **Tensiones de funcionamiento**
  - 1,8 - 5.5V (ATtiny2313V)
  - 2,7 - 5.5V (ATtiny2313)
- **Frecuencia de reloj**
  - ATtiny2313V: 0 - 4 MHz@ 1.8 - 5.5V, 0-10 MHz@ 2.7 - 5.5V
  - ATtiny2313: 0-10 MHz@ 2.7 - 5.5V, 0-20 MHz@ 4.5 - 5.5V

Para desarrollar software para el uso de ATmega puede ser realizado con BASIC, ensamblador, C y el uso de entornos diversos proveedores especializados. Principalmente el uso de herramientas libres y de código abierto en este sector y tener el AVR Studio 4 (sólo para Windows), una excelente herramienta para el desarrollo de Atmel de ensamblador y simulación de funcionamiento de microcontroladores AVR. Esta herramienta también apoya la integración con los países del CCG para ayudar a la depuración.

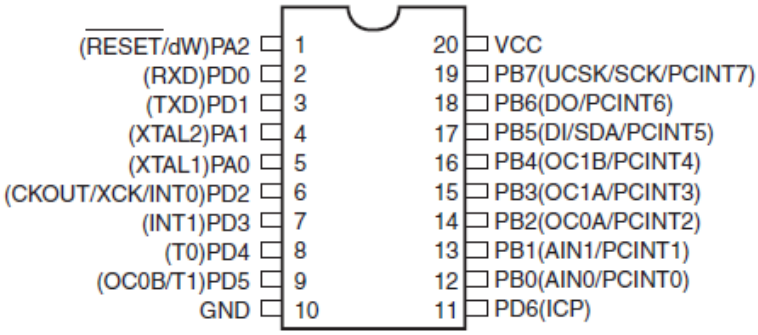


Figura. 5.2. Distribución de pines ATtiny2313.

5.3. MODEM GSM

Con el fin de cumplir con el objetivo de reutilizar equipos terminales que se encuentren en desuso y reducir el costo del proyecto, se ha utilizado para la implementación del mismo, el modem GSM embebido en un teléfono de la marca Nokia, modelo 5070 que se presenta en la Figura. 5.4. Algunas características de este modem son: Es un modem embebido que puede ser accedido mediante el cable de conexión DKU5 el cual va conectado al puerto de teléfono celular (Figura. 5.5) mediante los pines 6, 7,8 descritos en la tabla .5.1. Este modem presenta: Velocidades de trasmisión de datos hasta 9600 bps y soporta comandos AT. Admite modo texto y modo PDU.

Figura. 5.3. Nokia 5070.

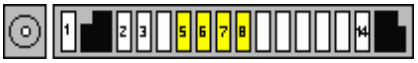


Figura. 5.4. Puerto del teléfono Nokia.

# Pin	Nombre Pin	Descripción
1	Vin	Charger input
2	GND	Charger ground
3	ACI	Accessory Control Interface (short with pin 2 for handsfree recognition)
4	V Out	Connected to pin 3 in DKU-2 usb data cable
5	USB Vbus	Also act as USB power detection? Should be connected to USB pin 1 in usb data cable.
6	FBus Rx/USB D+	USB exists only in some models*. Should be connected to USB pin 3 in usb data cable.
7	FBus Tx/USB D-	USB exists only in some models*. Should be connected to USB pin 2 in usb data cable.
8	GND	Data GND
9	X Mic-	Audio in - Ext. Mic input negativo
10	X Mic+	Audio in - Ext. Mic input positive
11	HS Ear L-	Audio out - Ext. Audio out - left, negativo
12	HS Ear L+	Audio out - Ext. Audio out - left, positivo
13	HS Ear R-	Audio out - Ext. audio out - right, negativo
14	HS Ear R+	Audio out - Ext. audio out - right, positivo. Pins 10-14 may be used for antenna connection.

Tabla. 5.1. Descripción de pines del puerto del teléfono Nokia

CAPÍTULO 6

PROTOTIPO RECEPTOR

6.1. FUNCIONAMIENTO DEL PROTOTIPO RECEPTOR

El funcionamiento es receptar, filtrar y procesar los mensajes SMS que serán enviados por la aplicación HMI del dispositivo móvil, los cuales serán enviados a la HMI de la PC.

6.2. DESCRIPCIÓN

El prototipo receptor esta dividido el tres bloques principales Bloque de Acceso a GSM, Bloque Central, Bloque de Comunicación con la PC como se pueden observar en la Figura. 6.1 las cuales se describe a continuación.

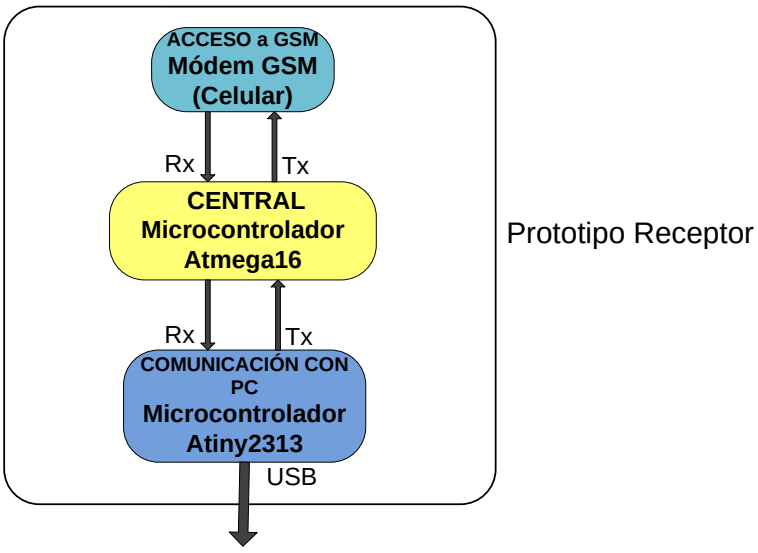


Figura. 6.1. Diagrama de bloques del Prototipo Receptor

6.2.1. Bloque Central

El prototipo receptor esta gobernado por el microcontrolador ATMEGA16 el cual se encarga de recibir los mensajes del Modem GSM, procesarlos y enviarlos al

microcontrolador ATtiny2313,para enviarlos a la PC mediante USB. De igual manera recibe el mensaje de la PC para poderlo enviar a través del Modem GSM.

En la Figura. 6.2 se muestra las componentes de entrada y salida del bloque central. El microcontrolador ATmega16 genera las instrucciones para el manejo del Modem GSM por medio del envío de comandos AT, los cuales se explicaron el CAPITULO 2.

Además, este bloque consta de LEDs INDICADORES que muestran el estado en el que se encuentra el Prototipo como se puede observar en la Figura 6.5. También en esta figura se presenta el Reset del Prototipo junto con los pines donde se conectan los otros componentes.

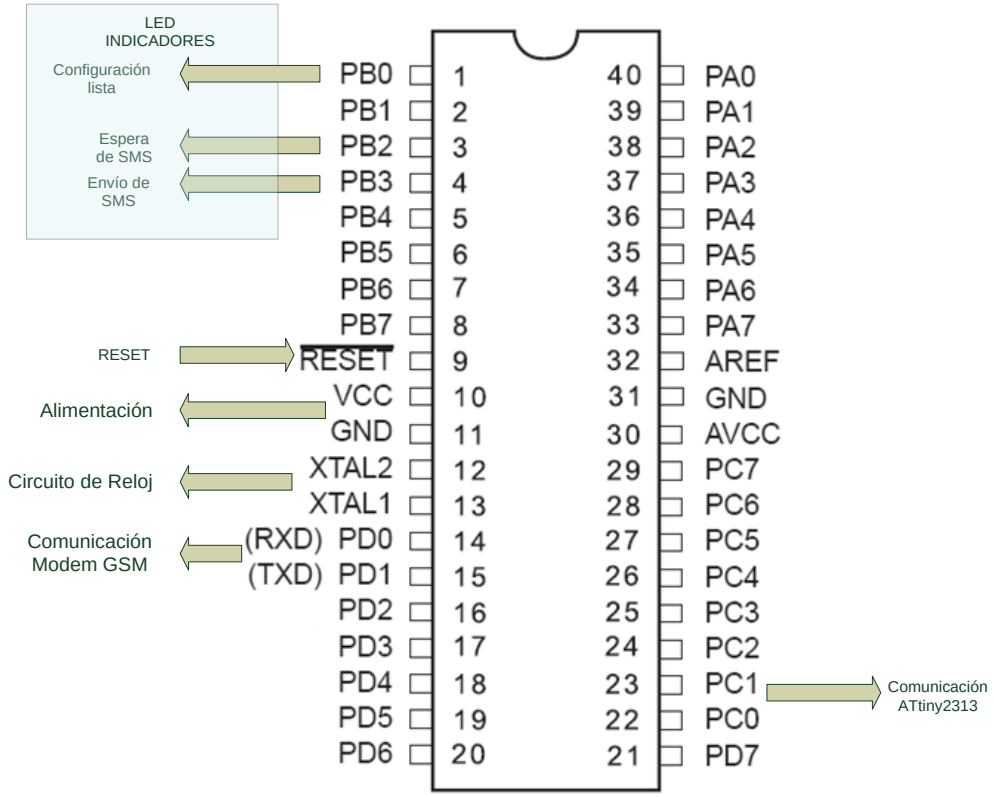


Figura. 6.2. Diagrama Bloque Central

6.2.2. Bloque de acceso a la red GSM

El bloque de acceso a la red GSM está conformado por un modem GSM de un celular en desuso. En este caso se ha utilizado un teléfono de la marca Nokia que se

comunica con el microcontrolador mediante comandos AT para poder manipular el servicio de mensajes cortos (SMS) a través del cable DKU-5.

En la Figura. 6.3 se muestra la conexión entre el modem GSM y el microcontrolador. Es importante mencionar que los pines de transmisión y recepción del microcontrolador trabajan con niveles TTL de 0-0,8 V para el estado 0 lógico, y 2-5 V para el estado 1 lógico; mientras que el modem GSM del teléfono celular trabaja con niveles TTL de 0 V (estado 0 lógico) y 3,3 V (estado 1 lógico). Por esta razón, fue necesario añadir un circuito regulador para obtener el voltaje deseado en el pin Rx y así, evitar inconvenientes de incompatibilidad de niveles de voltaje. En el otro sentido de la comunicación serial no se presenta este inconveniente de incompatibilidad.

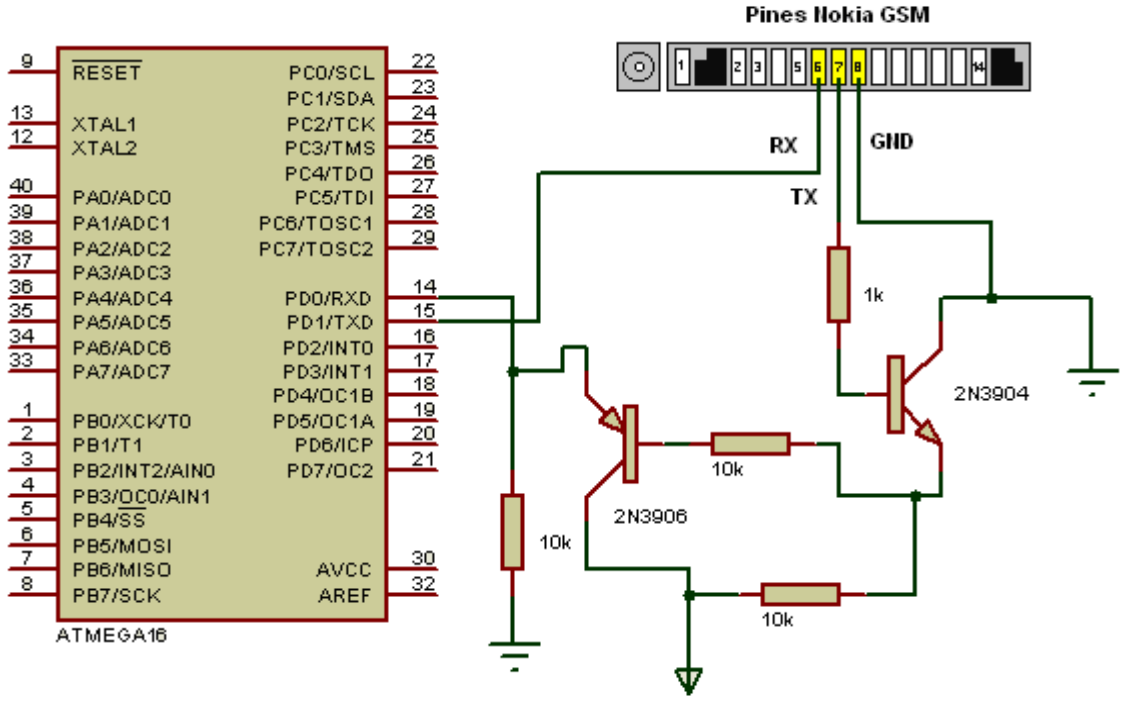


Figura. 6.1. Conexión entre el Modem GSM y el microcontrolador ATmega16.

6.2.3. Bloque de comunicación con la PC

El Bloque de comunicación con la PC está conformado por el microcontrolador el cual mediante comunicación USB se conecta a la PC de la que recibe la alimentación para el Sistema.

Para la comunicación con la PC se tomo en cuenta el auge que ha tenido la comunicación USB y la perdida del conector DB9 de la comunicación serial en las actuales computadoras; por dicha disponibilidad en todas las computadoras se eligió que la comunicaron debe ser mediante USB, de igual manera otra de las ventajas es la alimentación que brinda al circuito; por lo que fue necesario colocar un microcontrolador ATtiny2313 puesto que nos permite la comunicación USB con la PC. Como se indica el la Figura. 6.4

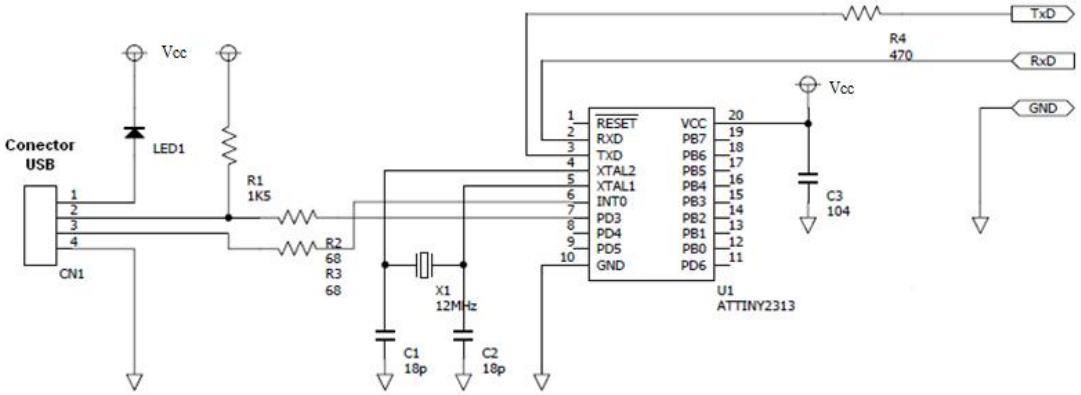


Figura. 6.2. Circuito para Comunicación con la PC.

6.2.4. Circuito implementado

En la Figura. 6.5 se puede observar el circuito esquemático en el cual esta definido el bloque central (Microcontrolador ATmega16, Led indicadores, Reset), comunicación con la PC, comunicación con el Modem GSM y como se encuentran conectados entre sí.

En la Figura. 6.6 y Figura. 6.7 se muestra tanto la vista superior como las pistas de la placa de circuito impreso y en las Figura. 6.8 y Figura. 6.9 se puede observar el circuito implementado.

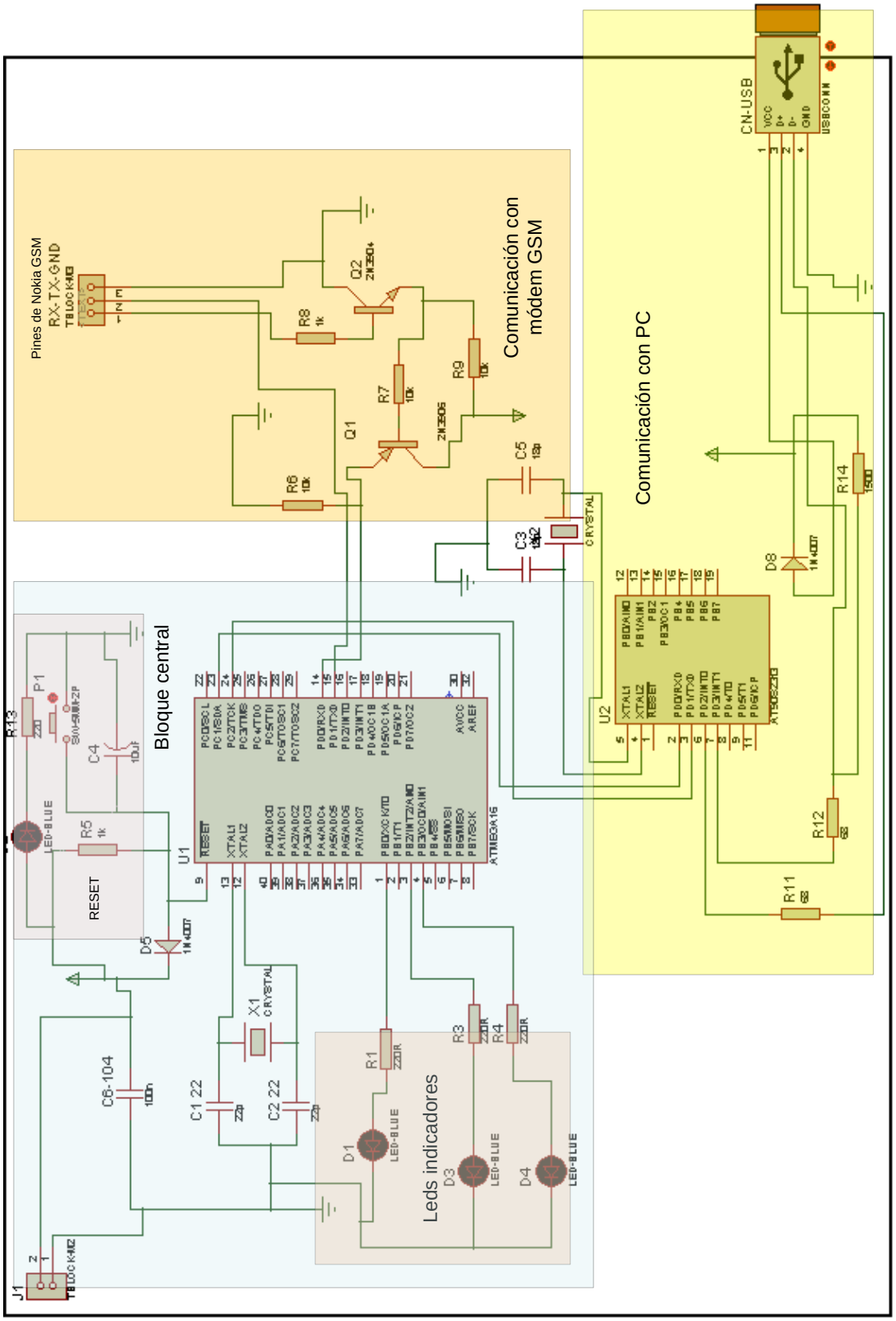


Figura. 6.3. Circuito esquemático del Prototipo receptor.

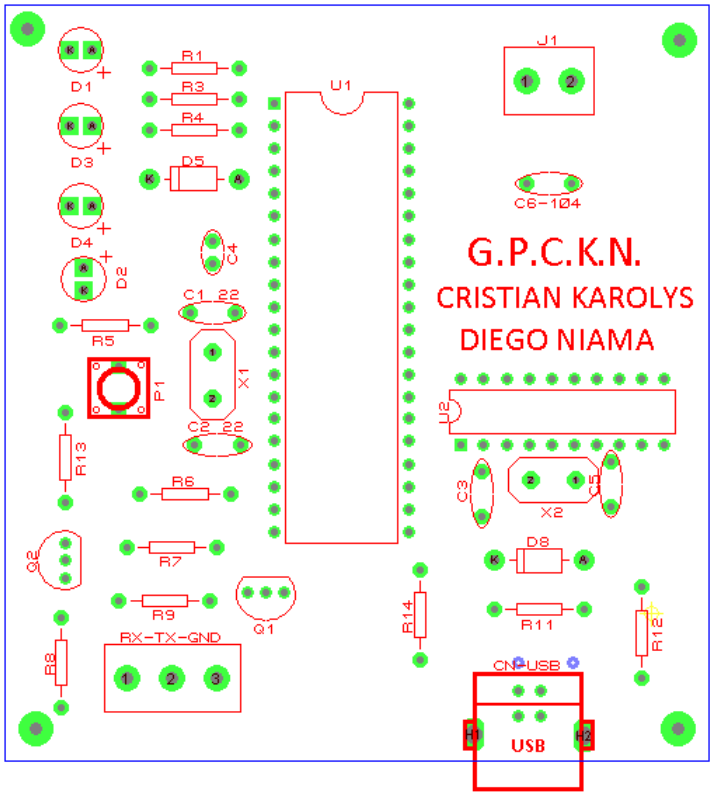


Figura. 6.4. Vista superior de la placa de circuito impreso.

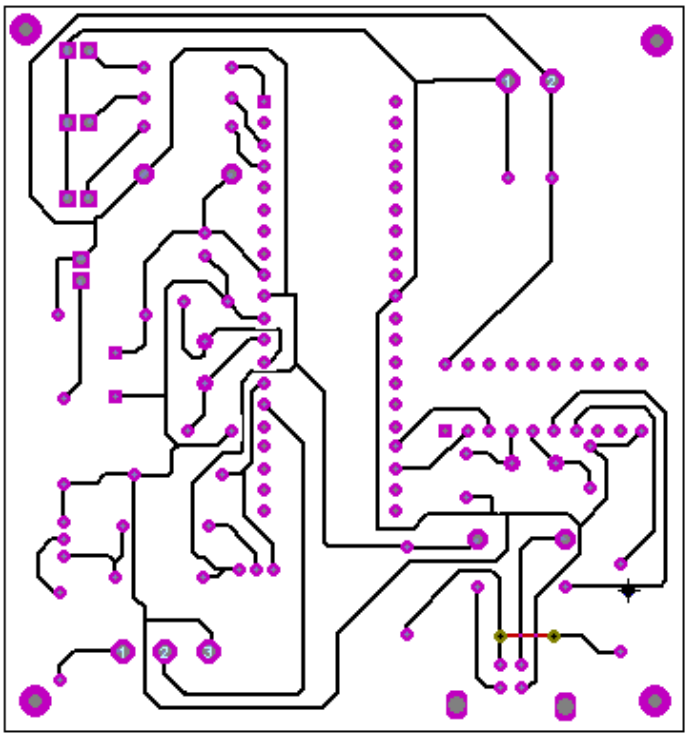


Figura. 6.5. Pistas de la placa de circuito impreso.

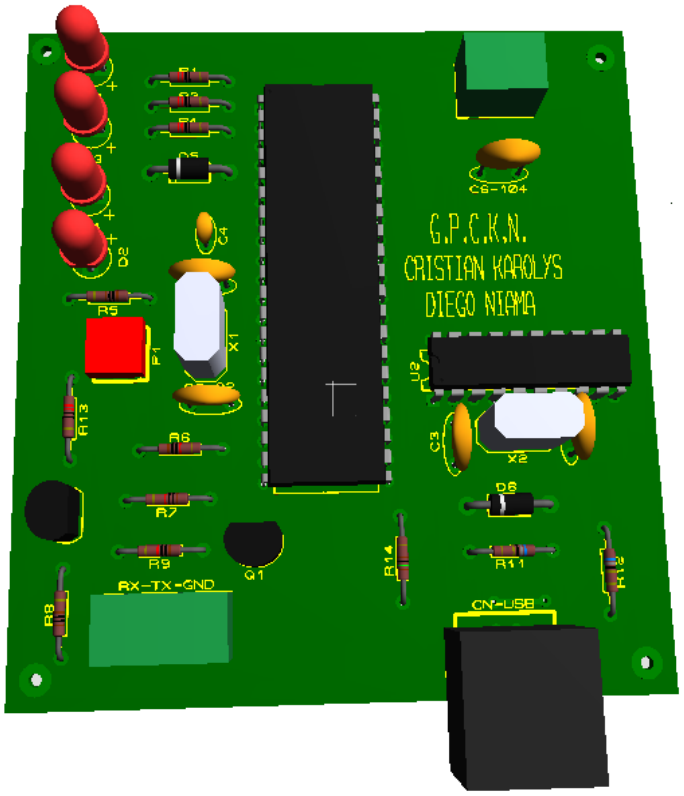


Figura. 6.6. Circuito ha ser implementado en 3D.

Figura. 6.7. Circuito implementado.

6.3. PROGRAMA DEL MICROCONTROLADOR

En este acápite se encuentra detallado el programa que fue realizado y grabado en el microcontrolador ATmega16.

El programa que controla el microcontrolador, se realizo en el compilador BASCOM-AVR.

El proceso lógico de programación del microcontrolador ATmega 16, se muestra en la Figura. 6.10, en ésta se describe el diagrama de flujo de programación utilizado en el microcontrolador.

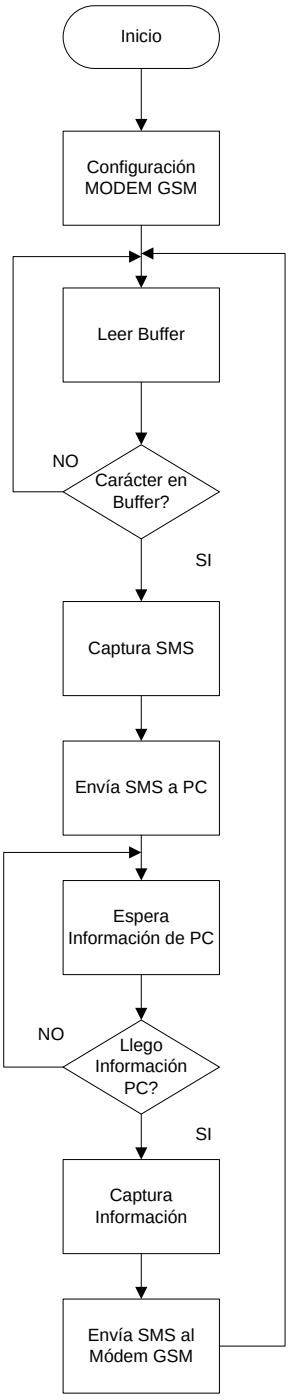


Figura. 6.8. Diagrama de flujo del microcontrolador ATmega16

6.3.1. Configuración Modem GSM

La parte principal del programa es establecer la comunicación con el Modem GSM el cual se realiza mediante comandos AT que se describe a continuación.

Inicia la Comunicación

```
Print "AT" + Chr(13) + Chr(10)
```

Apaga el echo

```
Print "ATE0" + Chr(13) + Chr(10)
```

Configura el Modem GSM como modo texto a los mensajes SMS

```
Print "AT+CMGF=1" + Chr(13) + Chr(10)
```

Habilita el paso directo del SMS al microcontrolador sin que el Modem GSM reciba en el buzón de entrada

```
Print "AT+CNMI=1,2,0,0,0" + Chr(13) + Chr(10)
```

6.3.2. Sistema de Envió y recepción de Mensajes SMS

- **Recepción del SMS**

  Como ya se sabe el Modem GSM inicialmente fue configurado para que al ingresar un mensaje SMS este no sea procesado por el Modem, sino este envíe por el puerto serial hacia el microcontrolador ATMEGA16.

  A continuación se describe el código de la captura del mensaje SMS.

```
Car_rx1 = Ischarwaiting()
```

  Para detectar un carácter en el Buffer del microcontrolador este Buffer lee continuamente, Ischarwaiting() nos devuelve el valor de 1si el carácter fue detectado.

  Al momento de detectar caracteres en el Buffer, el microcontrolador captura uno a uno los caracteres hasta una longitud de 52, que es el tamaño de la información de cabecera del mensaje, la cual esta formada por el número del remitente, la hora y fecha.

```
For I = 1 To 52
        Car_rx = Waitkey()
        Mensaje_cel = Mensaje_cel + Car_rx
Next
```

A continuación de la información de cabecera se encuentra el carácter de inicio de mensaje el cual esta definido por "^", a partir de este carácter se comienza a capturar el mensaje SMS hasta encontrar en carácter "~" en cual nos indica el fin del mensaje. De esta manera tenemos la información completa para poder procesarla. Estos caracteres permiten filtrar los mensajes no deseados ya que al no contener los caracteres de inicio y fin del mensaje el microcontrolador los desecha impidiendo que produzca errores o fallas en el sistema.

Al capturar "^" comienza a capturar el mensaje y Captura del mensaje hasta el carácter de fin de mensaje ~

```
If Car_rx = "^" Then
While Car_rx <> "~"
    Car_rx = Waitkey()
    Mensaje_cel = Mensaje_cel + Car_rx
Wend
```

- **Envío del mensaje del microcontrolador a la PC**

Al tener la información completa se procede a enviarla a la PC. El cual al no existir handshaking en la comunicación entre la computadora y el microcontrolador se envía carácter a carácter

```
For I = 1 To Longitud_mensaje
Mensaje_compu = Mid(mensaje_cel, I, 1)
Print #1, Mensaje_compu
Next
```

- **Recepción de la respuesta de la PC**

El microcontrolador al enviar la información a la PC se queda en espera de la respuesta de la misma ya que al programar un puerto serial auxiliar este no consta de la función Iscarwaiting() la cual nos indica la presencia de información.

Para poder detener la captura del mensaje se ha colocado un carácter de fin de mensaje el cual se define por "$", así de esta manera se ha logrado capturar el mensaje completo.

```
While Car_compu <> "$"
  Car_compu = Waitkey(#2)
  Mensaje_compu = Mensaje_compu + Car_compu
Wend
```

- **Envío del SMS a la aplicación J2ME**

Para enviar el mensaje SMS se necesita el tamaño del mensaje en formato PDU como ya fue explicado en el CAPITULO 3,este tamaño se encuentra en los tres primeros caracteres del mensaje recibido desde la PC

```
Tam_pdu = Mid(mensaje_compu , 1 , 3)
```

Posterior a al tamaño del mensaje se encuentra el mensaje en formato PDU

```
Mensaje_compu = Mid(mensaje_compu , 4 , Longitud_mensaje)
```

Para enviar la información es necesaria configurar el Modem GSM en modo PDU

```
Print "AT+CMGF=0" + Chr(13) + Chr(10)
```

Y finalmente se envía el mensaje como se indica a continuación

```
Print "AT+CMGS=" + Tam_pdu + Chr(13) + Chr(10)
Print Mensaje_compu ; Chr(26)
 Print Chr(13)
 Print Chr(10)
```

Para que el ciclo se inicie es necesario configurar el mensaje en modo texto ya que el mensaje que se recibe desde la aplicación J2ME viene con este formato

```
Print "AT+CMGF=1" + Chr(13) + Chr(10)
```

CAPÍTULO 7

DISEÑO DE LA INTERFAZ HMI DE LA PC

7.1. DISEÑO DE LA APLICACIÓN HMI DE LA PC

La aplicación HMI de la PC, que maneja las tablas de la base de datos de la distribuidora y que genera los mensajes SMS de respuesta al dispositivo móvil, se desarrollo en Visual Basic gracias a la cantidad de controles estándar y no estándar que posee está herramienta. Además, de ser un lenguaje de programación que posee gran versatilidad para realizar aplicaciones muy amigables para el usuario, ya que ha sido diseñado para facilitar el desarrollo de aplicaciones en un entorno gráfico (GUI-GRAPHICAL USER INTERFACE).

7.2. CARACTERÍSTICAS DE VISUAL BASIC

- *Diseñador de entorno de datos:* Es posible generar, de manera automática, conectividad entre controles y datos mediante la acción de arrastrar y colocar sobre formularios o informes.
- *Los Objetos ActiveX:* Son una nueva tecnología de acceso a datos mediante la acción de arrastrar y colocar sobre formularios o informes.
- *Asistente para formularios:* Sirve para generar de manera automática formularios que administran registros de tablas o consultas pertenecientes a una base de datos, hoja de cálculo u objeto (ADO-ACTIVE DATA OBJECT).
- *Asistente para barras de herramientas:* Es factible incluir barras de herramientas es factible incluir barra de herramientas personalizada, donde el usuario selecciona los botones que desea visualizar durante la ejecución.
- *Aplicaciones HTML:* Se combinan instrucciones de Visual Basic con código HTML para controlar los eventos que se realizan con frecuencia en una página web.

- *Ventana de Vista de datos:* Proporciona acceso a la estructura de una base de datos. Desde esta también el Diseñador tiene acceso a consultas y administrar registros de Base de Datos.

7.3. DESARROLLO Y FUNCIONAMIENTO DE LA APLICACIÓN

La Aplicación desarrollada en Visual Basic recide los mensajes SMS que llegan al modem de la distribuidora y los procesa, al obtener la información necesaria para realizar la búsqueda en las tablas de la base de datos y así determinar el mensaje SMS de respuesta al dispositivo móvil. También realiza las operaciones de impresión de factura, la misma que se explicará más adelante.

La HMI realizada para la gestión de los datos almacenados en la PC, posee un formulario por cada tabla que se tiene almacenada en la PC, en este formulario se colocan los campos de la tabla, los cuales se visualizan en el control no estándar DataGrid. Además, de algunas funciones que se explicarán más adelante.

La Aplicación se encuentra diseñada para que al usuario, se le permita realizar diferentes operaciones con los registros de la base de datos, tales como inserción, borrado y edición de registros.

El proceso lógico con el que se programo la HMI de la PC se puede observar en la Figura. 7.1, Figura. 7.2, Figura. 7.3, y en la Figura. 7.4. En estas figuras se muestra el diagrama de flujo del programa. El código de programación de la HMI se puede observar en Anexo 5.

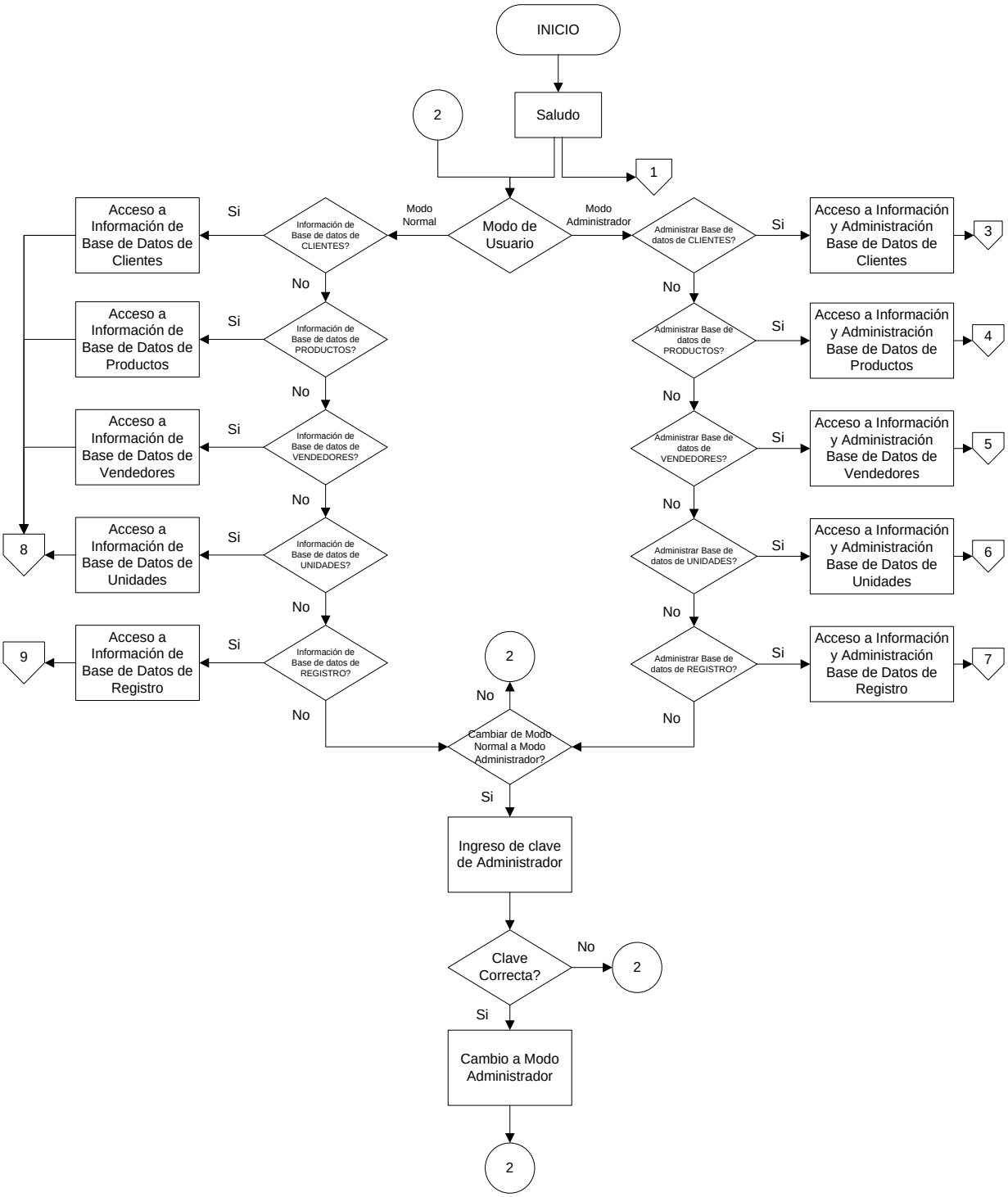


Figura. 7.1. Diagrama de Flujo de la HMI de la PC (I).

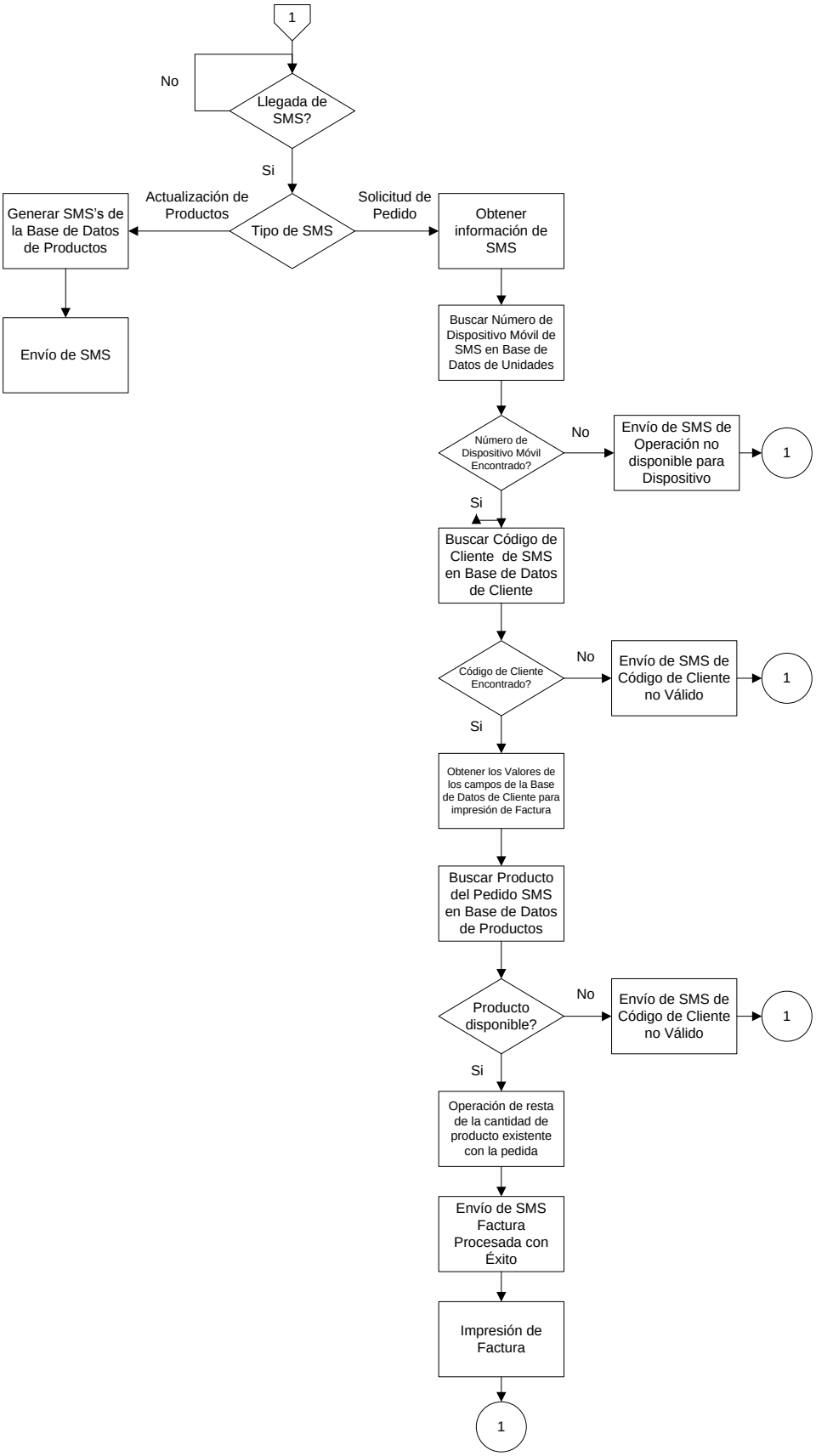


Figura. 7.2. Diagrama de Flujo de la HMI de la PC (II)

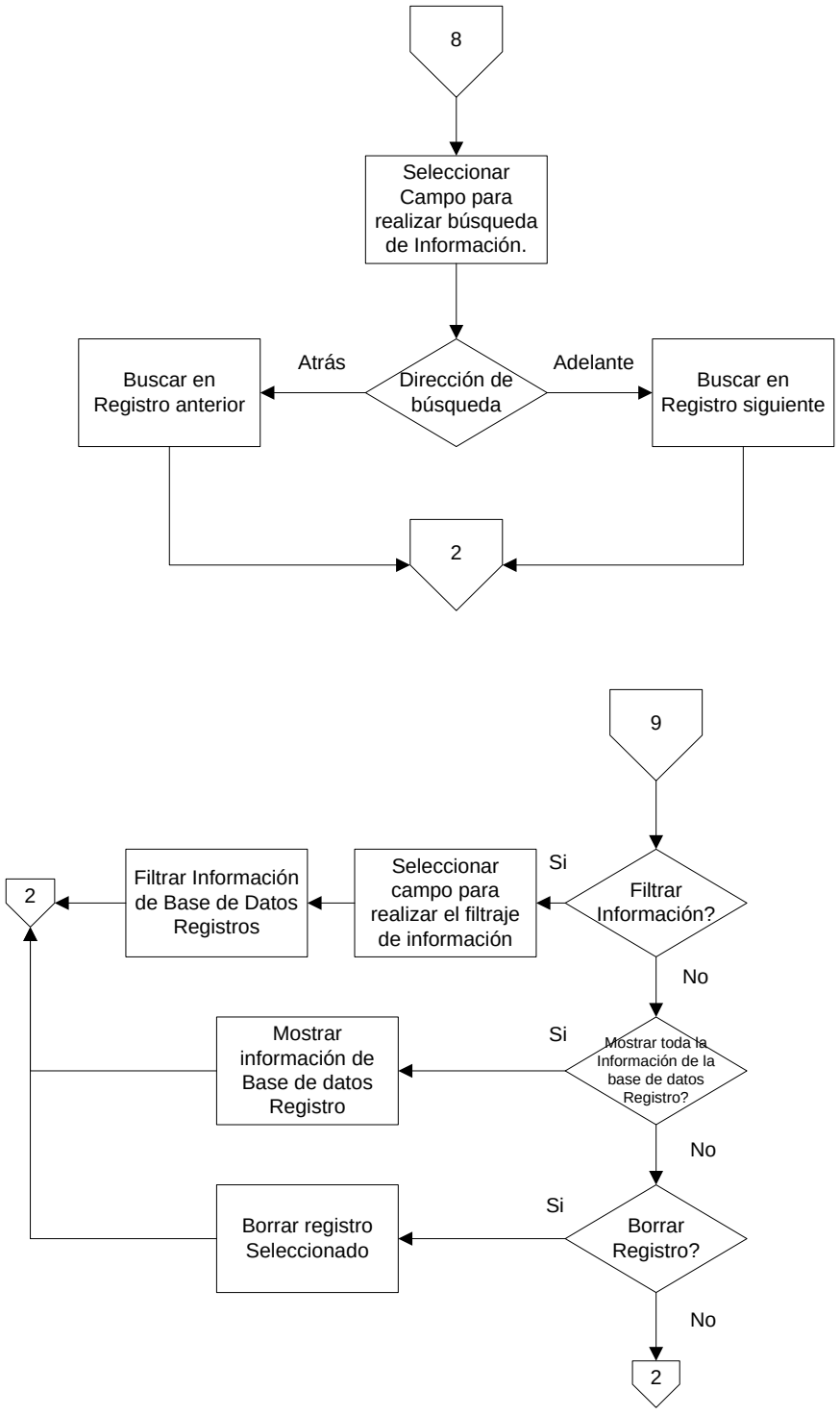


Figura. 7.3. Diagrama de Flujo de la HMI de la PC (III).

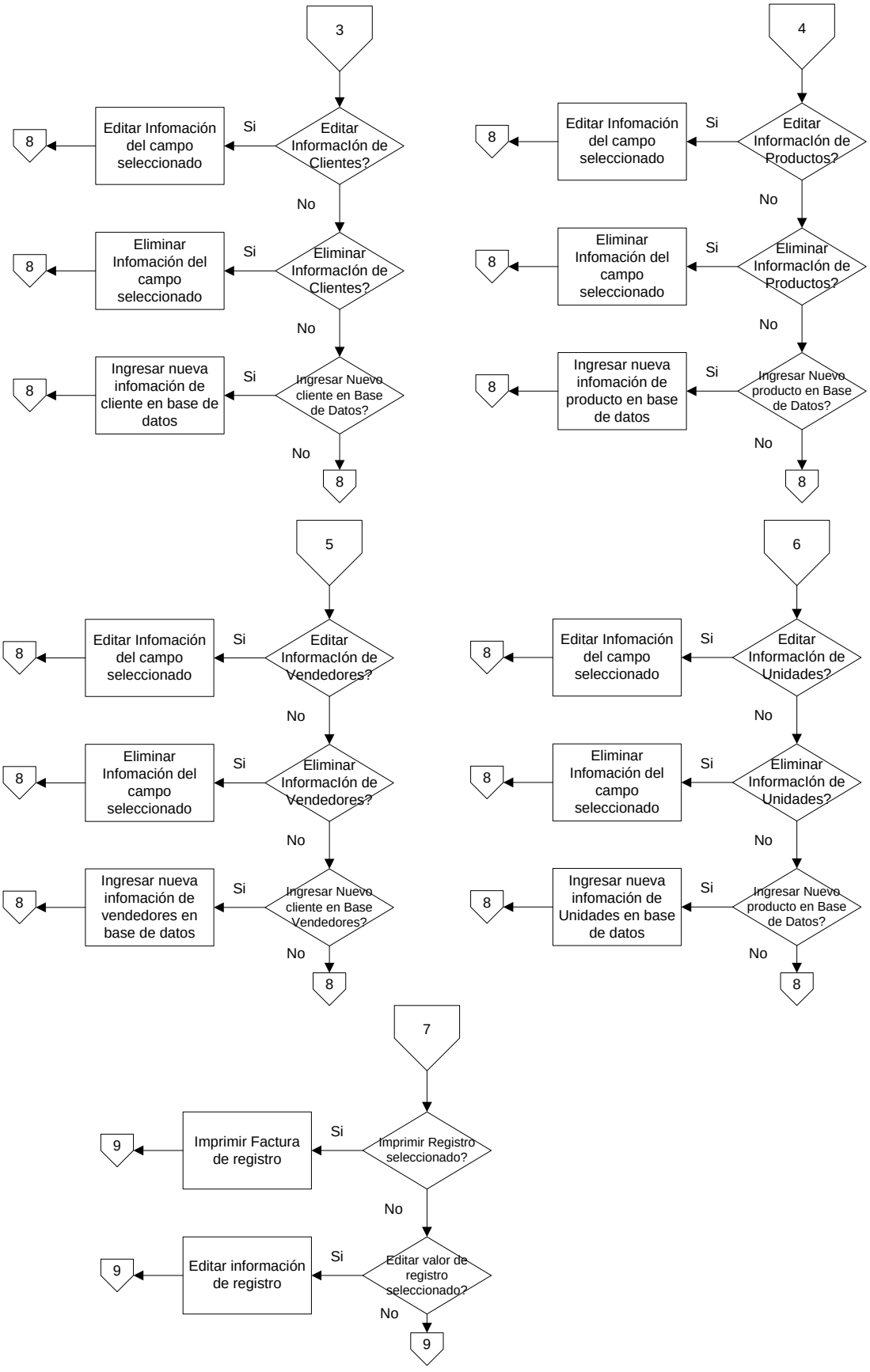


Figura. 7.4. Diagrama de Flujo de la HMI de la PC (IV).

A continuación se explicarán todos los formularios que contiene la aplicación, las secciones más relevantes del código de la aplicación HMI y las consideraciones que se tomaron en el desarrollo del mismo.

7.3.1. Formulario de Envío y Recepción de SMS

Este formulario permite al usuario observar los SMS que llegan de los dispositivos móviles y a su vez la respuesta al dispositivo móvil, que genera la aplicación dependiendo del SMS recibido. Además, permite visualizar los datos que se obtienen del SMS recibido tales como: número de celular del que envía el SMS, fecha y hora de envío de SMS, código de cliente que solicita el pedido, pedido del cliente.

En la Figura. 7.5 se puede observar todos los controles estándar y no estándar que se utilizaron en este formulario.

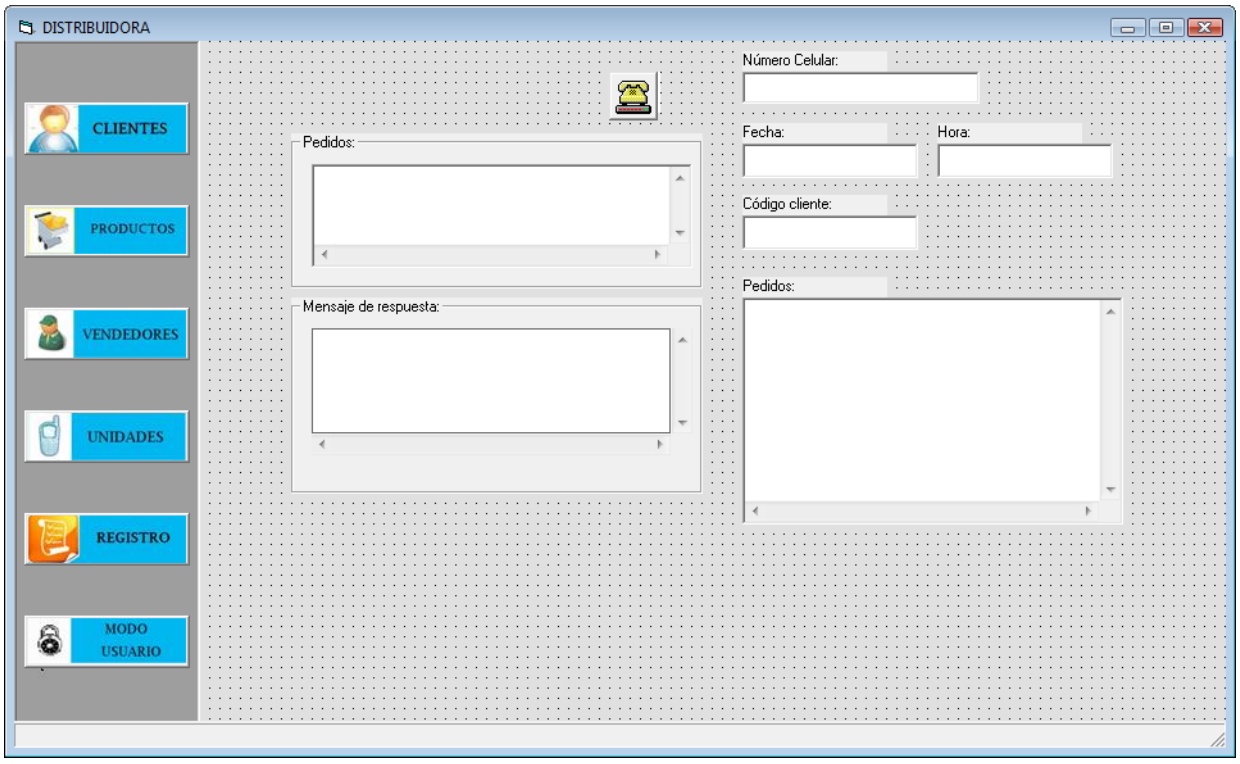


Figura. 7.5. Formulario de Envío y Recepción de SMS.

7.3.2. Formulario de Gestión de Tabla de Información de Clientes

En este formulario se puede observar la información correspondiente a la tabla de Clientes de la base de datos de la Distribuidora.

Se puede realizar una búsqueda de algún registro en la tabla, ingresando información relacionada al parámetro que se seleccione en el control ComboBox, estos parámetros pueden ser: Nombre, Ruta o Código. La búsqueda se puede realizar hacia adelante haciendo click en el CommandButton con caption ">>" y hacia atrás haciendo click en el CommandButton con caption "<<".

Además, permite realizar operaciones de inserción, edición y borrado de registros, siempre que se encuentre habilitado el modo de Administrador de la Aplicación, el mismo se explicará más adelante en la sección 7.3.7.

En la Figura. 7.6 se puede observar los controles estándar y no estándar utilizados en este formulario.

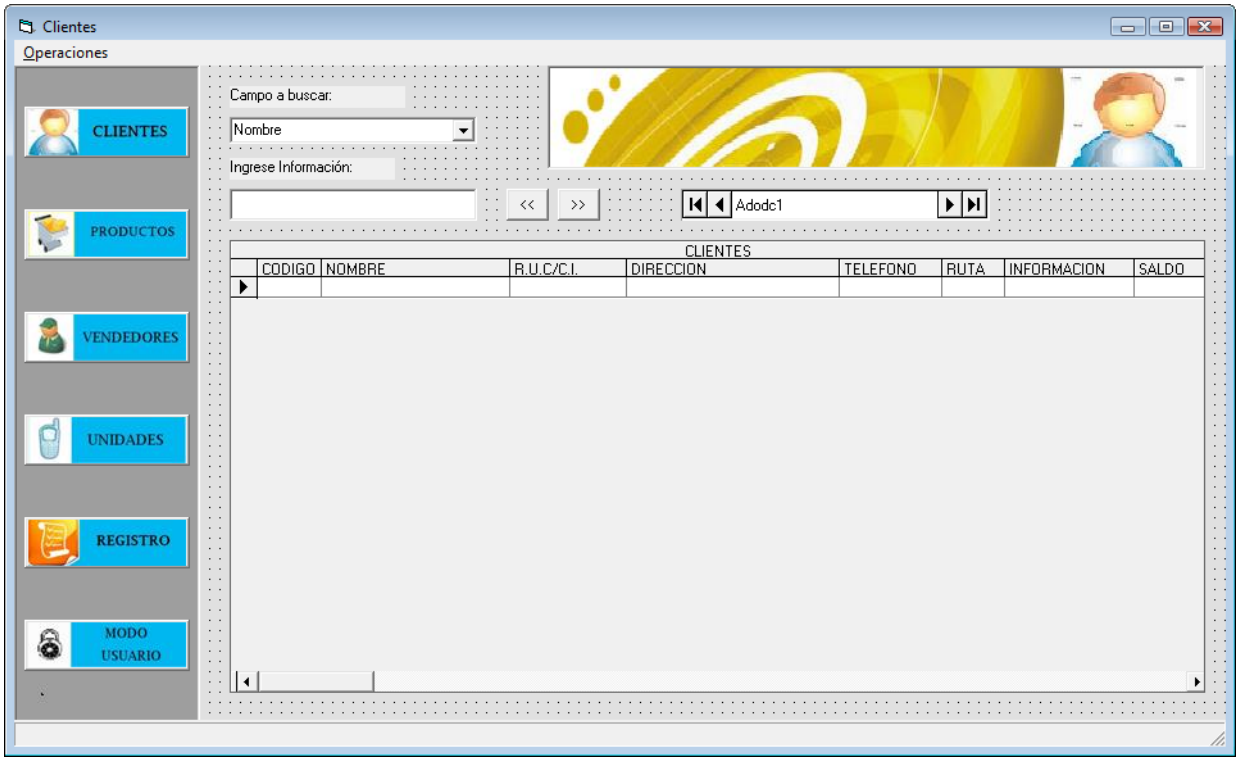


Figura. 7.6. Formulario de Gestión de Tabla de Información de Clientes.

Operaciones que se pueden realizar en el formulario:

Una vez que se ha seleccionado el modo de Administrador de la Aplicación HMI de la PC, el formulario permite realizar operaciones con los registros de la tabla, las operaciones se despliegan en el menú del formulario y se explican a continuación.

- *Editar:* Está operación permite editar algún registro que se haya seleccionado en el DataGrid.
- *Nuevo:* Está operación permite la inserción de un nuevo registro en la tabla de Clientes, en la que se deben ingresar por lo menos los parámetros que se encuentren con "*" en el formulario de la Figura. 7.7. Este formulario se usa también para la operación de Edición de un registro de la Tabla Clientes.

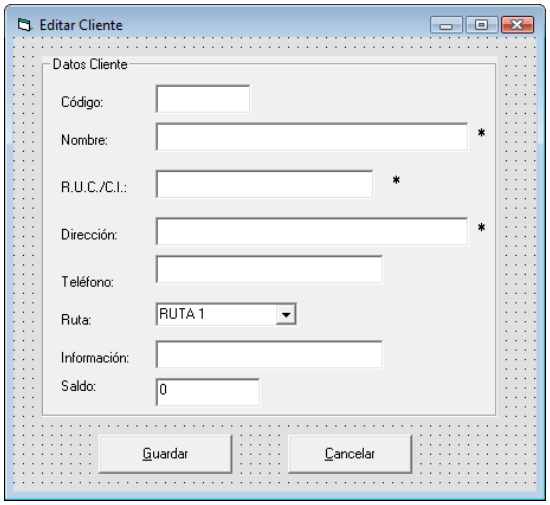


Figura. 7.7. Formulario de Edición e Inserción de Nuevo Registro en la Tabla Cliente.

El ingreso del campo Código de cliente se realiza automáticamente, es decir el programa genera el código ubicando si existe algún código disponible de algún registro eliminado, si no existiera un código disponible se asigna el siguiente código al último registro que se encuentre almacenado. El código se encuentra relacionado con la ruta que se elija.

- *Eliminar:* Está operación permite eliminar algún registro que se haya seleccionado en el DataGrid.

7.3.3. Formulario de Gestión de Tabla de Información de Productos

Este formulario permite al usuario tener acceso a la tabla de Productos de la base de datos de la distribuidora.

En la Figura. 7.8 se puede observar el formulario con todos los controles que se utilizaron para realizar la gestión de la Tabla de Productos, los mismos que realizan similar operación a la explicada anteriormente en la sección 7.3.2.

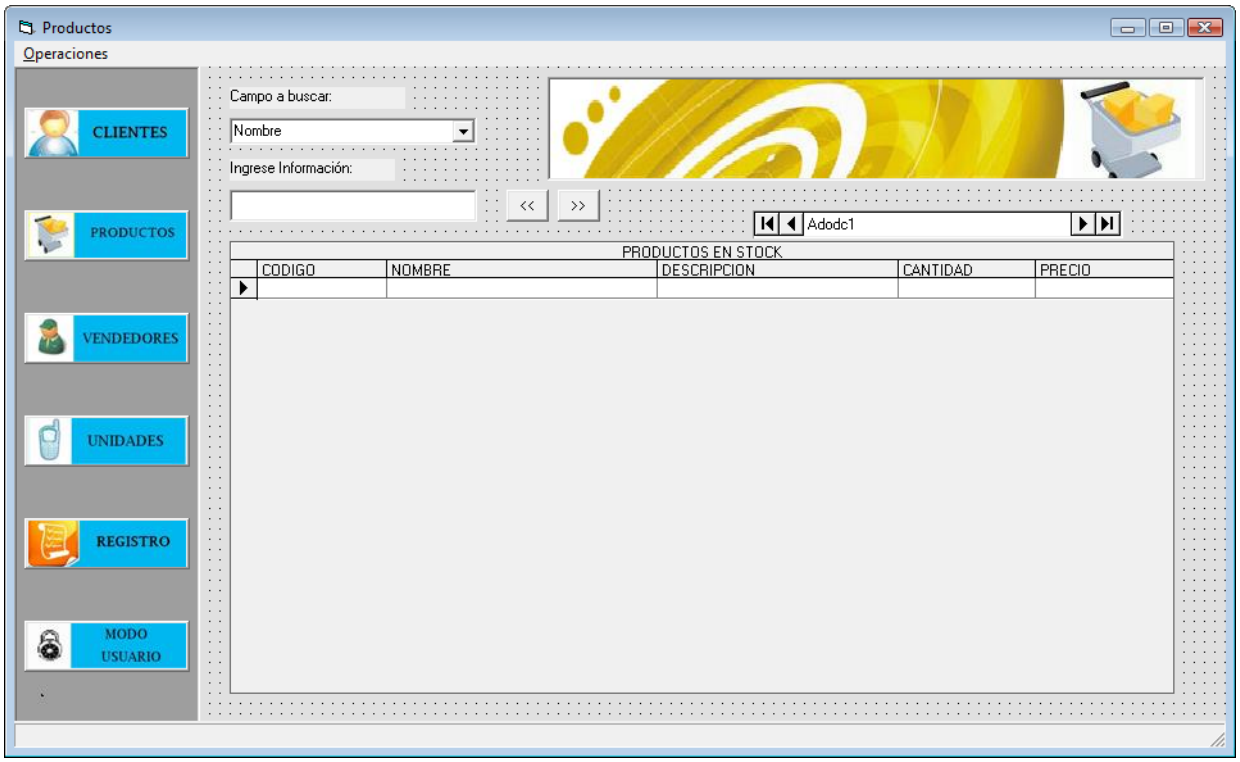


Figura. 7.8. Formulario para la Gestión de la Tabla de Información de Productos.

Las operaciones que se pueden realizar en este formulario son similares a las explicadas **¡Error! Marcador no definido.**en la sección 7.3.2. En la Figura. 7.99 se puede observar los parámetros que se deben ingresar, los parámetros con "*" son los mínimos para ser almacenados en el registro de la tabla de Productos.

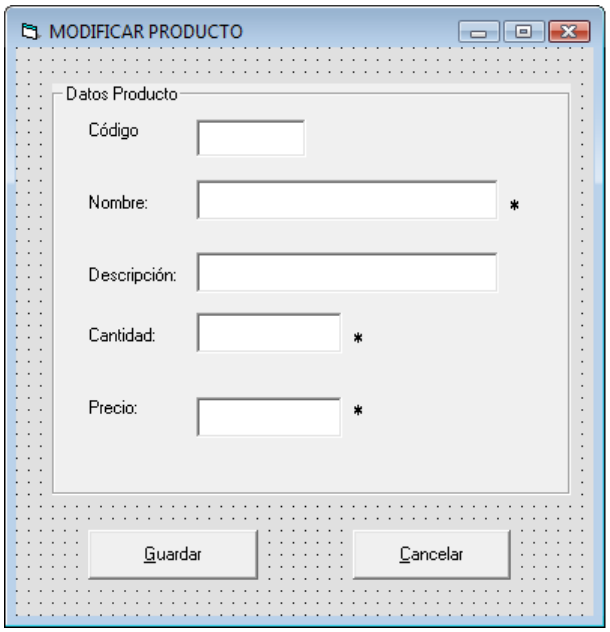


Figura. 7.9. Formulario de Edición e Inserción de Nuevo Registro en la Tabla Productos.

7.3.4. Formulario de Gestión de Tabla de Información de Vendedores

Este formulario permite al usuario tener acceso a la tabla de Vendedores de la base de datos de la distribuidora.

En la Figura. 7.1010 se puede observar el formulario con todos los controles que se utilizaron para realizar la gestión de la Tabla de Vendedores, los mismos que realizan similar operación a la explicada en en la página 70 7.3.2

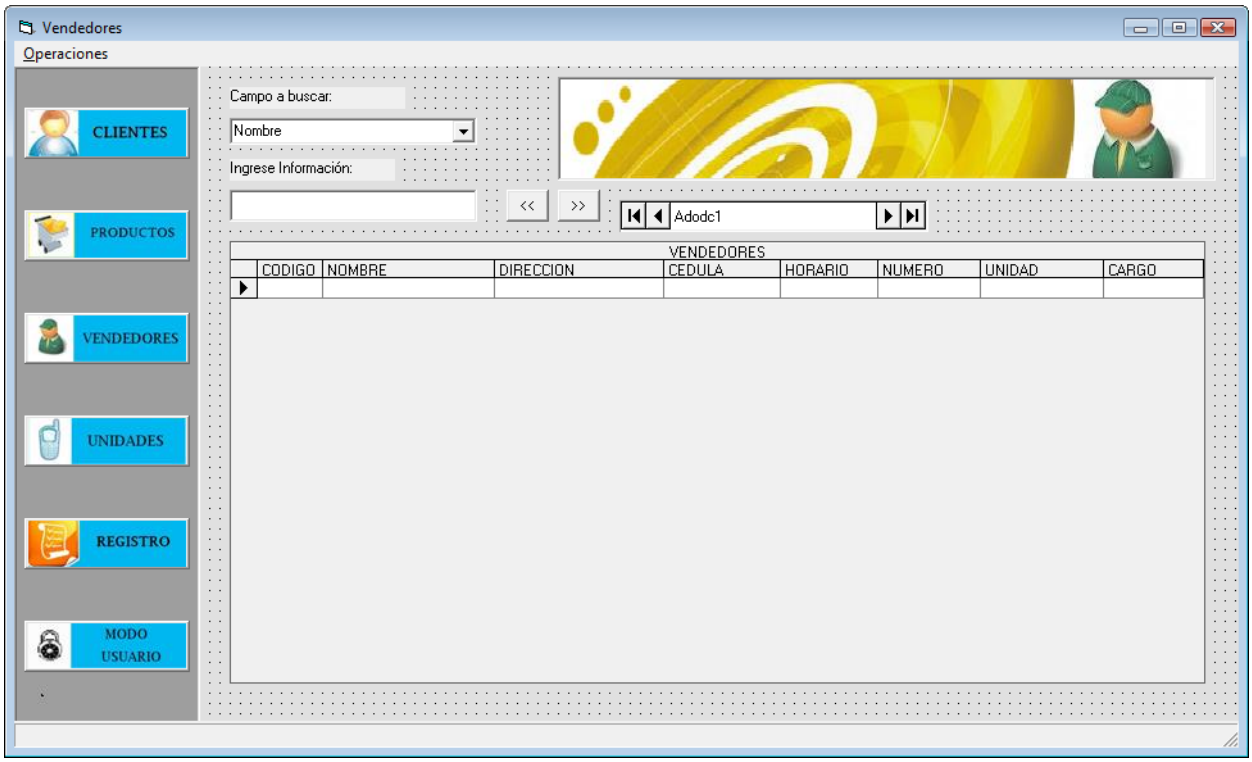


Figura. 7.10. Formulario para la Gestión de la Tabla de Información de Vendedores.

Las operaciones que se pueden realizar en este formulario son similares a las explicadas **¡Error! Marcador no definido.**en la sección 7.3.2. En la Figura. 7.11 se puede observar los parámetros que se deben ingresar, los parámetros con "*" son los mínimos para ser almacenados en el registro de la tabla de Vendedores.

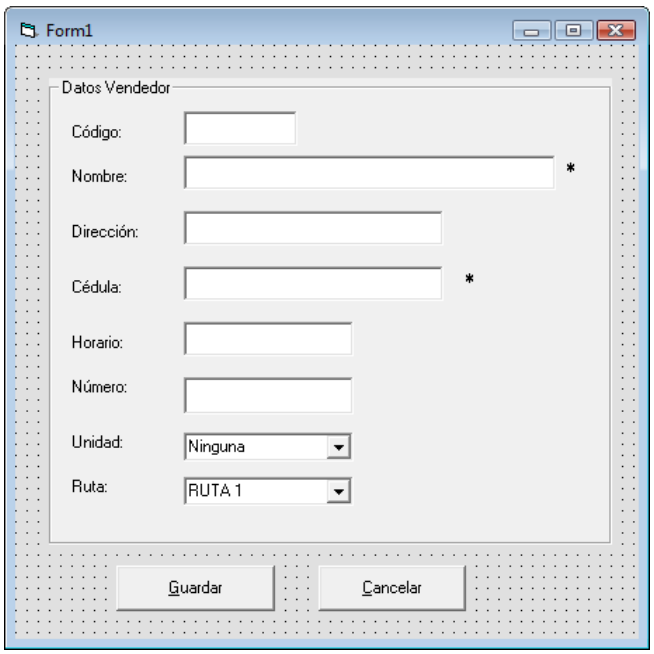


Figura. 7.11. Formulario de Edición e Inserción de Nuevo Registro en la Tabla Productos

7.3.5. Formulario de Gestión de Tabla de Información de Unidades

Este formulario permite al usuario tener acceso a la tabla de Unidades de la base de datos de la distribuidora.

En la Figura. 7.12 se puede observar el formulario con todos los controles que se utilizaron para realizar la gestión de la Tabla de Unidades, los mismos que realizan similar operación a la explicada en la página 70 7.3.2.

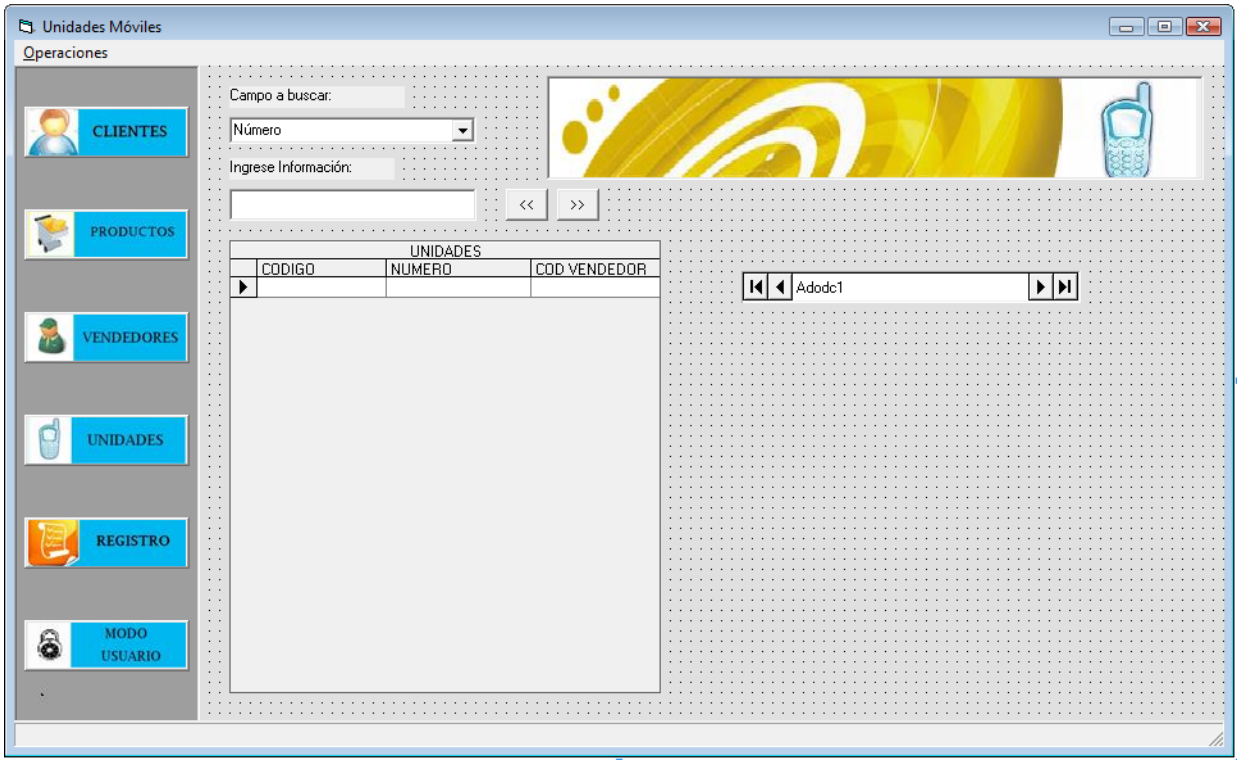


Figura. 7.12. Formulario para la Gestión de la Tabla de Información de Unidades.

Las operaciones que se pueden realizar en este formulario son similares a las explicadas **¡Error! Marcador no definido.**en la página 70 7.3.2. En la Figura. 7.13 se puede observar los parámetros que se deben ingresar, los parámetros con "*" son los mínimos para ser almacenados en el registro de la tabla de Vendedores.

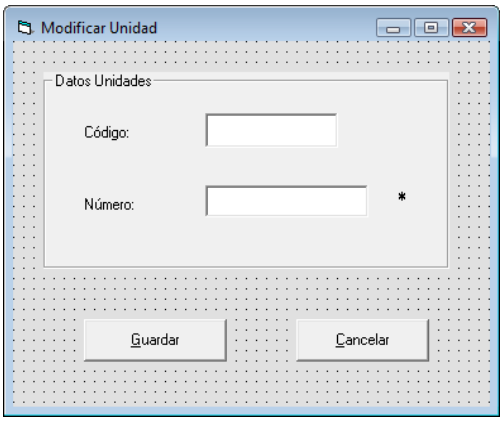


Figura. 7.13. Formulario de Edición e Inserción de Nuevo Registro en la Tabla de Unidades.

7.3.6. Formulario de Gestión de Tabla de Información de Registro

Este formulario permite al usuario tener acceso a la tabla de Registro de la base de datos de la distribuidora. En esta tabla se almacenan todos los pedidos que han sido procesados con éxito, es decir se almacena los pedidos que han generado la impresión de una Factura, la cual se explicará en la sección 7.7.1.

Se puede realizar un filtrado de información de registros en la tabla, ingresando información relacionada al parámetro que se seleccione en el control ComboBox, estos parámetros pueden ser: Nombre del Cliente, Código de Vendedor, Hora y Fecha de Facturación. El Filtrado se puede realizar haciendo click en el CommandButton con caption "Filtrar". Al hacer click en el CommandButton con Caption "Mostrar Todo" se puede mostrar toda la información de la Tabla, es decir sirve para contrarrestar el efecto que produce cuando se filtra una información sobre el DataGrid.

En la Figura. 7.14 se puede observar el formulario con todos los controles que se utilizaron para realizar la gestión de la Tabla de Registros.

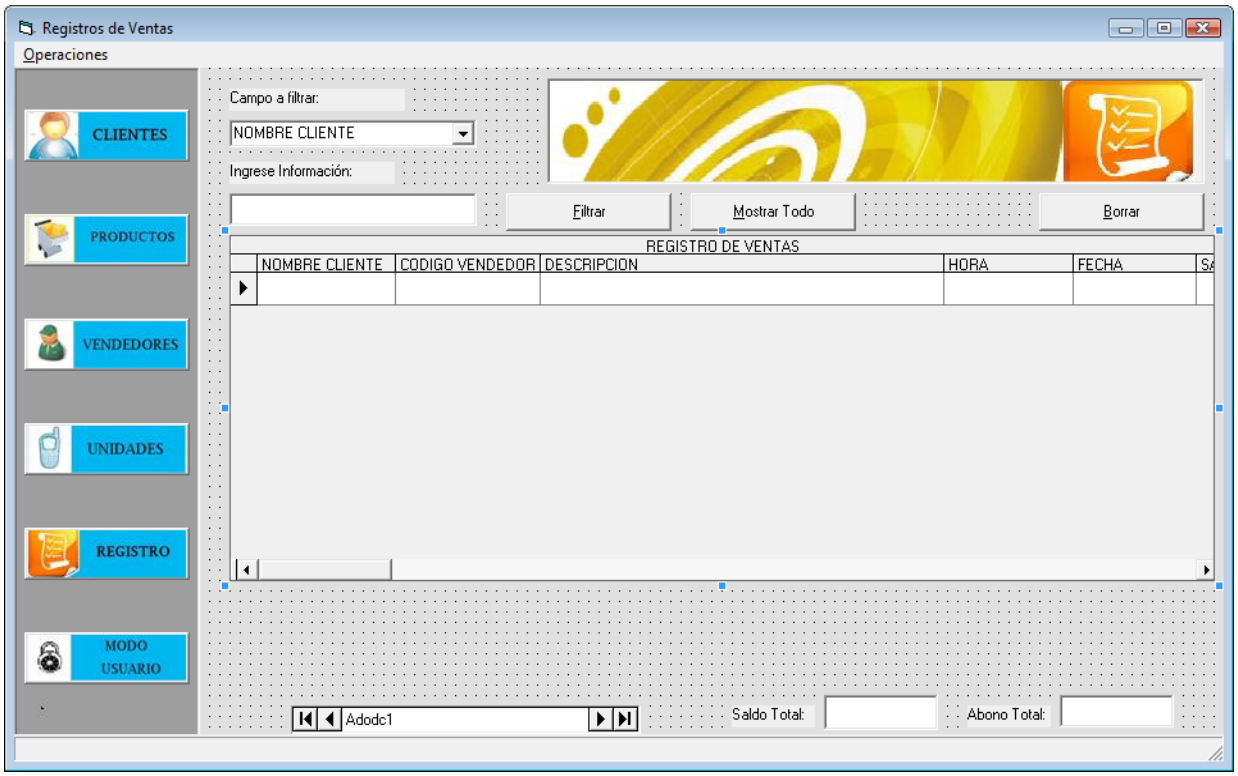


Figura. 7.14. Formulario para la Gestión de la Tabla de Información de Unidades.

El CommandButton con caption "Borrar" de la Figura. 7.14 permite eliminar un registro de la tabla de Registros de Ventas de la Base de Datos de la Distribuidora. Una vez que se ha hecho click sobre el CommandButton aparece el formulario de la que permite ingresar el número de registros que se desea eliminar. La eliminación de registros se realiza desde el primer registro que se encuentre almacenado en la Tabla de Registros de Ventas.

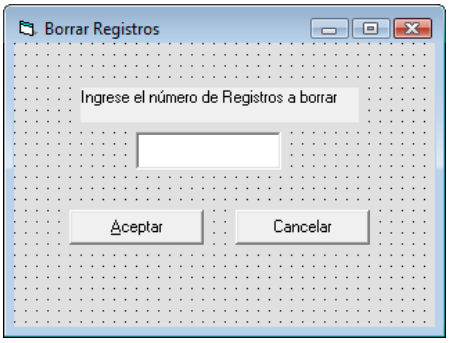


Figura. 7.15. Formulario para ingreso de registros a eliminar en la Tabla Registro de Ventas.

Operaciones que se pueden realizar en el formulario:

Las operaciones que se pueden realizar en el formulario al hacer click en el menú operaciones del formulario son las que se explican a continuación:

- Imprimir: Está operación permite imprimir una factura del registro de la tabla de Registros de Ventas que se haya seleccionado en el DataGrid.
- Editar Valores: Está operación permite editar los valores de los campos Saldo y Abono del registro que se haya seleccionado en el DataGrid, de tal manera que se pueda llevar un control de ingreso de Abono y Saldo totales de toda la Tabla de Registro de Ventas, estos valores se despliegan en los TextBox con los nombres correspondientes.
  En la Figura. 7.16 se puede observar el formulario en el que se pueden ingresar los valores para realizar las operaciones de Abono y saldo totales de la Tabla.

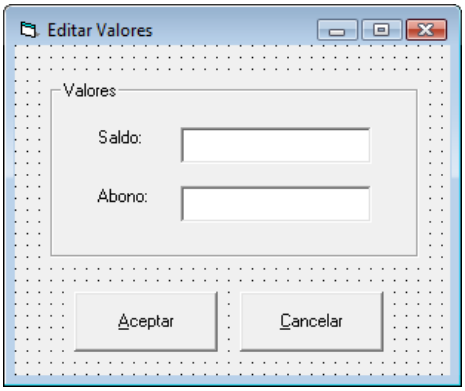


Figura. 7.16. Formulario de Edición de Nuevo Registro en la Tabla de Registros de Ventas.

7.3.7. Formulario para selección de Modo de Usuario

En este formulario se puede elegir el modo en el que se desee que trabaje la aplicación, a continuación se describe los modos en los que puede trabajar la aplicación:

- *Administrador:* En este modo de trabajo el usuario tiene la posibilidad de habilitar las operaciones que tienen los formularios anteriormente explicados, es decir en este modo el usuario puede modificar la información que se tiene almacenada en la base de datos.

- *Normal:* Este es el modo de trabajo por defecto de la aplicación, en este modo el usuario no tiene habilitado el menú de operaciones de los Formularios, es decir en este modo el usuario únicamente puede observar la información que se encuentra en la aplicación más no manipularla.

En la Figura. 7.17 se puede observar el formulario con los CommandButton que permiten elegir el modo en el que se desee que trabaje al aplicación.

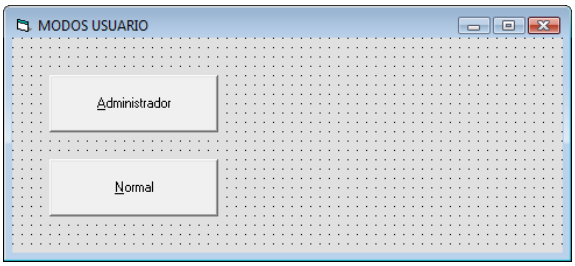


Figura. 7.17. Formulario para selección de Modo de Usuario.

Una vez que se haga clic en el Command Button con caption "Administrador", aparece el formulario de la Figura. 7.18, en este formulario se debe ingresar la clave para cambiar de modo de usuario, la misma que será comparada con la que se encuentra almacenada en la base de datos en la tabla Clave. Además, se puede cambiar la clave ingresando la clave actual y la nueva clave, la misma que editara el valor del campo de la Tabla de Clave con el nuevo valor.

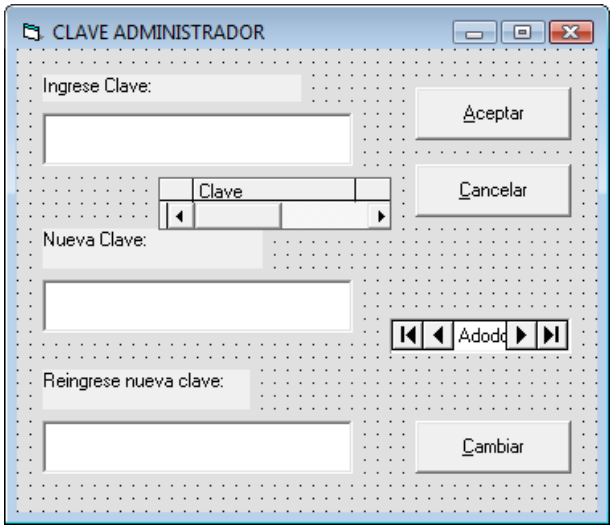


Figura. 7.18. Formulario para ingreso de clave de Administrador.

7.4. COMUNICACIÓN COM EL MODEM GSM

La comunicación con el Modem GSM (celular) se realiza por medio de la utilización del control no estándar de Visual Basic MSCOMM, a continuación se describen las principales propiedades de este control.

Propiedades del control MSCOMM

- **CommPort:** Indica el número del puerto serie usado. Cambiando esa propiedad se puede cambiar el puerto de comunicación que se va ha utilizar.
- **Settings:** Indica la velocidad, paridad, número de bits y bits de stop (parada) que se van a usar en la comunicación.
  - Los valores posibles para *velocidad* son (en baudios):

    50 100 110 300 600 1200 2400 4800 9600 14400 19200 y 28800
  - Los valores posibles para *paridad* son:

    N - No envía bit de paridad ni hace comprobación de paridad en la recepción.

    O - Envía y comprueba paridad, con el criterio de paridad IMPAR

    E - Envía y comprueba paridad, con criterio de paridad PAR
  - Los valores para el parámetro *Bits de Información* pueden ser:

    7 - Se envían / reciben 7 bits por trama de información.

    8 - Se envían / reciben 8 bits por trama de información

    5 - Se envían / reciben 5 bits por trama de información.
  - Los valores para el parámetro *Bits de parada* pueden ser:

    1 - Se envía un bit de parada

    2 - Se envían 2 bits de parada
- **Handshaking:** Especifica el método de control sobre el flujo de información. En una comunicación serie se necesita conocer si el puerto puede enviar información (necesita saber si el Modem está preparado para recibirla) y necesita indicarle al Modem que está preparado para recibir información. Establece las condiciones de control que uno va a tener sobre otro.

  El Control de Flujo puede hacerse de dos formas:
  - Mediante las señales auxiliares del puerto (RTS, CTS, DSR, DTR), que son cables adicionales que tendrán una tensión positiva respecto a los 0V del equipo si esa señal está activada, o una tensión negativa si no lo está.
  - Mediante señales especiales que se envían por los dos cables que transportan la información. Mediante estas dos señales se puede controlar que el ordenador envíe

información o deje de enviarla. De igual forma, podemos indicarle al Modem que envíe o no envíe. Estas señales especiales se denominan X-ON y X-OFF.

La propiedad Handshaking controla la forma de realizar este proceso. Puede tomar los siguientes valores:

0 - No existe Control de Flujo

1 - Control de Flujo mediante XON - XOFF

2 - Control de Flujo mediante Request To Send (RTS) y Clear To Send (CTS)

3 - Control de Flujo mediante XON - XOFF y RTS - CTS

- **InBufferSize:** Mediante esta propiedad se puede establecer el tamaño del Buffer (almacén de datos) de entrada. Este Buffer sirve para poder recibir datos sin que tenga que intervenir la aplicación continuamente para controlar el puerto de entrada.
- **OutBufferSize:** Mediante esta propiedad controlamos el tamaño del Buffer de salida. El tamaño de los Buffers depende de la aplicación y de la velocidad de comunicación. Mientras más grande sea el tamaño del buffer la velocidad de transferencia debe ser mayor.
- **RThreshold, SThreshold:** Estas dos propiedades especifican el número de caracteres que deben estar presentes en los Buffers de Recepción y Transmisión respectivamente, para que se produzca el evento OnComm relativo a recepción y transmisión de caracteres. (Eventos vReceive y EvSend).
- **InputLen:** Por defecto, cuando se lee el Buffer de recepción, se leen todos los caracteres, quedando el Buffer vacío. Si se le asigna a esta propiedad un valor distinto de 0, cada vez que se lee el Buffer de recepción leerá un número de caracteres igual a la cantidad ingresada, permaneciendo los caracteres restantes en el Buffer a la espera de una nueva lectura.
- **ParityReplace:** Si la comunicación se realiza con bit de paridad (Par o Impar), en recepción se comprueba byte a byte la recepción de la paridad correcta. Si se recibe un Byte que no tiene paridad correcta, lo más probable es que ese Byte (carácter) se haya recibido defectuoso. Esta propiedad permite sustituir un carácter que ha llegado con bit de paridad incorrecto por otro carácter (? predeterminado) o por una cadena de caracteres (Error, por ejemplo).
- **RTSEnable:** Activa (Pone a 1) la señal RTS (Request To Send - Petición de envío). Esta señal debe ponerse a 1 para indicar al Modem (o al equipo que va a recibir la

comunicación) que se desea enviar datos. Debe estar activada durante toda la transmisión de datos.

- **DTREnable:** Activa (Pone a 1) la salida DTR (Data Terminal Ready - Terminal de Datos Listo). Esta señal se emplea para decirle al Modem que el terminal está preparado para recibir datos.

Propiedades propias del Tiempo de Ejecución

- **PortOpen:** Abre el puerto de comunicación. Puede tener los valores True (para abrirlo) y False (para cerrarlo).
- **Output:** Envía caracteres al Buffer de salida.
- **Input:** Lee el Buffer de recepción. Se leerá un número de caracteres igual al valor de la propiedad InputLen. Cuando la propiedad InputLen tiene el valor 0, el Buffer se lee completo.
- **CommEvent:** Devuelve el evento más reciente que ha ocurrido para generar el evento general

Los parámetros de configuración de las propiedades de comunicación del control MSCOMM utilizados en la aplicación son los que a continuación se muestran en las líneas de código del programa:

```
'Especificar el puerto COM que se desea abrir
    MSComm.CommPort = 1 ' número del puerto (1, 2, ...)
' Limpiar las colas Rx y Tx
   MSComm.InBufferCount = 0
   MSComm.OutBufferCount = 0
'Lee todo lo que se encuentra en el buffer
   MSComm.InputLen = 0
'modo texto =0, binario=1
   MSComm.InputMode = 0
' Baudios, paridad, número de bits de datos y de parada
' velocidad=9600, paridad=No usada,
' cantidad de bits=8,
'cantidad de bits de parada (stop bits) = 1
'9600 baudios, sin paridad, 8 bits de datos y 1 bit de parada.
   MSComm.Settings = "9600,N,8,1"
'Caracteres que puede admitir el buffer de transmisión antes de que el control genere el
evento OnComm.
'Su valor predeterminado es 0
   MSComm.SThreshold = 1
```

```
'Caracteres que se van recibir antes de que el control genere
'el evento OnComm. Su valor predeterminado es 0.
   MSComm.RThreshold = 1
' Abrir el puerto de comunicaciones
   MSComm.PortOpen = True
```

Las líneas anteriormente descritas se encuentran ubicadas en el evento Load() del formulario de Envío y Recepción de SMS, explicado en 7.3.2, es decir una vez que se abra la aplicación todos los valores de los parámetros de las propiedades de la comunicación serán configurados.

7.5. VERIFICACIÓN DE PRODUCTOS EN STOCK

Para realizar la verificación de la existencia de productos en Stock, en el campo Cantidad de la tabla de Productos de la base de datos de la Distribuidora, se utilizó el evento comEvReceive del control MSCOMM de visual basic, el cual detecta cuando ha llegado algún SMS del dispositivo móvil.

En las siguientes líneas de código se explica con detalle como se procesa la información de un SMS que se recibe en la aplicación. Las líneas de código que permiten recibir el mensaje se presentan a continuación:

```
'Lectura de caracteres de SMS de una en una
caracter_celular = MSComm.Input
car_cel_asc = str(Asc(caracter_celular))
'Detecta carácter de enter en asc 10 - 13
If car_cel_asc <> 10 And car_cel_asc <> 13 Then
   txtRX.Text = txtRX.Text + caracter_celular
   mensaje_celular = mensaje_celular + caracter_celular
End If
txtRX.Text = mensaje_celular
```

Las líneas de código que permiten obtener la información del SMS, son las que se muestran a continuación:

```
'Detecta carácter de fin de SMS
If caracter_celular = "~" Then
```

```
        longitud = Len(mensaje_celular)
        caracter_celular = ""
        i = 1
        While caracter_celular <> "+"
                caracter_celular = Mid(mensaje_celular, i, 1)
                i = i + 1
        Wend
        mensaje_celular = Mid(mensaje_celular, i - 1, longitud - i + 2)
        car = ""
        'decodificacion del mensaje recibido
        num_celular = ""
        fecha = ""
        hora = ""
        num_celular = "0" + Mid(mensaje_celular, 12, 8)
        Text1.Text = num_celular
        fecha = Mid(mensaje_celular, 24, 8)
        Text2.Text = fecha
        hora = Mid(mensaje_celular, 33, 8)
        Text3.Text = hora
        i = 1
        'Detecta el carácter de inicio de información de SMS
        While car <> "^"
                car = Mid(mensaje_celular, i, 1)
                i = i + 1
        Wend
        ……..
        ……..
    End If
```

Las siguientes líneas de código muestran como se detecta si la información de SMS es una solicitud de pedido o actualización de la información de la base de datos del dispositivo móvil.

```
    'se lee el primer carácter de la información del SMS, si es &=actualización si es otro
    caracter=solicitud de pedido.
    car_op = Mid(mensaje_celular, i, 1)
    If car_op = "&" Then
        ……..
        ……..
    End If
```

Para las actualizaciones únicamente se permite en un número máximo de 10 donde 0 es la décima solicitud de actualización.

```
'Determina el número de la actualización que se solicita
num_act = Val(Mid(mensaje_celular, i + 1, 1))
If num_act = 0 Then
   num_act = 10
End If
```

Una vez que se ha determinado si es actualización o solicitud de pedido se procede a comprobar el número de celular del dispositivo que envía el SMS, para determinar si existe o no el número registrado como dispositivo válido. Las siguientes líneas de código realizan lo explicado:

```
'ubicamos al puntero de los registros al inicio para búsqueda
frm_unidades.Adodc1.Recordset.MoveFirst
'busca si consta en la base de datos el número de la unidad
frm_unidades.Adodc1.Recordset.Find "NUMERO ='" & num_celular & "'"
```

Si el número de dispositivo que envía el SMS se encuentra registrado en la tabla de Unidades de la base de datos de la Distribuidora, se busca el código de cliente en la tabla de clientes. Además, con el índice del registro encontrado se pude obtener la información de algunos campos de tabla de Unidades. Las siguientes líneas de código realizan está operación:

```
'Si consta la unidad en la tabla
If frm_unidades.Adodc1.Recordset.EOF = False Then
num_celular = num_celular_bcd(num_celular)
cod_unidad = ""
'copio la informacion del codigo del vendedor
cod_unidad = frm_unidades.Adodc1.Recordset.Fields.Item(1)
cod_vendedor = frm_unidades.Adodc1.Recordset.Fields.Item(3)
'ubicamos al puntero de los registros al inicio para búsqueda
frm_cliente.Adodc1.Recordset.MoveFirst
'busco si consta en la base el cliente
frm_cliente.Adodc1.Recordset.Find "CODIGO ='" & cod_cliente & "'"
... ... .
... ... .
```

```
End If
```

Una vez que se verifica la existencia del código de cliente en la tabla de información de clientes, se decodifica la información del pedido.

La decodificación de la información de los pedidos se realiza por medio del valor en decimal de la tabla de códigos ascii, por ejemplo para el primer producto en la tabla de productos es A siendo equivalente a P001, donde A en decimal según la tabla de códigos Ascii es 65 por está razón se resta 64 de los pedidos para obtener el número de 1 del código de producto (P001). Las letras utilizadas para la codificación según la tabla de códigos Ascii son desde 65 a 90 y desde la 97 a la 122. Las siguientes líneas de código realizan está operación:

```
'bucle para decodificación de sms para obtener código de productos
For i = 1 To num_productos
   producto(i) = Mid(productos, i + 2 * (i - 1), 1)
    If Val(Asc(producto(i))) <= 90 Then
            producto(i) = str(Val(Asc(producto(i))) - 64)
    Else
            producto(i) = str(Val(Asc(producto(i))) - 71)
    End If
    'Longitud de código de producto es de 4 caracteres
    longitud_cod = Len(producto(i))
    producto(i) = Mid(producto(i), 2, longitud_cod - 1)
    If longitud_cod = 2 Then
                    producto(i) = "00" + producto(i)
    End If
    If longitud_cod = 3 Then
                    producto(i) = "0" + producto(i)
    End If
    producto(i) = "P" + producto(i)
    cantidad_producto(i) = Mid(productos, i + 2 * (i - 1) + 1, 2)
    ... .
    ... .
Next
```

Una vez que se obtiene el código del producto por ejemplo P001, se busca el código en la tabla de Productos como se muestra en las siguientes líneas de código.

```
frm_producto.Adodc1.Recordset.MoveFirst
frm_producto.Adodc1.Recordset.Find "CODIGO ='" & producto(i) & "' cantidad_stock =
Val(frm_producto.Adodc1.Recordset.Fields.Item(4))
```

Con el código de producto encontrado se procede a verificar si existe o no producto en stock restando lo solicitado con el producto existe, como se muestra a continuación:

```
cantidad_pedida = Val(cantidad_producto(i))
cantidad = cantidad_stock - cantidad_pedida
```

Si la cantidad solicitada es mayor a la existente, se genera el mensaje de respuesta correspondiente al dispositivo móvil.

7.6. GENERACIÓN DE MENSAJES SMS DE RESPUESTA

La aplicación se encuentra diseñada para que responda a cualquier SMS recibido, para poder generar los mensajes de respuesta al dispositivo móvil que ha enviado el SMS, fue necesario el uso de formato PDU (Capítulo 3) para el manejo del envío de SMS del Modem GSM.

Por medio de una trama SMS-SUBMIT se puede configurar los parámetros para direccionar el SMS al puerto de espera de SMS de la aplicación desarrollada en Netbeans. Ver más en Capítulo 2.

A continuación se muestra un ejemplo de los parámetros que se necesitan enviar para una trama SMS-SUBMIT. En la **¡Error! No se encuentra el origen de la referencia.** se describe los parámetros enviados en un SMS con formato PDU.

Octeto(s)	Campo	Descripción
00	SCA	Longitud del número de teléfono del SMSC. En este caso la longitud es 0, lo que significa que el número del SMSC almacenado en el teléfono debe ser usado.
41	PDU TYPE	En este caso se han activado los bits 6 (UDHI) y 0 (MTI) para indicar que existe una cabecera en los datos de usuario y que se trata de una trama SMS-SUBMIT respectivamente.
00	MR	El valor "00" permite al teléfono determinar el número de mensaje de referencia por si mismo.
09	Address Length	Longitud del número de teléfono del SME de destino.
81	Type of Address	Tipo de número. El valor "81" indica que se trata del formato nacional del número de teléfono.
90522481F5	Phone Number	Número de teléfono en semioctetos (092542185) permutados de dos en dos. Debido a que la longitud del número de teléfono es impar (9 semioctetos), se añade una 'F' como si el número de teléfono fuera "092542185F". Si se usa el formato internacional (91 en lugar de 81, en el campo Type os Address), para Ecuador antepuesto el prefijo 593 (59392542185), la secuencia de octetos permutados sería 959390522481F5, en donde la longitud es 11 (0B hexadecimal), la cual también es impar.
00	PID	Protocolo SME-to-SME
00	DCS	Este mensaje es codificado de acuerdo al Alfabeto de 7 bits.
11	UDL	Longitud del campo UD (datos de usuario). Como en el campo DCS se especificó la codificación de los datos con 7 bits, la longitud indica el número de septetos (17). Si en el campo DCS se hubiera especificado la codificación con 8 bits o Unicode, la longitud haría referencia al número de octetos.
06	UDHL	Como en el campo PDU TYPE se especificó la existencia de una cabecera en los datos de usuario (UDH), entonces el valor "06" indica la longitud en octetos del campo UDH.
05	IEI	El valor "05" indica que se trata del elemento de información que permite realizar el direccionamiento a puertos de aplicación utilizando el esquema 16 bits.
04	IEDL	Longitud de los datos del elemento de información. El valor "04" indica el número de octetos, en este caso los dos primeros octetos determinan el puerto destino y los otros dos restantes el puerto origen.
4074	IED	Puerto destino 16500 (decimal). Estos octetos indican la dirección de 16 bits del puerto del receptor.
4074		Puerto origen 16500 (decimal). Estos octetos indican la dirección de 16 bits del puerto del emisor.
C6E0905A95069F4B	Data 7 bit encoding	Estos octetos representan el mensaje "FACTURAOK". El procedimiento para realizar la transformación de septetos a octetos se muestra en Anexo 1 *de conversión de 7 a 8 bits* (GSM 7-bit Default Alphabet y codificación de datos de 7 bits en octetos).

Tabla. 7.1. Parámetros de la trama SMS-SUBMIT para envió de SMS.

La función sms_decodificado(sms As String) realiza la conversión de los mensajes de respuesta en formato PDU, la cual se explica a continuación:

Se saca la longitud total del mensaje que se desea mandar como respuesta, una vez obtenida la longitud se procede a convertir cada carácter del mensaje en binario, como se puede observar en las siguientes líneas de código:

```
Longitud_mensaje = Len(sms)
i1 = 1
```

```
sms_decodificado = ""
For i1 = 1 To Longitud_mensaje
   Valors = Mid(sms, i1, 1)
   Valor = Asc(Valors)
   Binario = bin(Valor)
   String_binario(i1) =
   Mid(Binario, 2, 7)
Next
```

Una vez que se tiene todo el mensaje en forma binaria se realiza el procedimiento de conversión de septetos a octetos, lo que se muestra a continuación

```
String_binario(i1) = "f"
Aux1 = ""
sms_decodificado = ""
i1 = 1
j1 = 7
k1 = 1
While Aux1 <> "f"
   Aux1 = Mid(String_binario(i1 + 1), 1, 1)
   If Aux1 <> "f" Then
            'cada 8 caracteres decodificados se vuelve a iniciar el proceso de conversión
           'de 7 a 8 bits.
           If k1 <> 8 Then
                    aux = Mid(String_binario(i1 + 1), j1, k1)
                    String_binario(i1) = Mid(String_binario(i1), 1, 8 - k1)
                    String_binario(i1) = aux + String_binario(i1)
                    Hexa1 = Mid(String_binario(i1), 1, 4)
                    Hexa2 = Mid(String_binario(i1), 5, 4)
                    Hexa11 = Mid(Hex(dec(Hexa1)), 1, 1)
                    Hexa22 = Mid(Hex(dec(Hexa2)), 1, 1)
                    sms_decodificado = sms_decodificado + Hexa11 + Hexa22
                    i1 = i1 + 1
                    j1 = j1 - 1
                    k1 = k1 + 1
           Else
                    String_binario(i) = ""
                    i1 = i1 + 1
                    k1 = 1
                    j1 = 7
```

```
            End If
        Else
            String_binario(i1) = Mid(String_binario(i1), 1, 8 - k1)
            For g1 = 1 To k1
                String_binario(i1) = "0" + String_binario(i1)
            Next
            Hexa1 = Mid(String_binario(i1), 1, 4)
            Hexa2 = Mid(String_binario(i1), 5, 4)
            Hexa11 = Mid(Hex(dec(Hexa1)), 1, 1)
            Hexa22 = Mid(Hex(dec(Hexa2)), 1, 1)
            'se aumenta el '00' cuando el tamaño de la longitud del mensaje de 'respuesta es
            de 8
            If final_pdu = 0 Then
                If Hexa11 + Hexa22 <> "00" Then
                    sms_decodificado = sms_decodificado + Hexa11 + Hexa22
                End If
            Else
                sms_decodificado = sms_decodificado + Hexa11 + Hexa22
            End If
        End If
    Wend
```

Para formar la trama completa SMS-SUBMIT se utiliza la función sms_respuesta(resp As String, numero As String), la cual se explica a continuación.

```
    'la longitud del mensaje se saca en hexadecimal
    long_sms = Hex(Len(resp) + 8)
    If Len(long_sms) = 1 Then
        long_sms = "0" + long_sms
    End If
    bandera = 1
    long_resp1 = 0
    resta = 0
    resp_8 = 0
    resp_a = ""
    co = 1
    long_resp1 = Len(resp)
    'bucle para conversión para conversión de 7 a 8 bits cada 8 caracteres de  'mensaje
    While bandera = 1
        resta = long_resp1 - 8
```

```
        If resta >= 0 Then
                final_pdu = 0
                long_resp1 = resta
                resp_8 = Mid(resp, co, 8)
                resp_a = resp_a + sms_decodificado(resp_8)
                co = co + 8
                If resta = 0 Then
                        bandera = 2
                End If
        Else
                final_pdu = 1
                resp_8 = Mid(resp, co, long_resp1)
                resp_a = resp_a + sms_decodificado(resp_8)
                bandera = 2
        End If
    Wend
    'mensaje de respuesta con los parámetros de configuración de la trama SMS-BUBMIT
    'numero es el número telefónico permutado de dos en dos se obtiene como resultado de la
    función num_celular_bcd(n_celular As String)
    sms_respuesta = "00041000981" + numero + "0000" + long_sms + "060504407440 74" +
    resp_a
    'la longitud que se obtiene es la de número de octetos por eso se divide para 2
    long_sms = str((Len(sms_respuesta) - 2) / 2)
    longitud = Len(long_sms)
    long_sms = Mid(long_sms, 2, Len(long_sms) - 1)
    If longitud = 4 Then
         sms_respuesta = long_sms + sms_respuesta + "$"
    Else
        sms_respuesta = "0" + long_sms + sms_respuesta + "$"
    End If
```

Existen cinco tipos de respuesta que puede generar la aplicación las cuales se explica a continuación:

- **Factura procesada con éxito:** Este mensaje de respuesta se genera cuando ha existido la cantidad requerida de pedido en la tabla de productos, de la cantidad solicitada por el cliente. Las siguientes líneas muestran la respuesta generada:

```
    mensaje_respuesta = "FACOK"
    mensaje_respuesta = sms_respuesta(mensaje_respuesta, num_celular)
    MSComm.Output = mensaje_respuesta
```

- **Código de Cliente Inválido:** Este mensaje de respuesta se genera cuando el código enviado en el pedido no se ha encontrado en la tabla de clientes. Las siguientes líneas de código muestran como se envía el mensaje:

```
mensaje_respuesta = "COD"
mensaje_respuesta = sms_respuesta(mensaje_respuesta, num_celular)
MSComm.Output = mensaje_respuesta
```

- **Operación no disponible:** Este mensaje de respuesta se genera cuando el número telefónico del dispositivo móvil no se ha encontrado en la tabla de unidades. Las siguientes líneas de código muestran como se envía el mensaje:

```
mensaje_respuesta = "OND"
mensaje_respuesta = sms_respuesta(mensaje_respuesta, num_celular)
MSComm.Output = mensaje_respuesta
```

- **Mensajes de información de pedido no suficiente:** Este mensaje se genera cuando los productos de la solicitud de pedido excede en cantidad a lo que se encuentra en el campo cantidad de la tabla de productos. A continuación se muestra como se envía este mensaje:

```
For i = 1 To j – 1
        If Val(Mid(producto_faltante(i), 3, 2)) <= 25 Then
                producto_faltante(i) = Chr$(Val(Mid(producto_faltante(i), 3, 2)) + 64)
        Else
                |producto_faltante(i) = Chr$(Val(Mid(producto_faltante(i), 3, 2)) +
                        71)
        End If
        longitud = Len(cantidad_disponible(i))
        If longitud = 1 Then
                cantidad_disponible(i) = "0" + cantidad_disponible(i)
        End If
        men_faltante = men_faltante + producto_faltante(i) + cantidad_disponible(i)
Next
```

El mensaje decodificado envia las cantidades existentes del producto en la table de productos.

```
'mensaje cuando existe faltante en pedido
MSComm.PortOpen = False
MSComm.PortOpen = True
mensaje_respuesta = ""
mensaje_respuesta = men_faltante
mensaje_respuesta = sms_respuesta(mensaje_respuesta, num_celular
MSComm.Output = mensaje_respuesta
```

- **Actualizaciones:** Las actualizaciones de la base de datos se realizan por medio de la función actualizacion(num_act), la cual se explica a continuación:

  Por medio de un bucle se recorre todos los campos de nombre de producto y precio de la tabla de productos, se van formando mensajes con un tamaño de hasta 118 caracteres, ya que junto con el resto de datos de la trama SMS-SUBMIT que se envían en PDU se completa la longitud del SMS, se verifica el número de actualización que se solicita para enviar el resto de información de la tabla. A continuación se puede observar lo explicado:

```
frm_producto.Adodc1.Recordset.MoveFirst
While frm_producto.Adodc1.Recordset.EOF = False
    aux_act = ""
    cont = cont + 1
    While flag = 1
            aux_act = aux_act + frm_producto.Adodc1.Recordset.Fields.Item(2) + ":" +
frm_producto.Adodc1.Recordset.Fields.Item(5) + ";"
        tam_sms = Len(aux_act)
        If tam_sms < 118 Then
                act(cont) = aux_act
                frm_producto.Adodc1.Recordset.MoveNext
                If frm_producto.Adodc1.Recordset.EOF = True Then
                    flag = 2
                End If
        Else
                flag = 2
        End If
    Wend
    flag = 1
Wend
If num_ac <> 10 Then
        numa = Mid(str(num_ac), 2, 1)
Else
        numa = "0"
End If
If cont = 10 Then
        num_total = "&0"
Else
        num_total = "&" + Mid(str(cont), 2, 1)
End If
actualizacion = num_total + numa + ";" + act(num_ac)
```

Para enviar el mensaje de actualización se utiliza las siguientes líneas de código:

```
mensaje_respuesta = ""
mensaje_respuesta = actualizacion(num_act)
mensaje_respuesta = sms_respuesta(mensaje_respuesta, num_celular)
MSComm.Output = mensaje_respuesta
```

7.7. ELECCIÓN Y MANEJO DE IMPRESORA

Una impresora o dispositivo de impresión es un periférico que, cuando conectado a una computadora o a una red de computadoras, tiene la función de dispositivo de salida, imprimiendo textos, gráficos o cualquier otro resultado de una aplicación.

Heredando la tecnología de las máquinas de escribir, las impresoras sufrieron drásticas mutaciones a lo largo del tiempo. También con la evolución de la computación gráfica, las impresoras se fueron especializando para cada una de las especialidades. Así, se encuentran impresoras optimizadas para dibujo vectorial, para impresiones de imágenes, y otras optimizadas para texto.

La tecnología de impresión fue incluida en varios sistemas de comunicación, como el fax por ejemplo.

La impresora que se ha elegido para el prototipo es una Impresora de Impactos, por las siguientes ventajas y desventajas que presenta este tipo de impresora.

Ventajas

Las impresoras matriciales, como cualquier impresora de impacto, puede imprimir en papel multicapa o hacer copias carbón. Dichas impresoras tienen un bajo costo de impresión por página. Conforme se termina la tinta, la impresión pierde intensidad gradualmente en lugar de terminar repentinamente durante un trabajo. Pueden trabajar con papel continuo en lugar de requerir hojas individuales, lo que las hace útiles para impresión de registros de datos. Son buenas en general para situaciones en las que la resistencia y durabilidad sea más importante que la calidad de impresión.

Desventajas

Las impresoras de impacto suelen ser ruidosas, hasta el punto de que existen carcasas aislantes para su uso en entornos silenciosos. Sólo pueden imprimir texto y gráficos, con una resolución de color limitada, relativamente baja calidad y a poca velocidad. Aunque suelen ser la mejor solución para imprimir etiquetas y tickets, son propensas a que falle uno de los pines del cabezal de impresión, dejando zonas apagadas en el texto.

Para el manejo de la impresora se hizo uso del objeto Printer de Visual Basic, a continuación se describe las propiedades utilizadas de este objeto:

- **Printer.Font:** Obtiene o establece la familia de fuentes.
- **Printer.CurrentY:** Obtiene o establece las coordenadas verticales para la impresión o para unmétodo gráfico.
- **Printer.CurrentX**: Obtiene o establece las coordenadas horizontales para la impresión o para unmétodo gráfico.
- **Printer.EndDoc:** Finaliza una operación de impresión enviada al objeto Printer, liberando el documento al dispositivo de impresión o cola de impresión.

7.7.1. Generación e Impresión de Facturas

Para la generación e impresión de facturas se realizaron las siguientes líneas de código, la ubicación se hizo en referencia a una factura ya diseñada, las líneas comentadas explican como se realiza el procedimiento:

```
margen_x = 800
 margen_y = 1800
 nom_cliente_impresion = nombre_cliente
 fecha_impresion = fecha
 cod_impresion = cod_cliente
 direccion_impresion = direccion_cliente
 ruc_impresion = ruc_cliente
 telefono_impresion = telefono_cliente
 descripcion_impresion = mensaje_celular
 margen_y = margen_y + 400
 Printer.CurrentY = margen_y
 Printer.CurrentX = margen_x + 1200
 Printer.Print nom_cliente_impresion & " (" & cod_impresion & ")"
```

```
    margen_y = margen_y + 350
    Printer.CurrentY = margen_y
    Printer.CurrentX = margen_x + 1200
    Printer.Print direccion_impresion
    Printer.CurrentY = margen_y
    Printer.CurrentX = margen_x + 5800
    Printer.Print telefono_impresion
    margen_y = margen_y + 350
    Printer.CurrentY = margen_y
    Printer.CurrentX = margen_x + 1200
    Printer.Print fecha_impresion
    Printer.CurrentY = margen_y
    Printer.CurrentX = margen_x + 5800
    Printer.Print ruc_impresion
    car = ""
    'longitud de descripción
    longitud = Len(descripcion)
    j = 0
    k = 2
    l = 0
    margen_y = margen_y + 700
    'bucle para desplazarse  a  lo largo de todo el contenido de descripción
    For i = 1 To longitud
       car = Mid(descripcion, i, 1)
       If car = "_" Then
        'se incrementa el valor de j para ir obteniendo los datos de los productos por el
separador '_'
        j = j + 1
        If j = 1 Then
         cantidad_impresion = Mid(descripcion, k, i - k)
         'se incrementa el valor de k para saltarse el caracter de separacon '_'
         k = i + 1
         Printer.CurrentY = margen_y
         Printer.CurrentX = margen_x
         Printer.Print cantidad_impresion
         l = 1
        End If
        If j = 2 Then
         nombre_impresion = Mid(descripcion, k, i - k)
         k = i + 1
```

```
            Printer.CurrentY = margen_y
            Printer.CurrentX = margen_x + 900
          Printer.Print nombre_impresion
          End If
          If j = 3 Then
            precio_unitario_impresion = Mid(descripcion, k, i - k)
            k = i + 1
            Printer.CurrentY = margen_y
            Printer.CurrentX = margen_x + 5250
            Printer.Print precio_unitario_impresion
          End If
          If j = 4 Then
            precio_impresion = Mid(descripcion, k, i - k)
            k = i + 1
            Printer.CurrentY = margen_y
            Printer.CurrentX = margen_x + 6500
            Printer.Print precio_impresion
          End If
        End If
    'l=1 para indicar que ya se puede empezar a coger los datos del segundo producto (si
    hubiera)
        If l = 1 Then
        '*' separador de productos
        If car = "*" Then
          precio_total_impresion = Mid(descripcion, k + 1, i - 1 - k)
          Printer.CurrentY = margen_y
          Printer.CurrentX = margen_x + 6400
          Printer.Print precio_total_impresion
          margen_y = margen_y + 250
          'se incrementa el valor de k para que se salte el caracter de * de 'descripción'
          k = i + 1
     'se coloca j=0 para empezar a coger la informacion de productos por el separador *
          j = 0
        End If
        End If
        '$' separador de fin de productos e indicador de suma total
        If car = "$" Then
          'el valor es de i+2 para saltarse el caracter de ' '
          precio_final_impresion = Mid(descripcion, i + 2, longitud - i)
          Printer.CurrentY = 10000
```

```
            Printer.CurrentX = 7200
            Printer.Print precio_final_impresion
            margen_y = margen_y + 250
            j = 0
          End If
        Next
        Printer.EndDoc
```

7.8. DISEÑO DE LA BASE DE DATOS

Una base de datos es un "almacén" que permite guardar grandes cantidades de información de forma organizada para que luego se pueda encontrar y utilizar fácilmente.

Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más **columnas** y **filas**. Las columnas guardan una parte de la información sobre cada elemento que queramos guardar en la tabla, cada fila de la tabla conforma un registro.

En la Figura. 7.19, se puede observar la cardinalidad de las relaciones entre las tablas que se crearon en la base de datos de la Distribuidora.

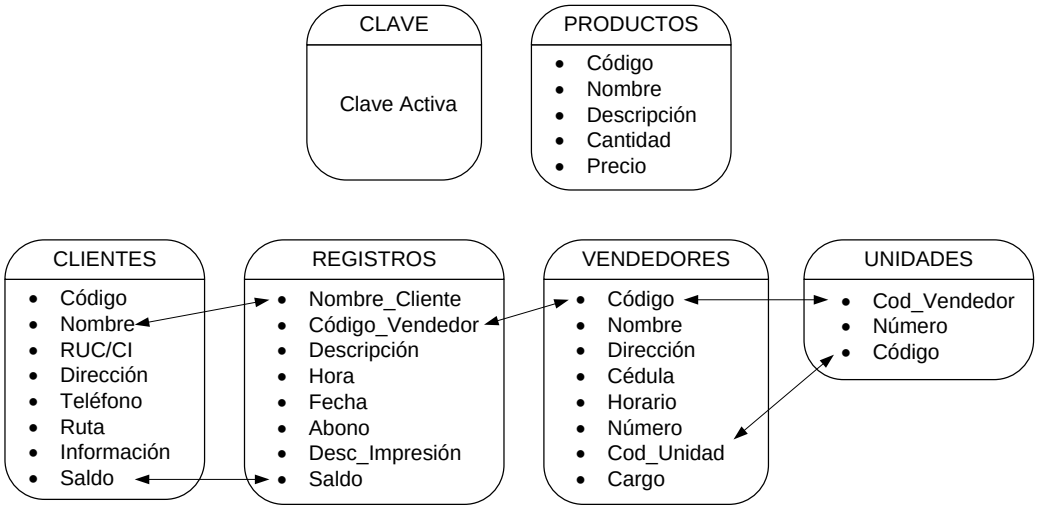


Figura. 7.19. Tablas de la Base de Datos de la Distribuidora.

7.8.1. Elección de la Base de Datos

La base de datos seleccionada para almacenar la información que se manipulará en Visual Basic es Microsoft Access.

Microsoft Access es un programa Sistema de gestión de base de datos relacional creado y modificado por Microsoft para uso personal de pequeñas organizaciones. Es un componente de la suite Microsoft Office aunque no se incluye en el paquete "básico".

Es un software de gran difusión entre pequeñas empresas cuyas bases de datos no requieren de excesiva potencia, ya que se integra perfectamente con el resto de aplicaciones de Microsoft y permite crear pequeñas aplicaciones con unos pocos conocimientos del Programa.

Microsoft Access permite crear formularios para insertar y modificar datos fácilmente. También tiene un entorno gráfico para ver las relaciones entre las diferentes tablas de la base de datos.

Tiene un sistema de seguridad de cifrado bastante primitivo y puede ser la respuesta a proyectos de programación de pequeños y medianos tamaños.

7.8.2. Tabla de Clave

Está tabla permite almacenar la clave para acceder al modo de administrador explicado en la sección 7.3.7.

En la Figura. 7.20 se puede observar los campos de la Tabla Clave de la base de datos y el tipo de campo que se está utilizando.

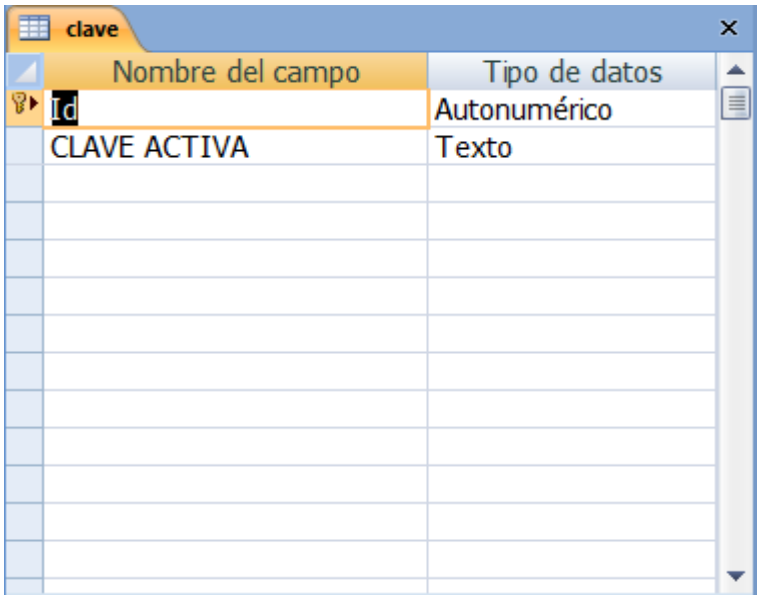


Figura. 7.20. Tabla de Clave.

En la Figura. 7.21 se puede observar los registros de la tabla Clave, en esta tabla se tiene únicamente un registro para almacenar la clave que se encuentra activa.

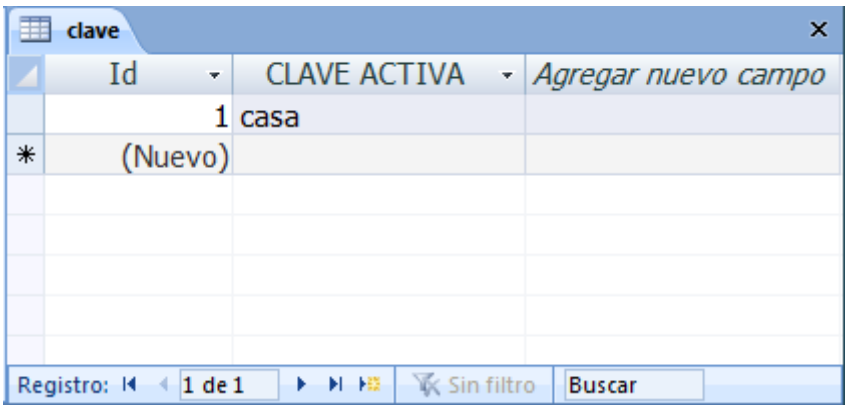


Figura. 7.21. Registros de la Tabla Cliente.

7.8.3. Tabla de Clientes

Está tabla permite almacenar la información correspondiente a los Clientes, en la Figura. 7.22**¡Error! No se encuentra el origen de la referencia.** se puede observar los campos de la Tabla Clientes de la base de datos y el tipo de campo que se está utilizando.

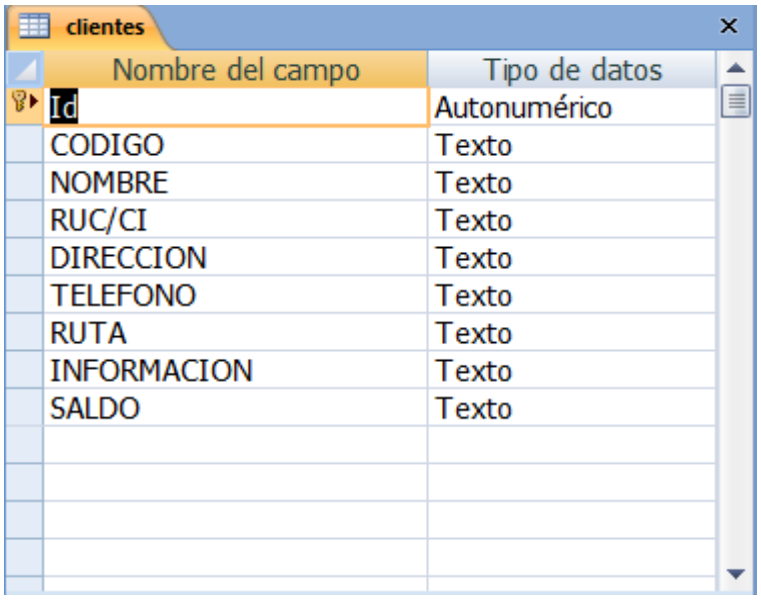


Figura. 7.22. Tabla de Clientes.

En la Figura. 7.23 se puede observar los registros de la tabla Clientes.

clientes

Id	CODIGO	NOMBRE	RUC/CI	DIRECCION	TELEFONO	RUTA	INFORMACION	SALDO
1	A008	diego niama	0603449463001	norte	2222222	RUTA 1	congelador	0
2	C003	cristian karolys	0502498074001	10 de agosto	2333333	RUTA 3	no paga	0
3	B001	cristina escudero	1502498074001	12 de octubre	2444444	RUTA 2	debe	0
4	A002	carlos merchan	2502498074001	6 de diciembre	2555555	RUTA 1	no data	0
5	A005	cesar niama	3502498074001	los pinos	2666666	RUTA 1	no data	0
6	D010	david puga	4598737348001	carolina	2777777	RUTA 4	no data	134
7	A006	julio larco	5502498074001	amazonas	2888888	RUTA 1	no data	55.5
8	A001	carlos	6502498074001	shyris	2999999	RUTA 1	no data	1725
9	D001	jual lopez	7502498074001	la colina	2111111	RUTA 4	no data	49
10	A007	polo valencia	8502498074001	cesar davila	3222222	RUTA 1	no data	0
11	A004	paul canizares	5432498074001	algarrobos	3444444	RUTA 1	no data	0
12	A003	pedro larea	4738463756001	amazonas	123423232	RUTA 1	no data	4
13	B002	julio larco	123456789	la espe	23232323	RUTA 2	no data	0
14	A009	juan pueblo	0587898764	Granados	123456733	RUTA 1	no data	0
15	A010	federico intriafo	123234324	la kenedy	no data	RUTA 1	no data	0

Registro: 12 de 15 Sin filtro Buscar

Figura. 7.23. Registros de la tabla Cliente.

7.8.4. Tabla de Productos

Está tabla permite almacenar la información correspondiente a los Productos, en la Figura. 7.24 **¡Error! No se encuentra el origen de la referencia.** se puede observar los campos de la Tabla Productos de la base de datos y el tipo de campo que se está utilizando.

productos

Nombre del campo	Tipo de datos
Id	Autonumérico
CODIGO	Texto
NOMBRE	Texto
DESCRIPCION	Texto
CANTIDAD	Texto
PRECIO	Texto

Figura. 7.24. Tabla de Productos.

En la Figura. 7.25 se puede observar los registros de la tabla Productos.

productos

Id	CODIGO	NOMBRE	DESCRIPCI	CANTIDAD	PRECIO
1	P001	Baloncito	a	914	1
39	P002	Topsyto	crema	948	2.50
40	P003	Alfajor	galleta	994	1
41	P004	Vaso	vainilla	993	5
42	P005	Galletopsy	frutilla	976	2
44	P006	Junior	vainilla	980	4
45	P007	Picoleta	tres sabores	995	2
46	P008	Clonito	agua	994	1
47	P009	Chiqui Agua	agua	987	1
48	P010	Sundae	mora	996	10
49	P011	Chiqui choc	chocolate	998	12
50	P012	Krococono	cono	985	3
51	P013	Dona	ron pasa	996	1.50
52	P014	Buggy Gum	chicle	999	1
58	P015	Mani boom	chocolate	992	1

Registro: 1 de 15 Sin filtro Buscar

Figura. 7.25. Registros de la Tabla Productos.

7.8.5. Tabla de Registros

Está tabla permite almacenar la información correspondiente a los Registros de venta, en la Figura. 7.26 **¡Error! No se encuentra el origen de la referencia.**se puede observar los campos de la Tabla de Registros de la base de datos y el tipo de campo que se está utilizando.

registros

Nombre del campo	Tipo de datos
Id	Autonumérico
NOMBRE_CLIENTE	Texto
CODIGO_VENDEDOR	Texto
DESCRIPCION	Texto
HORA	Texto
FECHA	Texto
SALDO	Número
ABONO	Número
DESC_IMPRESION	Texto

Figura. 7.26. Tabla de Registros.

En la Figura. 7.27 se puede observar los registros de la tabla Registros.

registros

NOMBRE_CLIENTE	CODIGO_VENDEDOR	DESCRIPCION	HORA	FECHA	ABONO
pedro larea	V004	, 04 Baloncito	20:34:29	09/06/12	0
carlos	V004	, 05 Baloncito	20:37:25	09/06/12	0
diego niama	V004	, 08 Baloncito	20:43:35	09/06/12	0
carlos	V004	, 02 Topsyto, 01 Junior, 10 Krococono	13:36:28	09/06/15	0
carlos	V004	, 01 Baloncito, 04 Galletopsy, 01 Clonito	13:44:26	09/06/15	0
carlos	V004	, 01 Baloncito	14:49:39	09/06/15	0
carlos	V004	, 01 Baloncito, 03 Chiqui Agua	10:58:15	09/06/16	0
carlos	V004	, 01 Alfajor, 03 Junior	11:06:57	09/06/16	0
carlos	V004	, 05 Baloncito, 03 Junior	11:12:59	09/06/16	0
carlos	V004	, 05 Baloncito, 06 Topsyto	11:37:50	09/06/16	0
carlos	V004	, 01 Baloncito	11:47:38	09/06/16	0
diego niama	disponible	, 01 Baloncito, 03 Topsyto, 02 Alfajor, 01 Vaso, 03 Galletopsy, 04 Junior	12:14:55	09/06/16	0

Registro: 40 de 49 Sin filtro Buscar

Figura. 7.27. Registros de la Tabla de Registros.

7.8.6. Tabla de Unidades

Está tabla permite almacenar la información correspondiente a las unidades móviles, en la Figura. 7.28 **¡Error! No se encuentra el origen de la referencia.** se puede observar los campos de la Tabla de Unidades de la base de datos y el tipo de campo que se está utilizando.

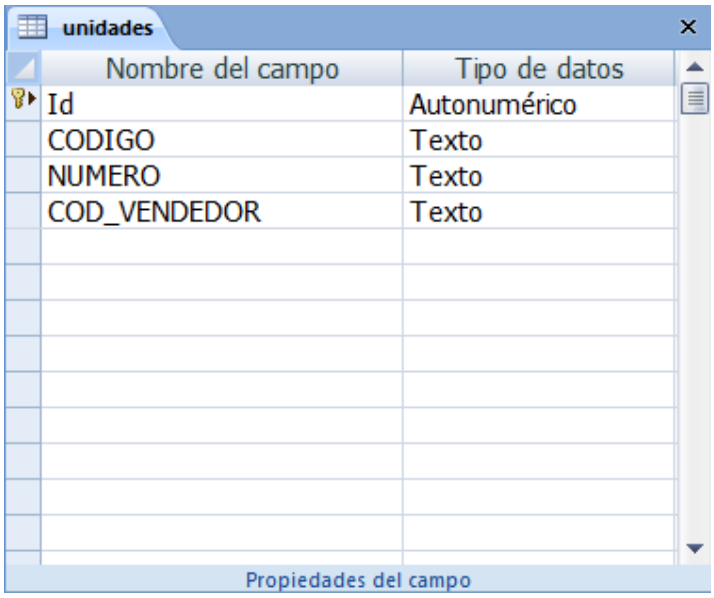


Figura. 7.28. Tabla de Unidades.

En la Figura. 7.29 se puede observar los registros de la tabla Unidades.

Figura. 7.29. Registros de la Tabla Unidades.

7.8.7. Tabla de Vendedores

Está tabla permite almacenar la información correspondiente a los trabajadores de la distribuidora, en la Figura. 7.30 **¡Error! No se encuentra el origen de la referencia.**se puede observar los campos de la Tabla de Registros de la base de datos y el tipo de campo que se está utilizando.

vendedores

Nombre del campo	Tipo de datos
Id	Autonumérico
CODIGO	Texto
NOMBRE	Texto
DIRECCION	Texto
CEDULA	Texto
HORARIO	Texto
NUMERO	Texto
COD_UNIDAD	Texto
CARGO	Texto

Figura. 7.30. Tabla de Vendedores.

En la Figura. 7.31 se puede observar los registros de la tabla Unidades.

vendedores

Id	CODIGO	NOMBRE	DIRECCIOI	CEDULA	HORARIO	NUMERO	COD_UNID	CARGO
1	V001	CARLOS	norte	0502475694	10-2	1234567	U007	RUTA 1
5	V002	Juan valdez	este	4353546346	15-19	2080817	U002	RUTA 1
6	V004	pedro	la colina	1793849347	6-18	2343435	U004	RUTA 1
7	V005	juan piguavi	no data	no data	10-6	no data	Ninguna	RUTA 1
11	V008	federico	no data	2342343242	no data	no data	Ninguna	RUTA 1
13	V003	juan perez	quito	05038594001	10-18	2508333	U001	RUTA 1
16	V011	gabriel cerda	latacunga	0603449462	13-14	2856789	U006	RUTA 1

Registro: 1 de 7 Sin filtro Buscar

Figura. 7.31. Registros de la Tabla Unidades.

7.9. CONEXIÓN DE LA BASE DE DATOS CON LA APLICACIÓN

1) Primero se incluye el componente "Microsoft ADO data control 6.0 (OLEDB)". Se puede hacer pulsando Ctrl.-T o en el menú Proyecto → Componentes.

2) Se coloca en el formulario el control y pulsamos F4 para ver sus propiedades.

3) Se hace Click en la propiedad ConnectionString y aparecerá el botón de puntos suspensivos. Se hace click en el botón. En la Figura. 7.32 se puede observar la propiedad del control ADO.

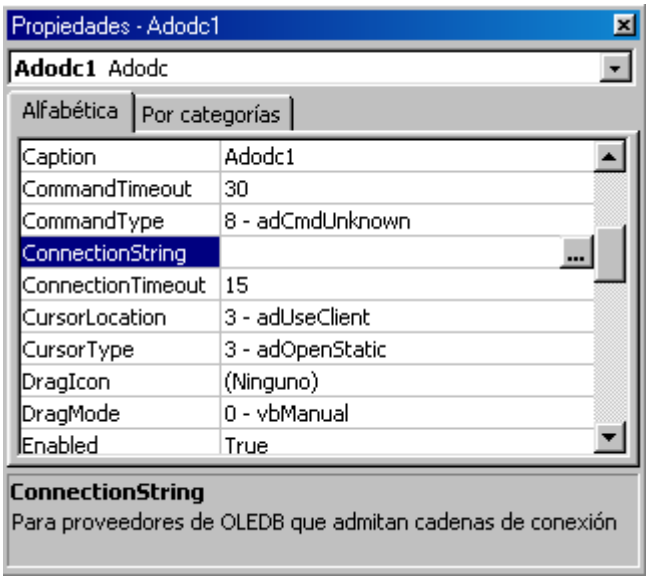


Figura. 7.32. Propiedades del control ADO.

4) Ahora aparece el asistente que nos mostrará 3 opciones. Utilizar un DSN de archivo, utilizar un controlador ODBC o utilizar una cadena de conexión. En todos los casos, a la derecha de la opción hay una opción que permite seleccionar o generar el origen de los datos. Usar la cadena de conexión (opción por defecto) y pulsar el botón generar. En la Figura. 7.33 se puede observar el asiste de conexiones del control ADO.

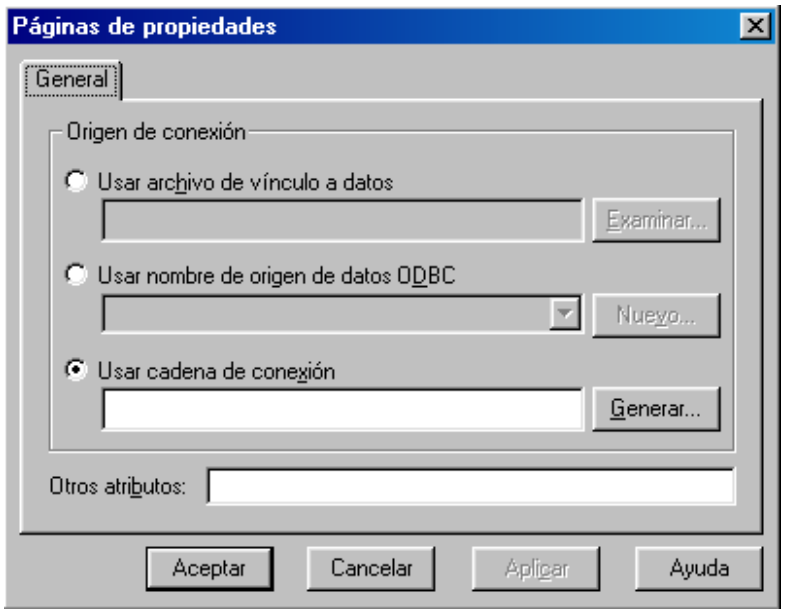


Figura. 7.33. Asistente de conexión.

5) Al pulsar el botón generar, aparece otra ventana en la que se tiene cuatro pestañas aunque únicamente se necesitan dos de ellas para crear una cadena de conexión correcta y probada. Se selecciona el proveedor de datos que se desea utilizar y se pulsa el botón siguiente para pasar a la siguiente pestaña.

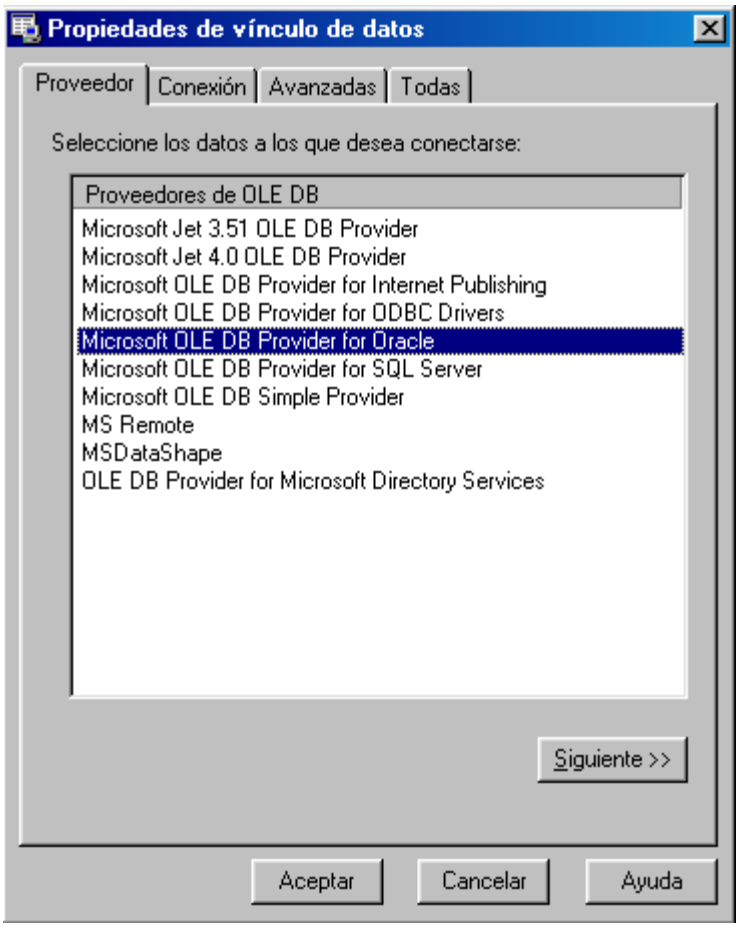


Figura. 7.34. Selección de proveedor de conexión.

6) En la Pestaña conexión, se tiene que proporcionar los datos correspondientes al proveedor de la base de datos a la que se quiere conectar. Para conectar con una base de datos de tipo MS Access (MS Jet 4.0 OLEDB Provider para la versión 2000 de Access, MS Jet 3.5 OLEDB Provider para la versión 97 de Access), tenemos que decirle el nombre y path de la BD. Para ello se puede utilizar el botón de los puntos suspensivos, que abrirá una ventana que permitirá seleccionar el archivo .mdb. Si se utiliza el proveedor de datos de **Oracle**, se tendrá que indicar como nombre de servidor la cadena de conexión utilizada en el SQL Net o en el Net8 de Oracle (la misma cadena que especifica cuando se conecta a través de SQL*Plus).

   En esta pestaña existen dos opciones interesantes:

   - "Contraseña en Blanco", si la marcamos no permitirá teclear la contraseña.
   - "Permitir guardar la contraseña", dependiendo de si está o no seleccionada, incluirá o no en la cadena de conexión la clave del usuario.

   Después de haber dado esta información, se pulsa el botón Probar conexión. En el caso de que haya algún error, el asistente nos lo indicará con un mensaje. En la Figura. 7.35

se puede observar la conexión fallida y en la Figura. 7.36 se puede observar el mensaje que se muestra cuando la conexión ha sido exitosa.

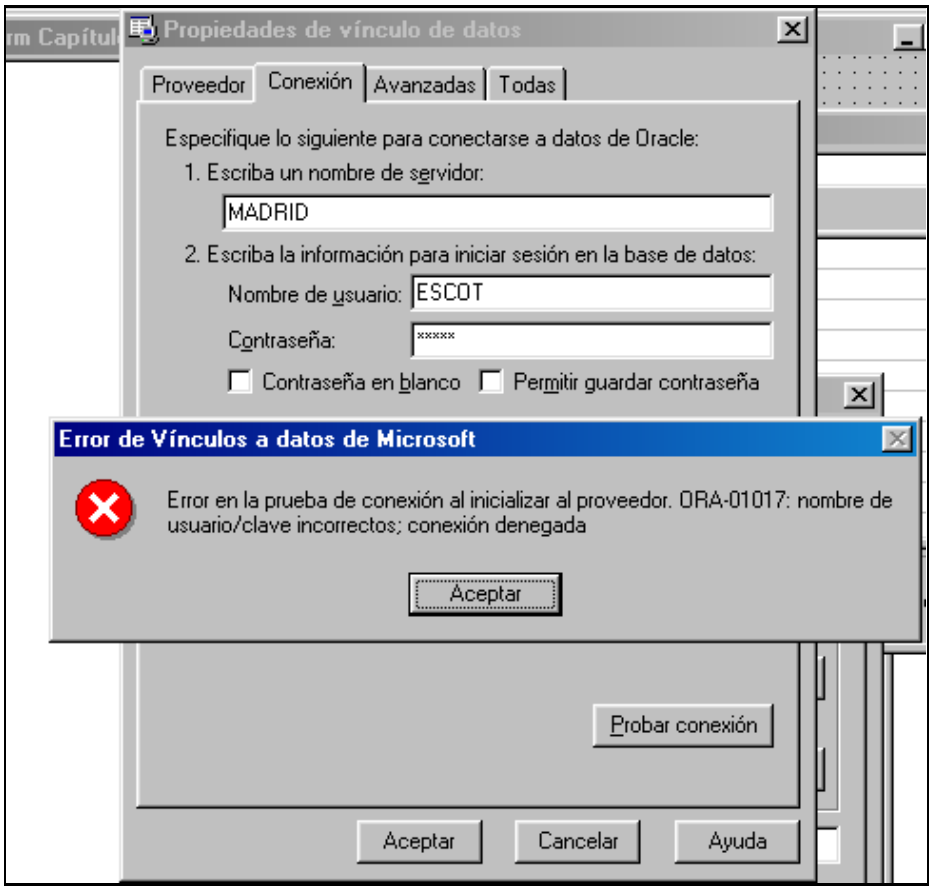


Figura. 7.35. Prueba de conexión (fallida) con Proveedor de datos de Oracle.

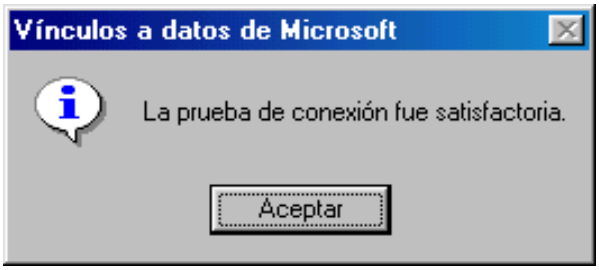


Figura. 7.36. Prueba exitosa.

7.10. INSTALACIÓN DE SOFTWARE EN LA PC

Para la instalación de la aplicación HMI desarrollada en Visual Basic se utilizo el generador de proyectos de Visual Basic, que se encuentra en la pestaña Archivo de Visual Basic, este generador crea la extensión .exe de la aplicación. En la Figura. 7.37 se puede observar el generador de aplicaciones de Visual Basic.

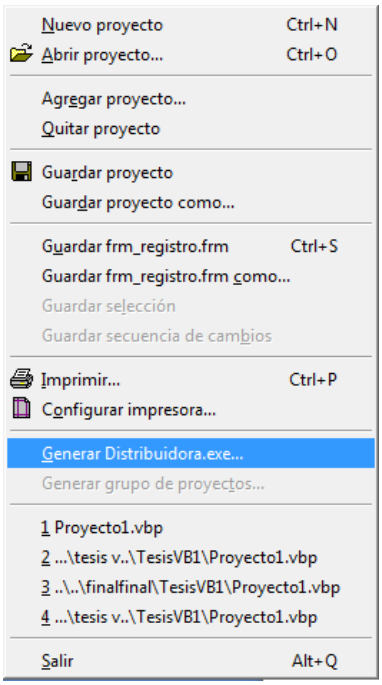


Figura. 7.37. Generador de Proyectos.

Una vez que se hace clic en el generador se solicita el nombre que se desea dar a la aplicación.

CAPÍTULO 8

INTERFAZ J2ME PARA EL DISPOSITIVO MÓVIL

8.1. DISEÑO DE LA APLICACIÓN HMI.

La aplicación para el dispositivo móvil es desarrollada para facilitar la elaboración de pedidos y poder tareas como:

- Codificar el pedido ya que al utilizar SMS de GSM el tamaño máximo de caracteres que se puede enviares 160 por lo que es necesario enviar codificado en el mensaje de pedido.
- Manejar una base de datos de todos los productos existentes en la distribuidora
- Almacenar los pedidos en la memoria del dispositivo móvil.
- Mantener la información segura bloqueado el ingreso a los datos.
- Actualizar la información de los productos.
- No congestionar el buzón de entrada del dispositivo móvil.
- Decodificar información enviada por la distribuidora.

8.1.1. Descripción

La interfaz para el dispositivo móvil es una aplicación desarrollada en Netbeans sobre la plataforma J2ME el cual tiene una base de datos de los productos que posee la distribuidora con su respectivo costo.

La aplicación se encarga de codificar el pedido el que a través de los datos ingresados en un form en el cual consta de una tabla con los productos de la distribuidora.

Al ya tener el mensaje codificado la aplicación procede a enviar la solicitud del pedido a la distribuidora que posterior al análisis del pedido responderá a la aplicación J2ME el cual le indica al usuario la respuesta de la distribuidora; al no recibir una respuesta la aplicación puede reenviar el pedido o almacenarlo en la memoria del dispositivo móvil para posteriormente ser procesada.

8.1.2. Diagrama de Flujo

La secuencia que sigue la aplicación se describe en el diagrama de flujo que se muestra en las Figuras 8.1, 8.2 y 8.3.

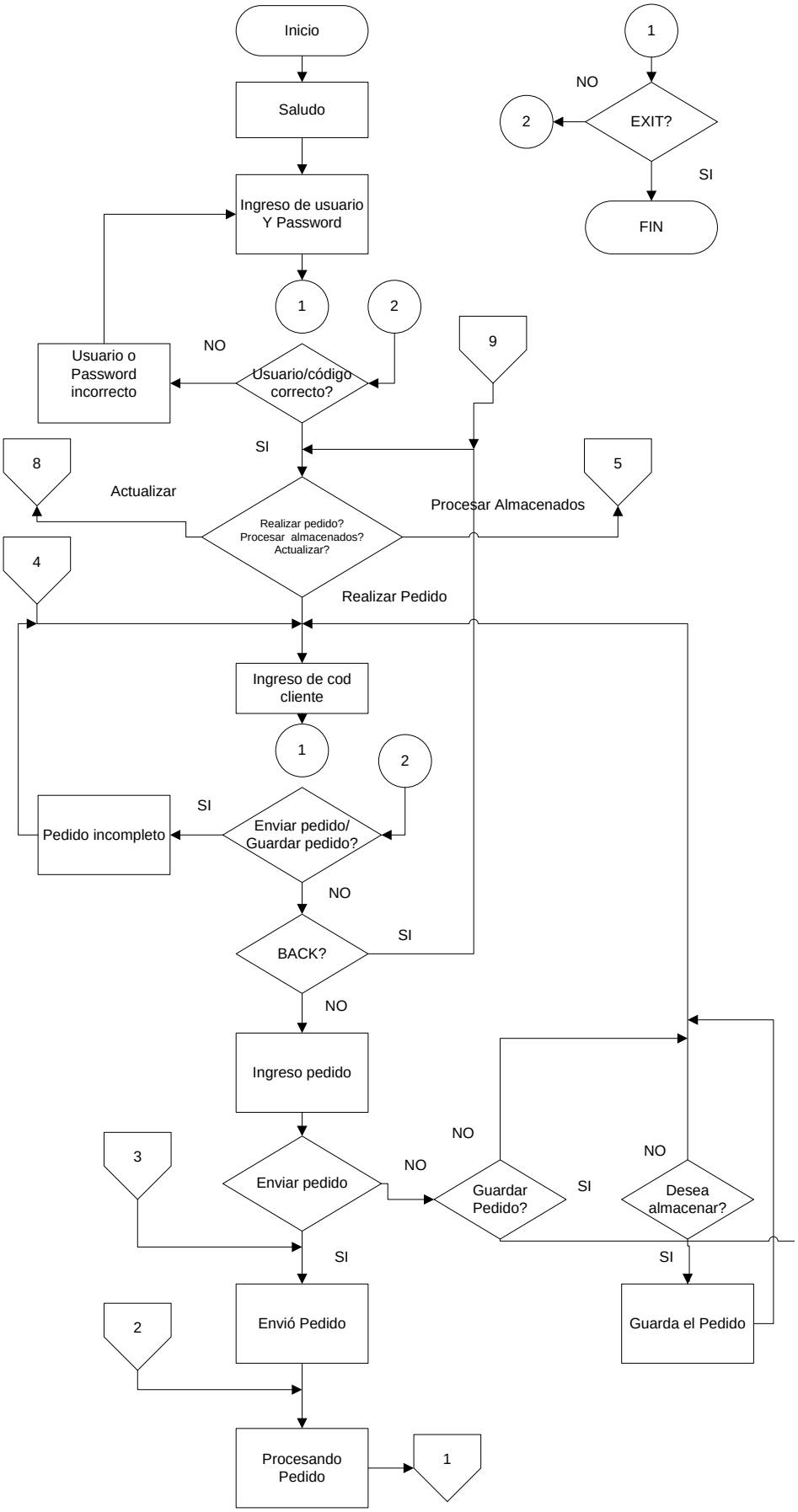


Figura. 8.1. Diagrama de flujo de la aplicación Java (I).

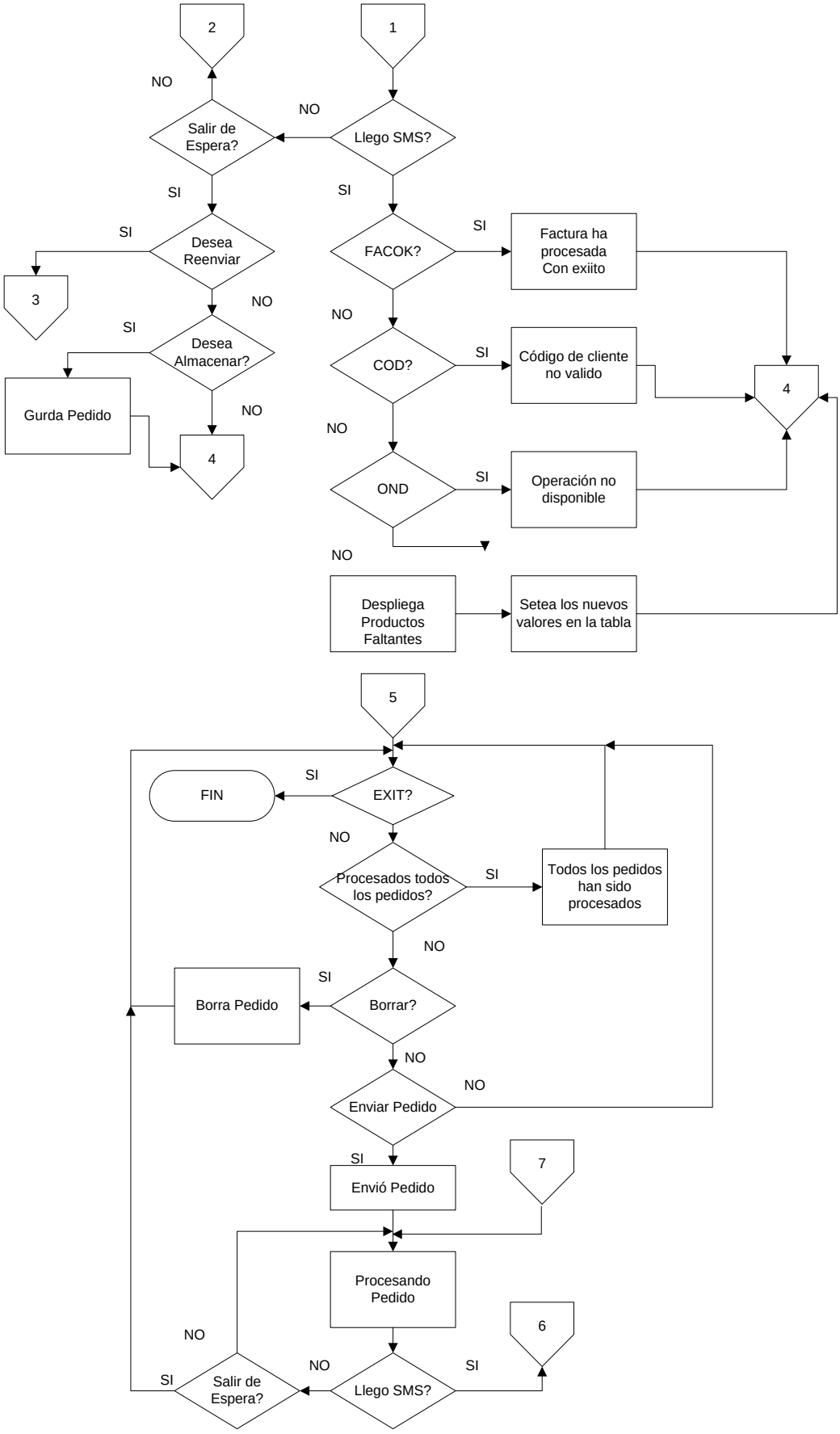


Figura. 8.2. Diagrama de flujo de la aplicación Java (II).

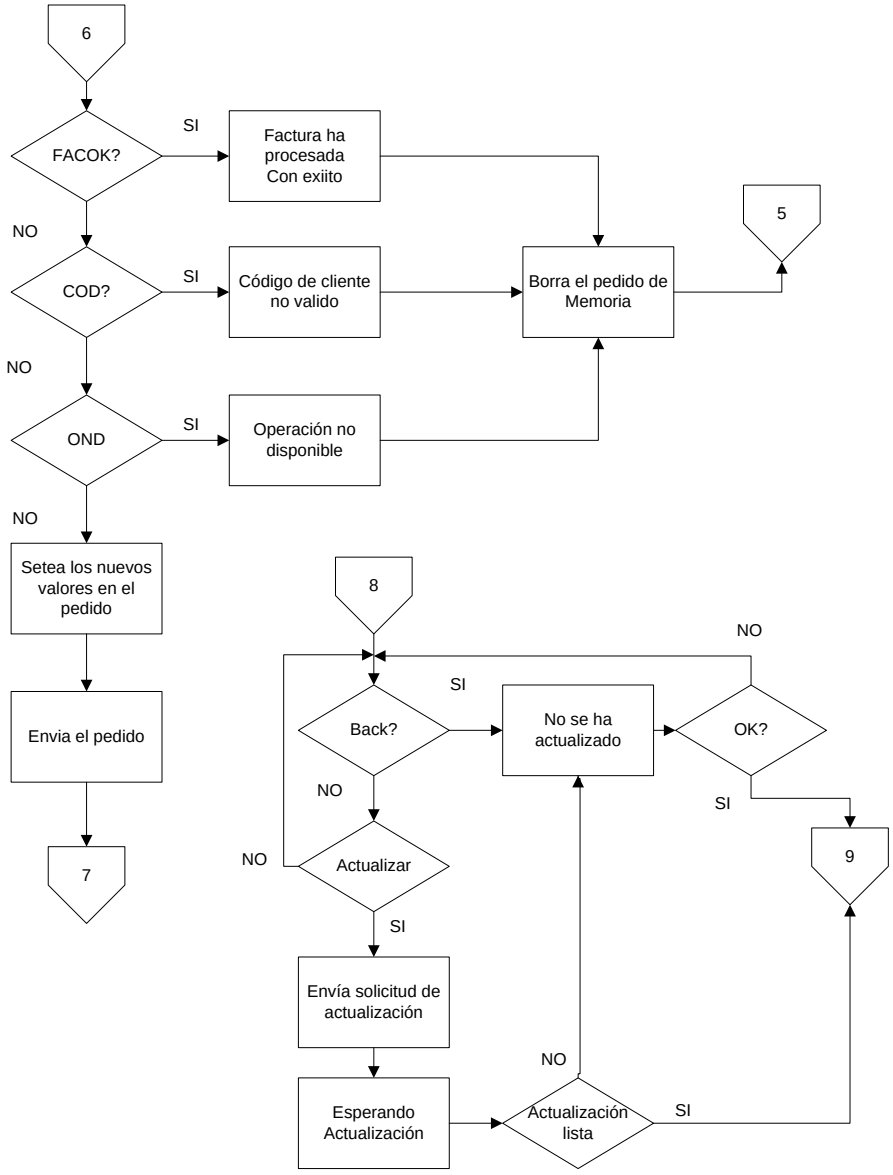


Figura. 8.3. Diagrama de flujo de la aplicación Java (III).

8.1.3. Pantalla de Saludo (SplashScreen)

Simplemente es una pantalla de bienvenida para mejorar el aspecto de la aplicación. Puesto que se ofrece a los usuarios la primera impresión de la aplicación.

Figura. 8.4. Pantalla de Saludo.

8.1.4. Pantalla de Inicio de Sesión (LoginScreen)

La pantalla de inicio de sesión constituye un valioso componente de interfaz de usuario con elementos tales como nombre de usuario, Password y botón de acceso.

Se vio necesario realizar esta pantalla para evitar que personas no autorizadas manipulen la aplicación o realicen pedidos.

Figura. 8.5. Pantalla de Inicio de Sesión.

8.1.5. Pantalla de aviso Error de Usuario o Password (alert)

Es un mensaje de error indicando que el usuario y/o la clave ingresada es incorrecta.

Figura. 8.6. Pantalla de aviso Error de Usuario o Password.

8.1.6. Pantalla de Menú de Operaciones de Usuario(Form 1)

En esta Pantalla se presenta un menú (choiceGroup) en el cual están las tres tareas que puede realizar la aplicación.

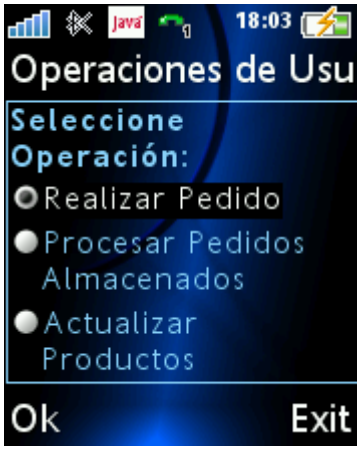


Figura. 8.7. Pantalla de Menú de Operaciones de Usuario.

8.1.7. Pantalla de Ingreso de Pedidos (Form)

Es la pantalla principal de trabaja en la cual se ingresa el código del cliente en un TextField y la cantidad del producto que se va a pedir; indicada en un TableModel como se indico anteriormente es una tabla.

En este Form consta de cuatro Botones que son:

- Exit para salir de la aplicación.

- Back para regresar a la Pantalla de Menú de Operaciones de Usuario.
- Enviar pedido para enviar el mensaje del pedido a la distribuidora.
- Guardar pedido Para guardar el pedido en la memoria del dispositivo móvil cuando no posea señal.

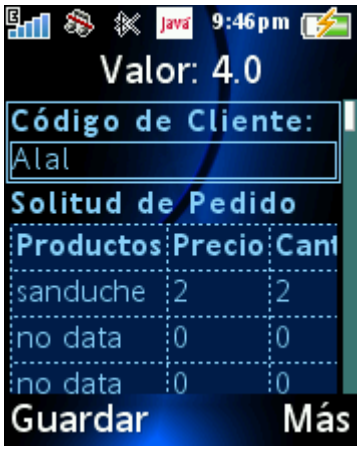


Figura. 8.8. Pantalla de Ingreso de Pedidos.

Figura. 8.9. Botones de Pantalla de Ingreso de Pedidos.

8.1.8. Pantalla de aviso de Código o productos incompletos(alert5)

Es un aviso en el cual indica que algún campo esta incompleto esto es para que no produzca un error en el pedido.

Figura. 8.10. Pantalla de aviso de Código o productos incompletos.

8.1.9. Pantalla de Procesando Pedidos(alert1)

Esta pantalla espera las diferentes respuestas de la distribuidora para posteriormente indicar visualmente cual es la notificación y tomar su respectivo proceso.

Figura. 8.11. Pantalla de Procesando Pedidos.

La distribuidora envía cuatro diferentes respuestas:

a.) FACOK.

Cuando la factura en la distribuidora ha sido procesada con éxito teniendo en cuenta que debe cumplir las condiciones descritas en el CAPITULO 6 como son Código de cliente correcto, Registrado el número de la unidad móvil y que todos los productos pedidos existan en Stock.

Posterior al aviso se reinician todos los valores de la Pantalla de Ingreso de Pedidos.

Figura. 8.12. Pantalla de Respuesta de Pedido FACOK.

b.) COD.

Al enviar el pedido con un código erróneo, el cual no conste en la base de datos de en la distribuidora.

Posterior al aviso se regresa a la Pantalla de Ingreso de Pedidos sin modificar ningún valor.

Figura. 8.13. Pantalla de Respuesta de Pedido COD.

c.) OND.

Cuando en la base de datos de la distribuidora no consta el número de el que se realiza el pedido.

Posterior al aviso se reinician todos los valores de la Pantalla de Ingreso de Pedidos.

Figura. 8.14. Pantalla de Respuesta de Pedido OND.

d.) Factura incompleta.

La distribuidora al no existir productos suficientes para completar el pedido le informa a la aplicación en el dispositivo móvil cuales son los productos que están incompletos con la cantidad que esta disponible para posteriormente setear los valores en la tabla de productos para poder realizar el pedido disponible.

Figura. 8.15. Pantalla de Respuesta de Pedido Factura incompleta.

8.1.10. Pantalla de Reenvío de Pedido(Form 2)

Cuando la aplicación no ha recibido ninguna notificación de la distribuidora y el usuario sale de la espera de notificación que es la pantalla de Procesando Pedidos, la aplicación nos indica un form con la pregunta si desea reenviar el pedido para procesar de nuevo el pedido.

Figura. 8.16. Pantalla de Reenvío de Pedido

8.1.11. Pantalla de Almacenamiento de Pedido(alert6)

Después de pregunta si desea reenviar el mensaje y al seleccionar "NO"; aparecerá una pantalla preguntando si desea almacenar el pedido en la memoria para posteriormente ser procesada.

Figura. 8.17. Pantalla de Almacenamiento de Pedido.

8.1.12. Pantalla de Procesamiento de Pedidos de Almacenados (Form 3)

En esta pantalla se va indicando uno a uno los pedidos que están almacenados en la memoria del dispositivo móvil.

Para procesar el pedido almacenado tiene el botón de enviar pedido que esta almacenado en un registro que trabaja como pila. Posteriormente la aplicación va a la

pantalla Procesando Pedidos la cual tiene el funcionamiento ya indicado anteriormente. Al recibir el mensaje va borrando uno a uno los pedidos procesados.

Figura. 8.18. Pantalla de Procesamiento de Pedidos de Almacenados.

8.1.13. Pantalla de aviso de no procesados todos los Pedidos Almacenados(alert4)

La Pantalla de Procesamiento de Pedidos de Almacenados tiene un botón Back el cual al ser seleccionado y al tener almacenados pedidos se despliega la pantalla de no procesados todos los Pedidos Almacenados dándonos la opción de retornar a procesar los pedidos almacenados o salir a al menú de operaciones.

Figura. 8.19. Pantalla de aviso de no procesados todos los Pedidos Almacenados.

8.1.14. Pantalla de Actualización de productos(Form 4)

Esta Pantalla contiene dos botones el de back para poder regresar y el de actualizar para enviar el pedido de actualización a la distribuidora y se activa la pantalla de Procesando información .

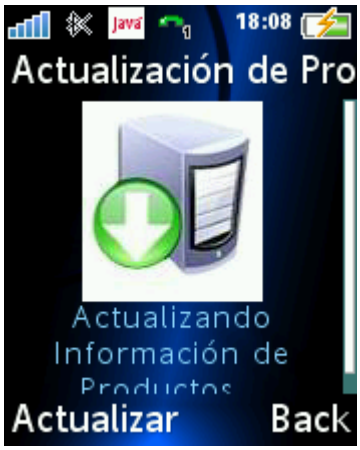


Figura. 8.20. Pantalla de Actualización de productos.

8.1.15. Pantalla de Procesando Información (alert3)

La aplicación entra en espera de la respuesta de la distribuidora en la que envía el número de actualización y de actualización y la lista de productos con su respectivo precio que posteriormente serán almacenados en un registro en la memoria del dispositivo móvil.

Figura. 8.21. Pantalla de Procesando Información.

Figura. 8.22. Pantalla de respuesta de Procesando Información.

8.1.16. Pantalla de aviso de no Actualizada toda la información (alert7)

Al recibir las actualizaciones este mensaje contiene el numero total de actualizaciones la cual nos indica cuantas veces es necesario pedir la actualización de no ser completadas todas las actualizaciones y presionar el botón back en la pantalla de Actualización de productos se despliega la pantalla de aviso de no Actualizada toda la información que quiere decir que no fueron procesadas todas las actualizaciones, esta pantalla contiene dos botones que nos permiten regresar a seguir actualizando la información o ir al menú de operaciones

Figura. 8.23. Pantalla de aviso de no Actualizada toda la información.

8.2. HARDWARE (SmartPhone o celular).

El requisito mínimo para ejecutar la aplicación es que el dispositivo móvil soporte la nueva versión de la tecnología Java™ (MIDP 2.0, CLDC 1.0) o superior, y las librerías

Wireless Messaging API (JSR 120 o JSR 205). La mayoría de terminales que se venden hoy en día cumplen ya con estos requisitos.

La lista de dispositivos que soportan J2ME es muy larga y variada; sin embargo, siendo una tecnología de reciente creación aún pueden ser enumerados en una lista. A continuación se presenta una tabla con los dispositivos que soportan esta tecnología.

8.2.1. Nokia

Figura. 8.24. Nokia E63.

Nokia 2855	Nokia 3152	Nokia 3155
Nokia 3155i	Nokia 3220	Nokia 3230*
Nokia 3250	Nokia 3600	Nokia 5000
Nokia 5140	Nokia 5140i	Nokia 5220
Nokia 5310	Nokia 5320	Nokia 5500
Nokia 5610	Nokia 5800	Nokia 6020
Nokia 6021	Nokia 6060	Nokia 6070
Nokia 6101	Nokia 6102	Nokia 6103
Nokia 6111	Nokia 6120	Nokia 6125
Nokia 6126	Nokia 6131	Nokia 6136
Nokia 6152	Nokia 6155	Nokia 6155i
Nokia 6165	Nokia 6170	Nokia 6230
Nokia 6230i	Nokia 6233	Nokia 6234
Nokia 6235	Nokia 6235i	Nokia 6255
Nokia 6260*	Nokia 6265	Nokia 6265i
Nokia 6270	Nokia 6280	Nokia 6282
Nokia 6600*	Nokia 6620*	Nokia 6630
Nokia 6670*	Nokia 6680	Nokia 6681
Nokia 6682	Nokia 6822	Nokia 7260
Nokia 7270	Nokia 7360	Nokia 7370
Nokia 7610*	Nokia 7700*	Nokia 7710
Nokia 8800	Nokia 8801	Nokia 9300
Nokia 9300i	Nokia 9500	Nokia E50
Nokia E51	Nokia E60	Nokia E61

Nokia E63	Nokia E70	Nokia N70
Nokia N71	Nokia N72	Nokia N73
Nokia N80	Nokia N90	Nokia N91
Nokia N92	Nokia N93	Nokia N95

Tabla. 8. 1 Teléfonos Nokia que soportan Java.

8.2.2. Siemens

Figura. 8.25. Siemens C 65.

C65	C66	C70
C72	C75	C81
CF75	CF76	CFX65
CL75	CX65	CX70
CX70 EMOTY	CX75	EF51
EF91	EF81	EL71
M65	M75	ME75
P51	S65	S66
S68	S6C	S6V
S75	S81	SFG75
SK65	SL65	SL75
SP65	SXG75	

Tabla. 8.2. Teléfonos Siemens que soportan Java.

8.2.3. Sony Ericsson

Figura. 8.26. SonyEricsson 580

D750i	F500	J300a
J300c	J300i	K300a
K300c	K300i	K310a
K310c	K310i	K500
K506	K508	K510a
K510c	K550	K600
K608i	K610c	K610i
K700	K700c	K700i
K750i	K750c	K790a
K790c	K790i	K800c
K800i	M600i	M608c
T303	P910a	P910c
P910i	P990c	P990i
R306	S600	S700
S700c	S700i	S710
S710a	SO506i	V600i
V800	V802SE	W200
W300i	W300c	W303
W380	W550c	W550i
W580i	W600	W700c
W700i	W710c	W710i
W760	W800	W810c
W810c	W850i	W890
W900c	W900i	W950c
W950i	W980	Z1010
Z500	Z500a	Z500i
Z520a	Z520c	Z520i
Z525a	Z530c	Z530i
Z550c	Z550i	Z710c
Z710i	Z800	

Tabla. 8.3 Teléfonos SonyEricsson que soportan Java.

8.2.4. Motorola

Figura. 8.27. Motorola w510.

A388	Accompli 009	A1200
T280i	T720	V60i
V66i	A820	W510
W5	W375	Z6
V8		

Tabla. 8. 4 Teléfonos Motorola que soportan Java

8.2.5. Samsung

Figura. 8.28. Samsung E215.

E215	F250L
M310	SGH-S100

Tabla. 8. 5 Teléfonos Samsung que soportan Java

8.2.6. LG

Figura. 8.29. LG SHINE SLIM.

CITRINE	RUBY	SECRET
SHINE SLIM	VESTA	VENUS
VIEW TY	LX5350	

Tabla. 8. 6 Teléfonos LG que soportan Java.

8.3. INSTALACIÓN DEL SOFTWARE EN EL DISPOSITIVO MÓVIL.

La instalación de la aplicación en dispositivos móviles es muy sencilla puede ser mediante conexión a la PC ya sea mediante el cable USB o a través del Bluetooth simplemente se copia al dispositivo móvil el archivo .jar que es creada por Netbeans y posteriormente se instala en el dispositivo ejecutando el archivo.

Al ya tener el Archivo .jar en el dispositivo móvil puede ser enviada por Bluetooth a otros dispositivos instalándose automáticamente.

8.4. ENVIÓ Y RECEPCIÓN DE MENSAJES GSM.

Para poder realizar el pedido es necezaria la comunicación entre la distribuidora y la aplicación por lo que la aplicación debe enviar el pedido y recibir la respuesta de la distribuidora mediante SMS.

8.4.1. Envío del Pedido

La parte fundamental de la aplicación es el envío del pedido a través de un SMS el cual es codificado para poder enviar la mayor cantidad de información.

La codificación es realizada mediante la ubicación del producto en la tabla es decir cada ubicación tiene su código el cual inicia en A siendo el primer producto y según vaya incrementando la posición incrementa alfabéticamente el código, iniciando con mayúsculas y posteriormente con minúsculas. A continuación se indica como conforma la codificación del mensaje

```
for (i=1 ; i<=num_filas-1 ; i++)
     { //se extrae el valor de cada producto en toda la tabla
        producto[i]=Integer.parseInt(tableModel1.getValue(2, i-1).toString());
        if (producto[i] != 0)
        {
           cod=i+64;
           if (cod>90)
           {
```

```
                cod=cod+6;
            }
            cod_producto=(char)cod;
            longitud=tableModel1.getValue(2, i-1).toString().length();

            if (longitud==1)
            {
           sms_enviar=sms_enviar+cod_producto+"0"+Integer.toString(producto[i]);
            }
            else
            {
                sms_enviar=sms_enviar+cod_producto+Integer.toString(producto[i]);
            }
        }
    }.
```

Una vez realizada la codificación del pedido se complementa colocando el código del cliente y los caracteres de inicio y parada para finalmente envía el pedido a la distribuidora.

```
sms_enviar="^"+nombre_cliente+sms_enviar+"~";
String num_distribuidora=numero_distribuidora;
MessageConnection conn=(MessageConnection)Connector.open(num_distribuidora);
TextMessage msg1=(TextMessage)
            conn.newMessage(MessageConnection.TEXT_MESSAGE);
msg1.setPayloadText(sms_enviar);
conn.send(msg1);
conn.close();
```

De igual manera funciona el envío para la actualización y para el envío de pedidos almacenados.

8.4.2. Recepción de respuesta de la distribuidora

Para realizar la recepción de los SMS que envía a la HMI del dispositivo móvil, se debe registrar el puerto en el que se receptara el SMS por la aplicación. En las siguientes líneas de código se puede observar el puerto que se registra para recibir el SMS y el método que se genera cuando recibe el SMS. Ver más en la sección 4.5

```
smsPort="16500";
   String smsConnection = "sms://:" + smsPort;
   try {
      smsconn = (MessageConnection)Connector.open(smsConnection);
      smsconn.setMessageListener(this);
   } catch (IOException ioe) {
      ioe.printStackTrace();
   }
 connections = PushRegistry.listConnections(true);
  if ((connections == null) || (connections.length == 0)) {
      alert1.setString("Procesando pedido...");
      alert1.setImage(image2);
  }
  thread = new Thread(this);
  thread.start();
```

8.5. SISTEMA DE RESPALDO

Para poder almacenar los pedidos en la memoria del dispositivo móvil en necesario crear un registro para poder disponer del espacio para almacenar lo cual es realizado al instalar la aplicación ya que se ejecuta en la función constructora.

```
 for (i=0;i<=39;i++){
            abrir_record_store();
            try{
               reg_pedidos.addRecord(vacio, 0, vacio.length);
            }catch (Exception e) {
               System.out.println("Error de inicializacion de registros: "+e.toString());
            }
            cerrar_record_store();
      }
```

Una vez creado los registros se puede almacenar los pedidos setendo el pedido en una de las posiciones del registro funcionando como pila

```
        abrir_record_store();
         try{
```

```
            reg_pedidos.setRecord(1,dato_pedido,0, dato_pedido.length);
          }catch (Exception e){
            System.out.println("Error al setear registros: "+e.toString());
          }
          cerrar_record_store();
```

Una vez que se a procesado el pedido almacenado es necesario eliminarlo en este caso se realiza sobreponiendo el pedido siguiente en el registro uno y así sucesivamente cada vez que se procese un pedido.

```
      for (rem=1;rem<=38;rem++){
            abrir_record_store();
            try{
              lec_registro1=reg_pedidos.getRecord(rem+1);
            }catch (Exception e){
              System.out.println("Error al leer registros: "+e.toString());
            }
            cerrar_record_store();

            abrir_record_store();
            try{
              reg_pedidos.setRecord(rem, lec_registro1,0, lec_registro1.length);
            }catch(Exception e){
              System.out.println("Error cerrando registros: "+e.toString());
            }
            cerrar_record_store();
          }
              String aux2="*";
          lec_registro1= aux2.getBytes();
          abrir_record_store();
          try{
            reg_pedidos.setRecord(39, lec_registro1,0, lec_registro1.length);
          }catch(Exception e){
            System.out.println("Error cerrando registros: "+e.toString());
          }
          cerrar_record_store();
        }
```

8.6. DIAGRAMA DE FLUJO DE NETBEANS

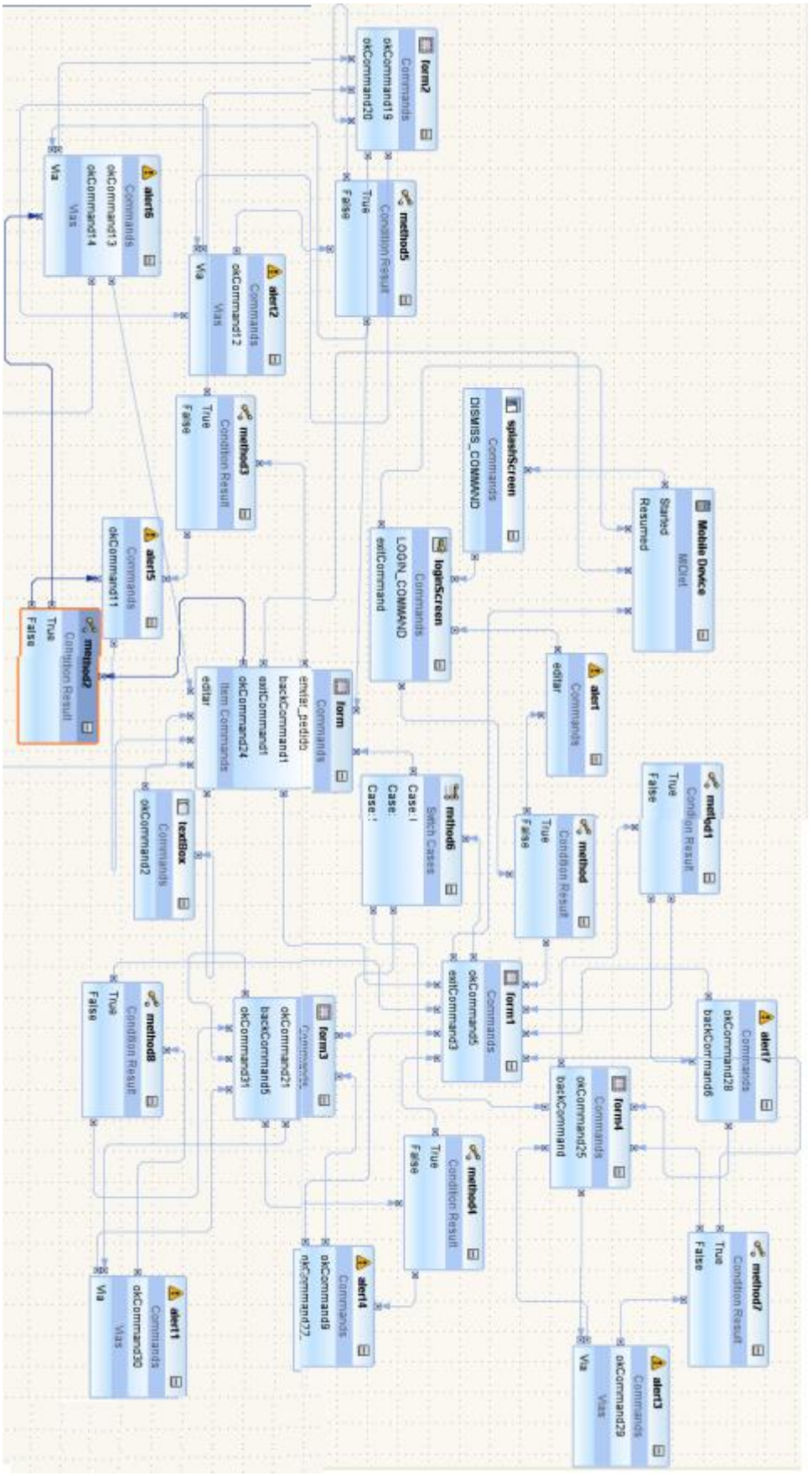


Figura. 8.30. Diagrama de flujo de Netbeans.

CAPÍTULO 9

PRUEBAS Y RESULTADOS

9.1. PRUEBAS REALIZADAS

Con el sistema en funcionamiento se realizaron pruebas para determinar la funcionalidad completa del mismo, es decir se probaron todos los posibles casos de envió y recepción de mensajes SMS desde la HMI del dispositvo móvil, los mismos que se explicaron en la sección 7.6. Los resultados de estas pruebas se muestran más adelante en este capítulo.

Además, se realizaron pruebas de medición de tiempo para determinar cuánto se demorá todo el proceso, desde que se envía el pedido hasta que se recibe la respuesta en el dispositivo móvil, obteniendo un promedio de 28 segundos.

También se realizo pruebas enviando dos mensajes SMS de solicitud de pedido al mismo tiempo, en las que se determino que se procesa el pedido que llega primero al modem de la distribuidora (celular y placa), el pedido que llega después al modem de la distribuidora se descarta.

9.2. PROCESAMIENTO DE PEDIDOS

9.2.1. Pedido Procesado con Éxito

Desde el dispositivo móvil se realiza la solicitud de pedido, como se indico en (sección donde se hace pedido en el dispositivo móvil), se envía el SMS al Modem GSM (celular) de la distribuidora, en la Figura. 9.1 se puede observar la llegada del SMS a la HMI de la PC.

Además, en la Figura. 9.1 se puede observar que el pedido ha sido procesado con éxito, ya que el mensaje de respuesta al dispositivo móvil es: FACOK (Factura procesada con éxito).

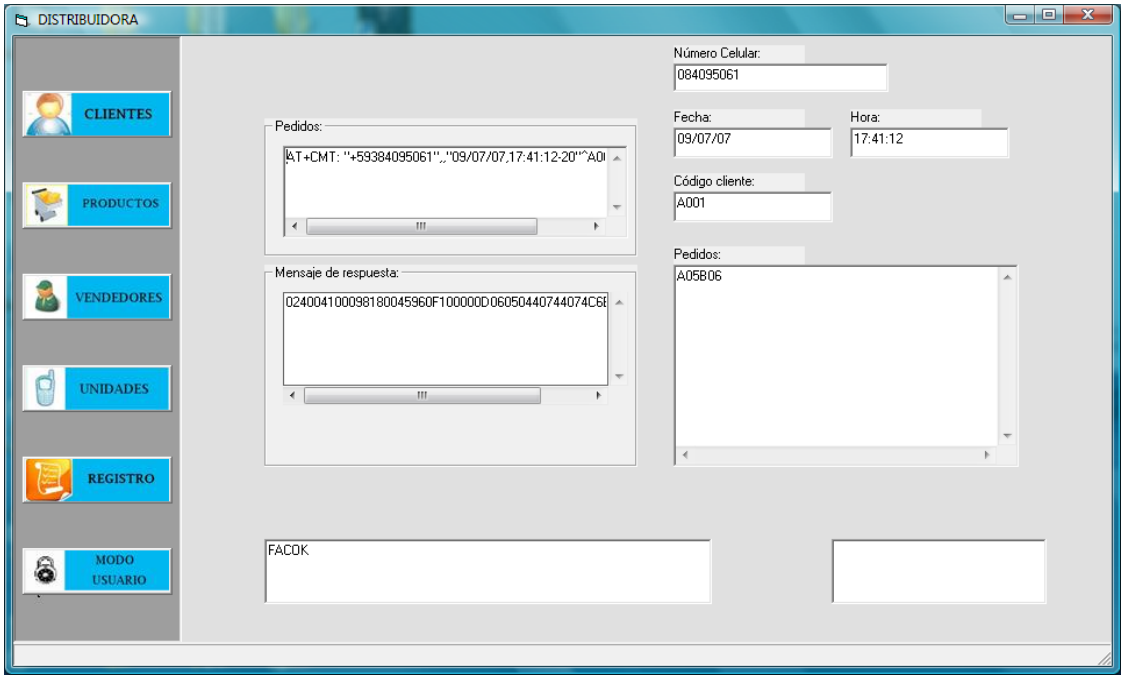


Figura. 9.1. Pedido procesado con éxito.

Una vez que se genera el SMS de respuesta de FACOK, se realiza la impresión de la factura, en la Figura. 9.2 se puede observar el documento que se genera para impresión.

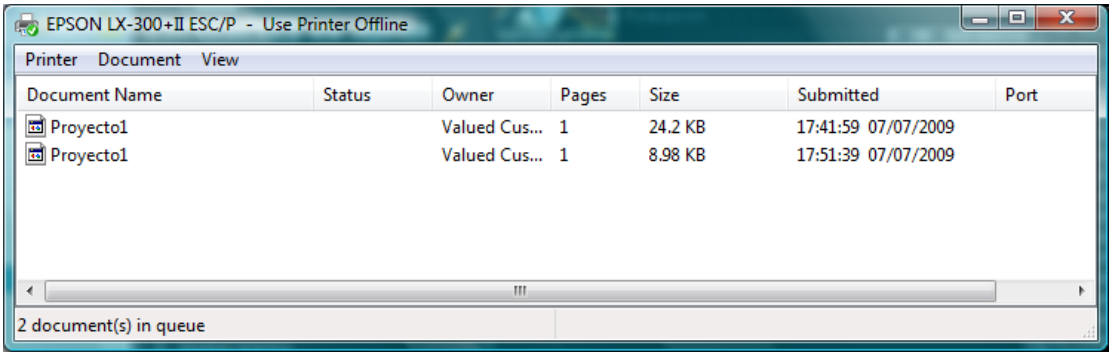


Figura. 9.2. Documentos en bandeja para impresión.

En la Figura. 9.3 se puede observar el SMS que responde la distribuidora en el dispositivo móvil, como resultado de haber procesado el pedido.

Figura. 9.3. SMS de respuesta de pedido procesado con éxito.

9.2.2. Pedido no procesado con éxito

Este SMS de respuesta que se genera cuando el producto que se envía en la solicitud de pedido excede al existe en la tabla de Productos.

En la Figura. 9.4 se puede observar la llegada de SMS en la HMI de la PC y la generación del SMS de respuesta cuando el pedido no ha sido procesado con éxito. El SMS que se genera de respuesta al dispositivo móvil es la información del producto existente en la tabla de Productos.

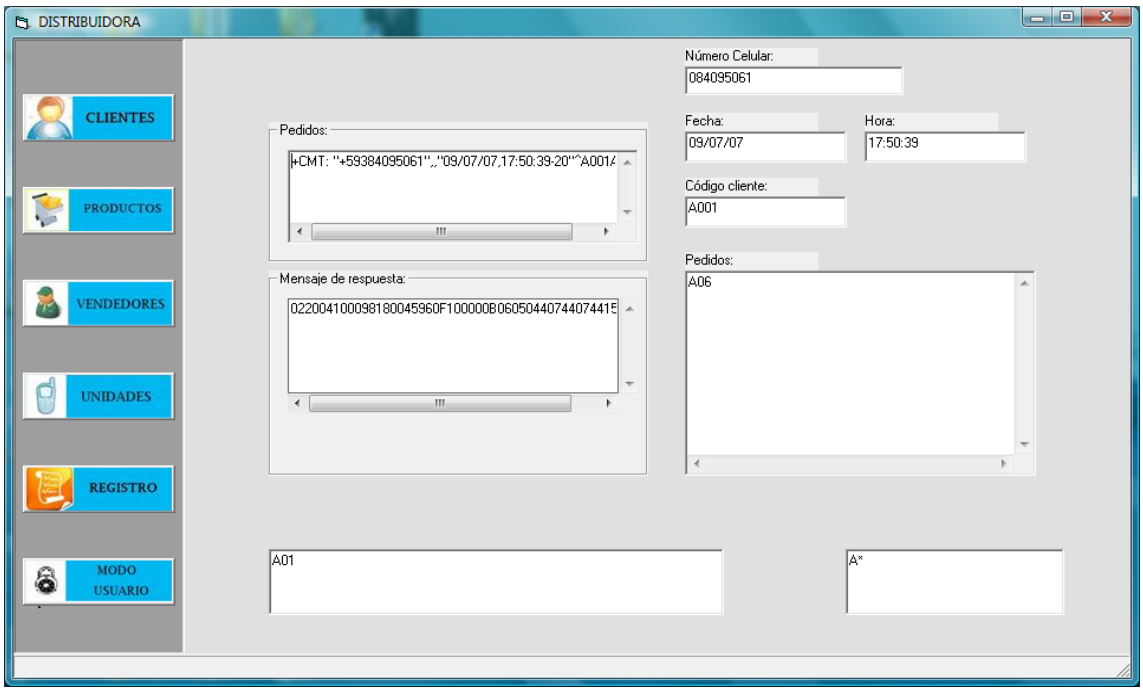


Figura. 9.4. Pedido no presado con éxito.

En la Figura. 9.5 se puede observar la llegada del SMS de respuesta al dispositivo móvil, indicando que producto no satisface con la cantidad solicitada en el pedido.

Figura. 9.5. SMS de respuesta de pedido no procesado con éxito.

9.2.3. Código de cliente no válido

Al enviar un SMS desde un dispositivo móvil con un código de cliente que no existe en la tabla de Clientes, se genera un SMS desde la HMI de la PC indicando que el código que se ha enviado no es válido. Ver más en 7.6 (sección de generación de sms de la HMI de la PC)

En la Figura. 9.6 se puede observar la llegada del SMS y la respuesta generada en la HMI de la PC.

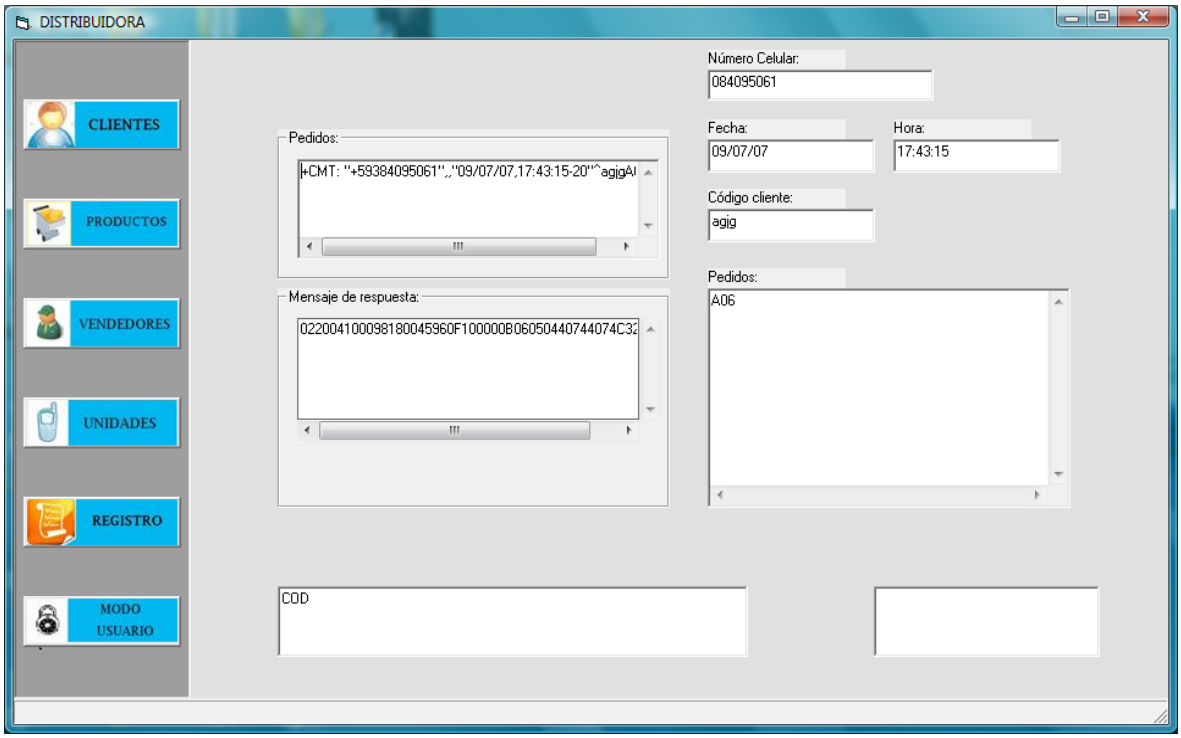


Figura. 9.6. Código de Cliente no válido.

En la se puede observar la llegada del SMS de respuesta al dispositivo móvil cuando la solicitud de pedido enviada por el mismo posee un código de cliente inválido.

Figura. 9.7. SMS de respuesta de Código de Cliente no válido.

9.2.4. Operación de dispositivo no válido

Al enviar un SMS desde un dispositivo móvil no registrado en la tabla de Unidades, se genera un SMS de respuesta indicando que el dispositivo que está solicitando el pedido no puede realizar está operación, como se explicó en 7.6 (sección de sms de respuesta de la HMI de PC).

En la Figura. 9.8 se puede observar la llegada de un SMS desde un dispositivo que no se encuentra registrado en la tabla de unidades y la respuesta que genera este SMS al dispositivo móvil.

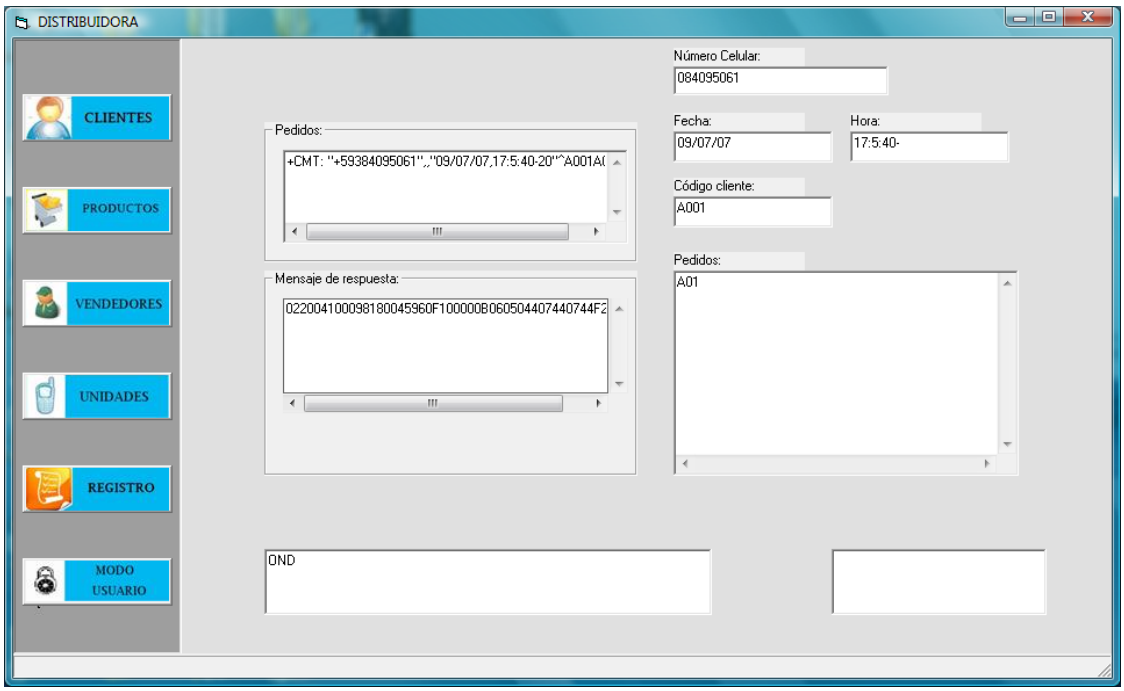


Figura. 9.8. Operación no disponible.

En la Figura. 9.9 se puede observar el SMS de respuesta que se envía desde la HMI de la PC, cuando la solicitud de pedido es proveniente de un dispositivo que no se encuentra registrado en la tabla de Unidades.

Figura. 9.9. SMS de respuesta de Operación no disponible.

9.2.5. Actualizaciones

Al recibir un SMS con una solicitud de actualización de la base de datos del dispositivo móvil, se genera un SMS's de respuesta con la información de la tabla de Productos.

En la Figura. 9.10 se puede observar la llegada de la solicitud de actualización de la tabla de Productos y el SMS de respuesta a esta solicitud.

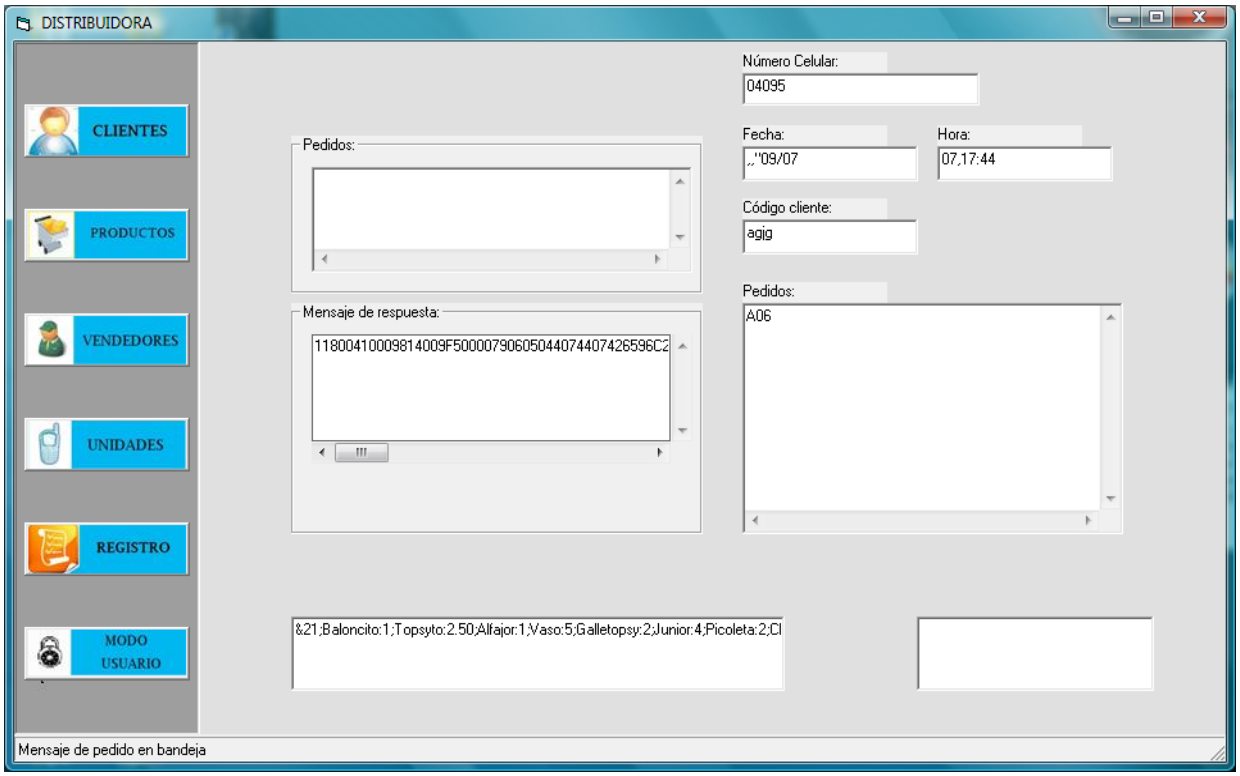


Figura. 9.10. Solicitud de Actualizaciones.

En la Figura. 9.11 se puede observar el SMS de respuesta cuando el dispositivo móvil solicita una actualización de la base de datos.

Figura. 9.11. SMS de respuesta de actualización de Productos.

CAPÍTULO 10

CONCLUSIONES Y RECOMENDACIONES

10.1. CONCLUSIONES

- La transmisión vía GSM es algo muy útil y de rápida implementación que no requiere de grandes inversiones para el usuario debido a que no hace falta comprar equipos muy costosos, con tan solo adquirir el Modem GSM es suficiente para poder transmitir desde cualquier lugar en donde exista cobertura celular.

- El Hardware adicional que se diseñó tiene un amplio rango de usos pues consta de un microcontrolador con varias salidas disponibles que pueden ser configurados para otro tipo de aplicación y con las utilidades del mismo.

- El Software desarrollado es muy flexible y adaptable al usuario de una manera muy sencilla el cual no necesita de supervisión para su funcionamiento.

- El formato de envío de mensajes SMS mediante tramas PDU, permite enviar información de control, adicional al texto del mensaje propiamente dicho, como por ejemplo: direccionar el mensaje hacia un puerto virtual específico del celular de manera que pueda activar una aplicación residente en el teléfono celular del usuario.

- La API WMA (Wireless Messaging API) permite implementar el envío y recepción de mensajes SMS desde y hacia la aplicación J2ME, siendo este medio de comunicación, tan extendido y aceptado por los suscriptores de telefona celular, la base del presente proyecto.

- Es importante considerar que las operaciones de control y localización de este prototipo no se realizan en tiempo real, debido a que dependen de los retardos en la transmisión de los mensajes SMS dentro de la red GSM y la congestión en la misma siendo el proceso en un promedio de 16s.

- Una ventaja de utilizar un lenguaje de alto nivel en la programación del microcontrolador , es que ha sido posible la implementación de dos líneas de comunicación serial por software para el intercambio de información tanto con el Modem GSM ,como con la PC

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

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

10.2. RECOMENDACIONES

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

- Para la implementación de este prototipo se recomienda la utilización como módulo GSM un celular reciclable ya que de esta manera disminuimos el costo del sistema y reutilizamos productos en desuso.

- Se recomienda extender la investigación de esta aplicación con la utilización de otros servicios basados en la red de telefonía celular GSM, como es el servicio MMS(Multimedia Messaging System) o redes para la transmisión de datos como la red GPRS (General Packet Radio Service), con el fin de abrir las puertas al desarrollo de aplicaciones que incluyan otro tipo de funcionalidades.

- Se recomienda continuar con la investigación sobre el desarrollo de aplicaciones en la plataforma J2ME, especialmente aquellas que implementen otros tipos de protocolos de comunicaciones, como por ejemplo bluetooth; así también, protocolos de alto nivel como http (HyperText Transport Protocol).

- Se recomienda usar la aplicación en celulares Motorola, ya que en las pruebas que se realizaron, los resultados fueron los esperados y ninguno de los modelos presentó algún mal funcionamiento.

ANEXO 1

GSM 7-bit Default Alphabet y codificación de datos de 7 bits en octetos

A.1.1 GSM 7-bit Default Alphabet

El siguiente es el alfabeto de 7 bits (7-bit Default Alphabet), especificado en el documento GSM 03.38. En la columna del extremo derecho se muestran los códigos decimales correspondientes ISO-8859-1.

Hex	Dec	Nombre del caracter	Caracter	ISO-8859-1
0x00	0	COMMERCIAL AT	@	64
0x01	1	POUND SIGN	£	163
0x02	2	DOLLAR SIGN	$	36
0x03	3	YEN SIGN		165
0x04	4	LATIN SMALL LETTER E WITH GRAVE	è	232
0x05	5	LATIN SMALL LETTER E WITH ACUTE	é	233
0x06	6	LATIN SMALL LETTER U WITH GRAVE	ù	249
0x07	7	LATIN SMALL LETTER I WITH GRAVE	ì	236
0x08	8	LATIN SMALL LETTER O WITH GRAVE	ò	242
0x09	9	LATIN CAPITAL LETTER C WITH CEDILLA	Ç	199
0x0A	10	LINE FEED		10
0x0B	11	LATIN CAPITAL LETTER O WITH STROKE	Ø	216
0x0C	12	LATIN SMALL LETTER O WITH STROKE	ø	248
0x0D	13	CARRIAGE RETURN		13
0x0E	14	LATIN CAPITAL LETTER A WITH RING ABOVE	Å	197
0x0F	15	LATIN SMALL LETTER A WITH RING ABOVE	å	229
0x10	16	GREEK CAPITAL LETTER DELTA	Δ	
0x11	17	LOW LINE	_	95

Hex	Dec	Nombre del caracter	Caracter	ISO-8859-1
0x12	18	GREEK CAPITAL LETTER PHI	Φ	
0x13	19	GREEK CAPITAL LETTER GAMMA	Γ	
0x14	20	GREEK CAPITAL LETTER LAMBDA	Λ	
0x15	21	GREEK CAPITAL LETTER OMEGA	Ω	
0x16	22	GREEK CAPITAL LETTER PI	Π	
0x17	23	GREEK CAPITAL LETTER PSI	Ψ	
0x18	24	GREEK CAPITAL LETTER SIGMA	Σ	
0x19	25	GREEK CAPITAL LETTER THETA	Θ	
0x1A	26	GREEK CAPITAL LETTER XI	Ξ	
0x1B	27	ESCAPE TO EXTENSION TABLE		
0x1B0A	27 10	FORM FEED		12
0x1B14	27 20	CIRCUMFLEX ACCENT	^	94
0x1B28	27 40	LEFT CURLY BRACKET	{	123
0x1B29	27 41	RIGHT CURLY BRACKET	}	125
0x1B2F	27 47	REVERSE SOLIDUS (BACKSLASH)	\	92
0x1B3C	27 60	LEFT SQUARE BRACKET	[	91
0x1B3D	27 61	TILDE	~	126
0x1B3E	27 62	RIGHT SQUARE BRACKET	]	93
0x1B40	27 64	VERTICAL BAR	|	124
0x1C	28	LATIN CAPITAL LETTER AE	Æ	198
0x1D	29	LATIN SMALL LETTER AE	æ	230
0x1E	30	LATIN SMALL LETTER SHARP S (German)	ß	223
0x1F	31	LATIN CAPITAL LETTER E WITH ACUTE	É	201
0x20	32	SPACE		32
0x21	33	EXCLAMATION MARK	!	33
0x22	34	QUOTATION MARK	”	34
0x23	35	NUMBER SIGN	#	35
0x25	37	PERCENT SIGN	%	37
0x26	38	AMPERSAND	&	38
0x27	39	APOSTROPHE	’	39
0x28	40	LEFT PARENTHESIS	(	40
0x29	41	RIGHT PARENTHESIS	)	41
0x2A	42	ASTERISK	*	42
0x2B	43	PLUS SIGN	+	43

Hex	Dec	Nombre del caracter	Caracter	ISO-8859-1
0x2C	44	COMMA	,	44
0x2D	45	HYPHEN-MINUS	-	45
0x2E	46	FULL STOP	.	46
0x2F	47	SOLIDUS (SLASH)	/	47
0x30	48	DIGIT ZERO	0	48
0x31	49	DIGIT ONE	1	49
0x32	50	DIGIT TWO	2	50
0x33	51	DIGIT THREE	3	51
0x34	52	DIGIT FOUR	4	52
0x35	53	DIGIT FIVE	5	53
0x36	54	DIGIT SIX	6	54
0x37	55	DIGIT SEVEN	7	55
0x38	56	DIGIT EIGHT	8	56
0x39	57	DIGIT NINE	9	57
0x3A	58	COLON	:	58
0x3B	59	SEMICOLON	;	59
0x3C	60	LESS-THAN SIGN	¡	60
0x3D	61	EQUALS SIGN	=	61
0x3E	62	GREATER-THAN SIGN	¿	62
0x3F	63	QUESTION MARK	?	63
0x40	64	INVERTED EXCLAMATION MARK	¡	161
0x41	65	LATIN CAPITAL LETTER A	A	65
0x42	66	LATIN CAPITAL LETTER B	B	66
0x43	67	LATIN CAPITAL LETTER C	C	67
0x44	68	LATIN CAPITAL LETTER D	D	68
0x45	69	LATIN CAPITAL LETTER E	E	69
0x46	70	LATIN CAPITAL LETTER F	F	70
0x47	71	LATIN CAPITAL LETTER G	G	71
0x48	72	LATIN CAPITAL LETTER H	H	72
0x49	73	LATIN CAPITAL LETTER I	I	73
0x4A	74	LATIN CAPITAL LETTER J	J	74
0x4B	75	LATIN CAPITAL LETTER K	K	75
0x4C	76	LATIN CAPITAL LETTER L	L	76
0x4D	77	LATIN CAPITAL LETTER M	M	77

Hex	Dec	Nombre del caracter	Caracter	ISO-8859-1
0x4E	78	LATIN CAPITAL LETTER N	N	78
0x4F	79	LATIN CAPITAL LETTER O	O	79
0x50	80	LATIN CAPITAL LETTER P	P	80
0x51	81	LATIN CAPITAL LETTER Q	Q	81
0x52	82	LATIN CAPITAL LETTER R	R	82
0x53	83	LATIN CAPITAL LETTER S	S	83
0x54	84	LATIN CAPITAL LETTER T	T	84
0x55	85	LATIN CAPITAL LETTER U	U	85
0x56	86	LATIN CAPITAL LETTER V	V	86
0x57	87	LATIN CAPITAL LETTER W	W	87
0x58	88	LATIN CAPITAL LETTER X	X	88
0x59	89	LATIN CAPITAL LETTER Y	Y	89
0x5A	90	LATIN CAPITAL LETTER Z	Z	90
0x5B	91	LATIN CAPITAL LETTER A WITH DIAERESIS	Ä	196
0x5C	92	LATIN CAPITAL LETTER O WITH DIAERESIS	Ö	214
0x5D	93	LATIN CAPITAL LETTER N WITH TILDE	Ñ	209
0x5E	94	LATIN CAPITAL LETTER U WITH DIAERESIS	Ü	220
0x5F	95	SECTION SIGN	§	167
0x60	96	INVERTED QUESTION MARK	¿	191
0x61	97	LATIN SMALL LETTER A	a	97
0x62	98	LATIN SMALL LETTER B	b	98
0x63	99	LATIN SMALL LETTER C	c	99
0x64	100	LATIN SMALL LETTER D	d	100
0x65	101	LATIN SMALL LETTER E	e	101
0x66	102	LATIN SMALL LETTER F	f	102
0x67	103	LATIN SMALL LETTER G	g	103
0x68	104	LATIN SMALL LETTER H	h	104
0x69	105	LATIN SMALL LETTER I	i	105
0x6A	106	LATIN SMALL LETTER J	j	106
0x6B	107	LATIN SMALL LETTER K	k	107
0x6C	108	LATIN SMALL LETTER L	l	108
0x6D	109	LATIN SMALL LETTER M	m	109
0x6E	110	LATIN SMALL LETTER N	n	110
0x6F	111	LATIN SMALL LETTER O	o	111

Hex	Dec	Nombre del caracter	Caracter	ISO-8859-1
0x70	112	LATIN SMALL LETTER P	p	112
0x71	113	LATIN SMALL LETTER Q	q	113
0x72	114	LATIN SMALL LETTER R	r	114
0x73	115	LATIN SMALL LETTER S	s	115
0x74	116	LATIN SMALL LETTER T	t	116
0x75	117	LATIN SMALL LETTER U	u	117
0x76	118	LATIN SMALL LETTER V	v	118
0x77	119	LATIN SMALL LETTER W	w	119
0x78	120	LATIN SMALL LETTER X	x	120
0x79	121	LATIN SMALL LETTER Y	y	121
0x7A	122	LATIN SMALL LETTER Z	z	122
0x7B	123	LATIN SMALL LETTER A WITH DIAERESIS	ä	228
0x7C	124	LATIN SMALL LETTER O WITH DIAERESIS	ö	246
0x7D	125	LATIN SMALL LETTER N WITH TILDE	ñ	241
0x7E	126	LATIN SMALL LETTER U WITH DIAERESIS	ü	252
0x7F	127	LATIN SMALL LETTER A WITH GRAVE	à	224

A.1.2 Codificación de datos de 7 bits en octetos

El mensaje "ESPE-DEEE" consiste de 9 caracteres, llamados septetos cuando cada uno es representado por 7 bits. Estos septetos necesitan ser transformados en octetos para la transferencia del mensaje SMS.

Mensaje	**E**	**S**	**P**	**E**	**-**	**D**	**E**	**E**	**E**
ISO-8859-1	69	83	80	69	45	68	69	69	69
Septetos	1000101	1010011	1010000	1000101	0101101	1000100	1000101	1000101	1000101
	1000101	1010011	1010000	1000101	0101101	1000100	1000101	1000101	1000101

El primer septeto (E) es convertido en octeto añadiendo el bit menos significativo del segundo septeto. Este bit es insertado a la izquierda, lo que produce 11000101 ("C5").

El bit menos significativo del segundo carácter es entonces usado, de tal manera que el segundo carácter (septeto) necesita dos bits (en negrilla) del tercer carácter para transformarse en octeto. Este proceso continua con el resto de caracteres produciendo los siguientes octetos:

Octetos	11000101	00101001	10110100	11011000	00100010	00010110	10001011		01000101
	C5	29	B4	D8	22	16	8B		45

Los 8 octetos procedentes de ”ESPE-DEEE” son C5 29 B4 D8 22 16 8B 45. [5].

ANEXO 2

Hoja de especificaciones del microcontrolador ATmega16.

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 16K Bytes of In-System Self-Programmable Flash
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 512 Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 1K Byte Internal SRAM
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels in TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad MLF
- Operating Voltages
  - 2.7 - 5.5V for ATmega16L
  - 4.5 - 5.5V for ATmega16
- Speed Grades
  - 0 - 8 MHz for ATmega16L
  - 0 - 16 MHz for ATmega16
- Power Consumption @ 1 MHz, 3V, and 25°C for ATmega16L
  - Active: 1.1 mA
  - Idle Mode: 0.35 mA
  - Power-down Mode: < 1 µA

ATMEL®

8-bit AVR® Microcontroller with 16K Bytes In-System Programmable Flash

ATmega16
ATmega16L

Summary

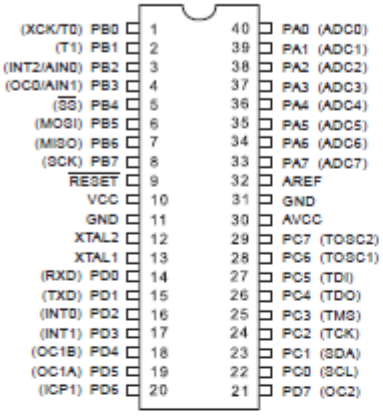
2466HS–AVR–12/03

Note: This is a summary document. A complete document is available on our Web site at www.atmel.com.

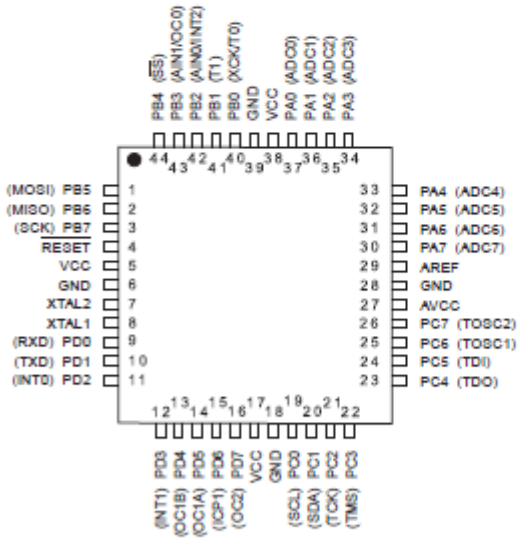
Pin Configurations

Figure 1. Pinouts ATmega16

PDIP



TQFP/MLF



Disclaimer

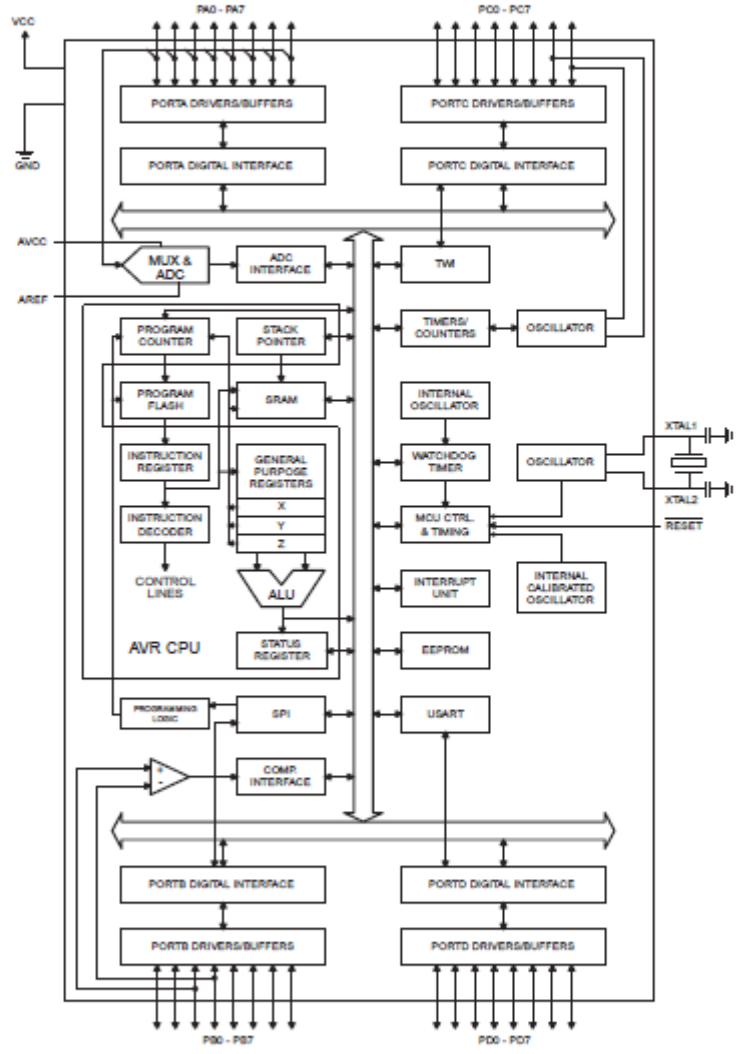
Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

Overview

The ATmega16 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega16 provides the following features: 16K bytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 512 bytes EEPROM, 1K byte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundary-scan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the USART, Two-wire interface, A/D Converter, SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next External Interrupt or Hardware Reset. In Power-save mode, the Asynchronous Timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

The device is manufactured using Atmel's high density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega16 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega16 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

Pin Descriptions

VCC

Digital supply voltage.

GND

Ground.

Port A (PA7..PA0)

Port A serves as the analog inputs to the A/D Converter.

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B (PB7..PB0)	Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port B also serves the functions of various special features of the ATmega16 as listed on page 56.
Port C (PC7..PC0)	Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs. Port C also serves the functions of the JTAG interface and other special features of the ATmega16 as listed on page 59.
Port D (PD7..PD0)	Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port D also serves the functions of various special features of the ATmega16 as listed on page 61.
RESET	Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 36. Shorter pulses are not guaranteed to generate a reset.
XTAL1	Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.
XTAL2	Output from the inverting Oscillator amplifier.
AVCC	AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.
AREF	AREF is the analog reference pin for the A/D Converter.

Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C	7
$3E ($5E)	SPH	–	–	–	–	–	SP10	SP9	SP8	10
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	10
$3C ($5C)	OCR0	Timer/Counter0 Output Compare Register								83
$3B ($5B)	GICR	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	46, 67
$3A ($5A)	GIFR	INTF1	INTF0	INTF2	–	–	–	–	–	68
$39 ($59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	83, 114, 132
$38 ($58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	84, 115, 132
$37 ($57)	SPMCR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	249
$36 ($56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	178
$35 ($55)	MCUCR	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	30, 66
$34 ($54)	MCUCSR	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	39, 67, 229
$33 ($53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	81
$32 ($52)	TCNT0	Timer/Counter0 (8 Bits)								83
$31[1] ($51)[1]	OSCCAL	Oscillator Calibration Register								28
	OCDR	On-Chip Debug Register								225
$30 ($50)	SFIOR	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	55,66,133,199,219
$2F ($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	109
$2E ($4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	112
$2D ($4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte								113
$2C ($4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								113
$2B ($4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte								113
$2A ($4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte								113
$29 ($49)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte								113
$28 ($48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte								113
$27 ($47)	ICR1H	Timer/Counter1 – Input Capture Register High Byte								114
$26 ($46)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte								114
$25 ($45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	127
$24 ($44)	TCNT2	Timer/Counter2 (8 Bits)								129
$23 ($43)	OCR2	Timer/Counter2 Output Compare Register								129
$22 ($42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	130
$21 ($41)	WDTCR	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	41
$20[2] ($40)[2]	UBRRH	URSEL	–	–	–	UBRR[11:8]				165
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	164
$1F ($3F)	EEARH	–	–	–	–	–	–	–	EEAR8	17
$1E ($3E)	EEARL	EEPROM Address Register Low Byte								17
$1D ($3D)	EEDR	EEPROM Data Register								17
$1C ($3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	17
$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	64
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	64
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	64
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	64
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	64
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	64
$15 ($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	65
$14 ($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	65
$13 ($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	65
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	65
$11 ($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	65
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	65
$0F ($2F)	SPDR	SPI Data Register								140
$0E ($2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	140
$0D ($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	138
$0C ($2C)	UDR	USART I/O Data Register								161
$0B ($2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	162
$0A ($2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	163
$09 ($29)	UBRRL	USART Baud Rate Register Low Byte								165
$08 ($28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	200
$07 ($27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	215
$06 ($26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	217
$05 ($25)	ADCH	ADC Data Register High Byte								218
$04 ($24)	ADCL	ADC Data Register Low Byte								218
$03 ($23)	TWDR	Two-wire Serial Interface Data Register								180
$02 ($22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	180

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← $FF − Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← $00 − Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd − 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← $FF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2 / 3
CP	Rd,Rr	Compare	Rd − Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd − Rr − C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd − K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)← Rd(7..4),Rd(7..4)← Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1

BIBLIOGRAFÍA

[1] International Engineering Consortium, Global System for Mobile Communication (GSM),http://www.iec.org/online/tutorials/gsm/topic02.asp. Fecha de consulta:15-11-2008

[2] Tecnología,Red,Datos,GSM,GPRS, http://www.digytech.net/v1/index.php?option=com_content&view=article&id=186&Itemid=297. Fecha de consulta:15-11-2008

[3] Tecnología GSM, http://www.carpross.com.mx/gps/tecnologiagsm.html. Fecha de consulta:16-11-2008

[4] Gómez J.G .,EL SERVICIO SMS: UN ENFOQUE PRACTICO, Escuela Técnica Superior de Ingeniería Informática UAM,2002. Fecha de consulta:20-11-2008

[5] Moya D.& Benítez E., Proyecto de Pregrado ,Prototipo de una tarjeta para el control y localización vehicular utilizando mensajes,2008. Fecha de consulta:05-03-2009

[6] Zapata S.& Vallejo A., Proyecto de Pregrado,Diseño e implementación de un sistema de vigilancia remota para una residencia utilizando plataformas GPRS e Internet,2007. Fecha de consulta: 05-02-2009

[7] SERGIO GÁLVEZ ROJAS LUCAS ORTEGA DÍAZ, JAVA A TOPE: J2ME (JAVA 2MICRO EDITION). EDICIÓN ELECTRÓNICA,2003. Fecha de consulta:15-12-2008

[8] Sergio Gálvez, Lucas Ortega, J2ME Java 2 Micro Edition, www.lcc.uma.es/galvez/J2ME.html, 2003, Fecha de consulta: 02-04-2009

[9] J2ME, http://pegasus.javeriana.edu.co/~mad/J2ME.pdf Fecha de consulta:07-01-2009

[10] VisualMobileDesignerPalatteReference, netbeans.org, http://wiki.netbeans.org/VisualMobileDesignerPalatteReference Fecha de consulta:10-01-2009

[11] ZENON CUCHO M., Microcontroladores, Pontificia Universidad Catolica Del Peru, http://www.lulu.com/items/volume_38/588000/588200/1/print/SESION_1_ATMEGA8.pdf 2007. Fecha de consulta:30-01-2009

[12] Visual Basic, http://msdn.microsoft.com Fecha de consulta:20-01-2009

[13] Control Frame de Visual Basic, http://www.recursosvisualbasic.com.ar/htm/tutoriales/control_frame.htm Fecha de consulta:20-01-2009

[14] Manuales, artículos, guías y tutoriales de Visual Basic, http://www.recursosvisualbasic.com.ar/htm/menu-principal/manuales-articulos-tutorial.htm#1. Fecha de consulta:06-04-2009

[15] Que son las bases de datos , http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/. Fecha de consulta:15-05-2009

[16] Tipos de impresoras, http://www.informatica-hoy.com.ar/hardware-perifericos/Tipos-de-impresoras.php. Fecha de consulta:20-05-2009

FECHA DE ENTREGA:_____

_____ _____

Sr. Cristian Karolys Sr. Diego Niama

AUTORES

_____

Ing. Víctor Proaño

DIRECTOR DE CARRERA DE INGENIERIA

EN ELECTRONICA Y AUTOMATIZACIÓN Y CONTROL

_____

Ing. Gonzalo Olmedo

DIRECTOR DE CARRERA DE INGENIERIA

EN ELECTRONICA Y TELECOMUNICACIONES