ESCUELA POLITÉCNICA DEL EJÉRCITO

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO PARA REGISTRAR LA CANTIDAD DE LECHE RECOLECTADA EN UN ORDEÑO MECÁNICO A TRAVÉS DE SENSORES Y DE LA UTILIZACIÓN DE MENSAJES SMS.

CARLOS IVÁN MERCHÁN LASCANO

SANGOLQUÍ – ECUADOR
2010

CERTIFICACIÓN

Certificación por parte del Director y Codirector de la elaboración del proyecto bajo su dirección, y pie de firmas:

_____	_____
Ing. Gonzalo Olmedo	Ing. Julio Larco
DIRECTOR	CODIRECTOR

RESUMEN

En el presente proyecto se diseño e implementó un sistema de monitoreo de cantidad de leche recolectada en un ordeño mecánico, a través de sensores y mensajes SMS, con lo que se pretende dar un mayor control de la cantidad exacta de leche que se está produciendo en las haciendas ganaderas. El sistema de monitoreo consta de dos etapas:

Etapa de Transmisión: Consiste en sensar la cantidad de leche que extrae el sistema de ordeño mecánico a través de sensores; los valores obtenidos de los sensores son procesados por un sistema de control (microcontrolador), el cuál mediante un circuito integrado reloj-calendario (DS1307) obtiene los valores de los sensores en dos horas específicas del día, posteriormente, estos valores son trasmitidos mediante un modem GSM en forma de SMS a un centro de gestión (Sistema Receptor).

Etapa de Recepción: Consiste en una aplicación que recibe los mensajes SMS a través del Modem, este SMS contiene los datos de la recolección de la leche del Sistema Transmisor, que son decodificados por la aplicación. Para la decodificación del mensaje se utilizó el software SharpDevelop. Una vez decodificado el mensaje el Software SharpDevelop almacena los datos en una Base de Datos, la cual fue previamente creada mediante el Software Microsoft SQL Server 2000. Los datos almacenados pueden ser consultados y presentados en un reporte, mediante los Software Crystal Report XI y Visual .Net 2005.

DEDICATORIA

Dedico este proyecto a mis padres, Carlos y Mariana, por su apoyo incondicional, por su comprensión y guía en todo mi crecimiento personal y profesional.

A mi hermana Eliana, por su constante apoyo en todo momento en mi vida.

A mi novia Lizeth, por estar siempre a mi lado dándome fuerzas para ser cada día mejor.

A mis tíos, Margot y Antonio, por ser mis segundos padres y brindarme siempre su apoyo, cariño y enseñanzas.

A toda mi familia, que con su amor, oración y enseñanzas, forman parte de este proyecto.

AGRADECIMIENTO

A DIOS, por darme la vida y permitirme culminar la etapa Universitaria.

A mis padres y hermana, por su apoyo incondicional durante todo momento en mi vida académica.

A mi novia, por apoyarme en todo momento durante el desarrollo de este proyecto.

A los Ingenieros, Gonzalo Olmedo y Julio Larco, por su guia y colaboración en el desarrollo de este proyecto.

PROLOGO

Mediante el diseño e implementación del sistema de monitoreo de la cantidad de leche en un ordeño mecánico, a través de mensajes SMS se pretende dar un mayor control de la cantidad exacta de leche que se está produciendo en las haciendas ganaderas y de esta forma evitar robos que perjudican a sus dueños.

El presente Proyecto consta del siguiente contenido:

El Primer Capítulo “TECNOLOGÍA GSM, ENFOCADA A LA TRANSMISIÓN DE MENSAJES (SMS)”, describe las principales características y arquitectura de la Tecnología GSM. También describe características, arquitectura, tipos, modelos de capas, beneficios, y aplicaciones del servicio de SMS.

El Segundo Capítulo “ANÁLISIS DEL SISTEMA DE ORDEÑO MECÁNICO”, describe tipos, elementos, frecuencias, tipos de sala, ventajas y desventajas del ordeño mecánico, para el análisis del diseño e instalación del Sistema de Monitoreo.

El Tercer Capítulo “DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE RECOLECCIÓN Y ENVÍO DE LA CANTIDAD DE LECHE (SISTEMA TRANSMISOR)”, describe las etapas que componen el Sistema de Transmisión, el funcionamiento a través de Diagramas de Flujo de cada etapa y la codificación de cada etapa. También describe el diagrama circuital, el circuito impreso, las herramientas (Software y Hardware) y los costos para el diseño del Sistema Transmisor.

El Cuarto Capítulo “DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE RECEPCIÓN DE LA CANTIDAD DE LECHE (SISTEMA RECEPTOR)”, describe las etapas que componen el Sistema de Recepción, el funcionamiento a través de Diagramas de Flujo de cada etapa y la codificación de cada etapa. También describe las herramientas (Software) y los costos para el diseño del Sistema Receptor.

El Quinto Capítulo describe las Conclusiones y Recomendaciones, obtenidas en la realización del Proyecto.

Los Anexos describen los Diagramas de Flujo y Codificación de los Sistemas Transmisor y Receptor, también describen las hojas técnicas de los elementos electrónicos utilizados para la realización del proyecto.

INDICE DE CONTENIDO

INDICE DE TABLAS

INDICE DE FIGURAS

GLOSARIO

TERMINO	SIGNIFICADO
DCS1800	Digital Cellular System 1800
GSM	Global System for Mobile Communications
SIM	Subscritor Identity Module
PIN	Personal Identification Number
BSS	Base Station Subsystem
BTS	Base Transceiver Station
BSC	Base Station Controller
NSS	Subsistema de conmutación y de red
MSC	Mobile Services Switching Center
GMSC	Gateway Mobile Services Switching Center
HLR	Home Location Register
VLR	Visitor Location Register
AuC	Authentication Center
EIR	Equipment Identy Register
GIWU	GSM Interworking Unit
OSS	Operation and Maintenance Subsystem
SMS	Short Message Service
SME	Short Messaging Entity
MSC	Mobile Switching Center
SMSC	Short Message Service Center
SMS-GMSC	SMS Gateway Mobile Switching Center

TERMINO	SIGNIFICADO
SMS-IWMSC	SMS Inter-working Gateway Mobile Switching Center
SM-AL	Short Message Application Layer
SM-TL	Short Message Transfer Layer
SM-RL	Short Message Relay Layer
SM-LL	Short Message Link Layer
NVRAM	Memoria RAM no volátil
SCL	System Clock
SDA	System Data
RS232	Norma para Comunicación Serial
TTL	Tecnología de construcción de circuitos electrónicos digitales
AT	Comandos para interfaz de comunicación
RTC	Real Time Clock
ATE	Comando AT para abiltar eco del modem
AT+IPR	Comando AT para establecer velocidad de transmisión del modem
AT+CMGF	Comando AT para establecer el modo de entrada de SMS del Modem
AT+CNMI	Comando AT para configurar el formato de SMS del Modem
AT+CSQ	Comando AT para indicar Señal del Modem
AT&W	Comando AT para cguardar cofiguración actual del Modem
AT+CMGS	Comando AT para enviar SMS
SQL	Structured Query Language
ODBC	Open DataBase Connectivity

CAPÍTULO 1

1. TECNOLOGÍA GSM, ENFOCADA A LA TRANSMISIÓN DE MENSAJES (SMS)

1.1. Introducción a la Tecnología GSM

Las telecomunicaciones en Europa siempre habían estado regidas por la estandarización. El CEPT (*Conférence Européene des Postes et Télécommunications*) es una organización para la estandarización presente en más de 20 países europeos. Todos estos factores, llevaron a la creación en 1982 de un nuevo cuerpo de estandarización dentro del CEPT, cuya tarea era especificar un único sistema de radiocomunicaciones para Europa a 900 MHz. El *Groupe Special Mobile* (GSM) tuvo su primer encuentro en Diciembre de 1982 en Estocolmo, bajo la presidencia de Thomas Haug de la administración Sueca. Treinta y una personas de once países estuvieron presentes en este primer encuentro. En 1990, por requerimiento del Reino Unido, se añadió al grupo de estandarización la especificación de una versión de GSM a la banda de frecuencia de 1800 ± 75 MHz. Esta variante se la llamó DCS1800 (*Digital Cellular System 1800*).

El significado actual de las siglas GSM se ha cambiado y en la actualidad significa Global System for Mobile Communications.

Los servicios de datos presentan una tasa de hasta 14,4 kbps. Posee servicios adicionales como FAX, servicio de mensajería corta, desvío de llamadas, prohibición de llamadas, llamada en espera, identificadores de número, informe de costos, *roaming*, entre otros.

1.2. Arquitectura del sistema GSM

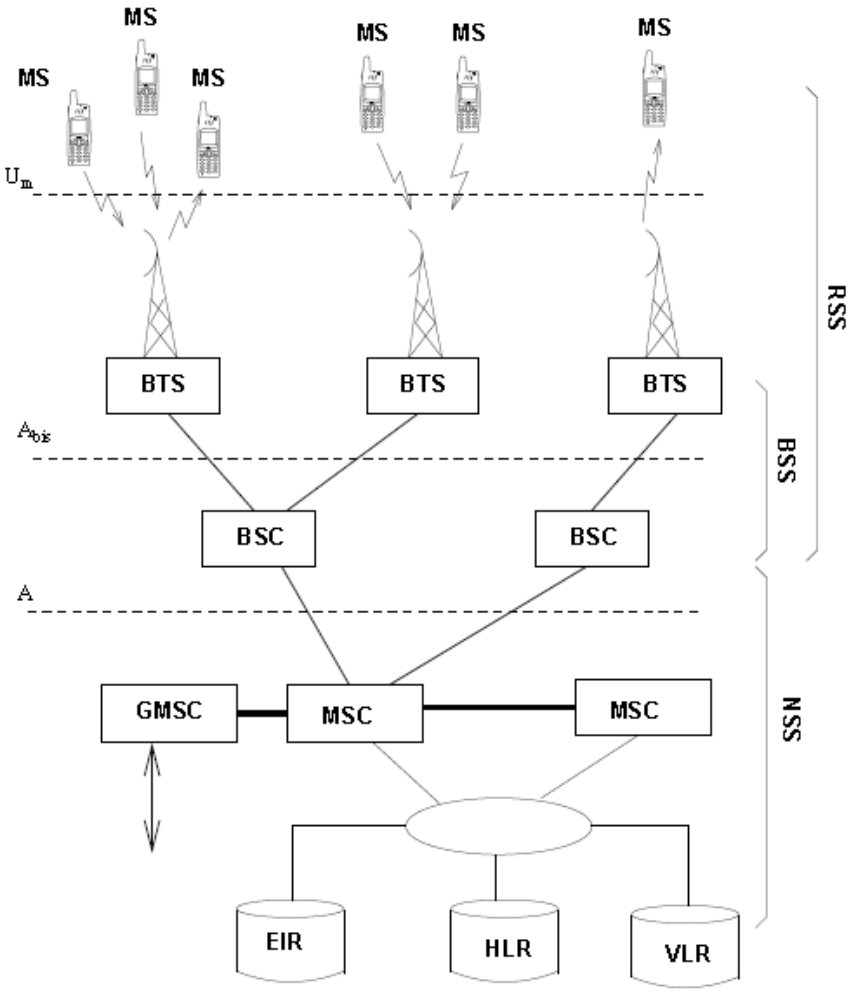


Figura 1.1 Arquitectura del sistema GSM[1]

En la Figura 1.1 se observa la arquitectura del sistema GSM, el cual se compone de cuatro bloques o subsistemas que engloban el conjunto de elementos de la jerarquía del sistema. Cada uno de estos subsistemas desempeña funciones específicas para, en su conjunto, ofrecer el servicio de telefonía móvil al usuario final. Los cuatro subsistemas son:

[1] U_m: Interfaz entre MS y BTS
A_{bis}: Interfaz entre BSC y BTS
A: Interfaz entre BSC y MSC

- La estación móvil (MS)
- El subsistema de estación base (BSS)
- El subsistema de conmutación y de red (NSS)
- El subsistema de operación y mantenimiento (OSS)

El BSS provee y maneja la transmisión entre los MS y el NSS. El NSS tiene la responsabilidad de manejar las comunicaciones y conectar al MS hacia las diferentes redes o con otro MS. El OSS provee los servicios de control y de manejo del sistema GSM. La interacción entre los subsistemas puede ser agrupada en dos partes principales:

- Operacional: Las redes externas entre el NSS, el BSS y el MS.
- Control: OSS de y hacia el servicio proveedor.

1.2.1. La estación móvil (MS)

Está formada por la estación móvil y el SIM[2]. El SIM (Ver Figura 1.2) es una pequeña tarjeta inteligente que sirve para identificar las características del Terminal. Esta tarjeta se encuentra interiormente en el móvil y permite al usuario acceder a todos los servicios que haya disponibles por su operador, sin la tarjeta SIM el operador no sirve para nada porque no se puede hacer uso de la red. El SIM está protegido por un número de cuatro dígitos que recibe el nombre de PIN[3]. La mayor ventaja de las tarjetas SIM es que proporcionan movilidad al usuario ya que puede cambiar de terminal y llevarse consigo el SIM. Una vez que se introduce el PIN en el terminal, el terminal empieza a buscar redes GSM que estén disponibles y va a tratar de validarse en ellas, una vez que la red (generalmente la que se tiene contratada) ha validado el terminal y el teléfono queda registrado en la célula que lo ha validado.

[2] Subscritor Identity Module
[3] Personal Identification Number

Figura 1.2 Tarjeta SIM

1.2.2. El subsistema de estación base (BSS)

El BSS está en contacto directo con las estaciones móviles a través de la interfaz aérea. Por lo tanto, incluye las máquinas encargadas de la transmisión y recepción de radio, y de su gestión. Además, el BSS está en contacto con los conmutadores del NSS. La misión fundamental del BSS es conectar la estación móvil y el NSS. El BSS tiene que ser controlado y, por lo tanto, debe estar en contacto con el OSS.

Consta de dos elementos:

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

La BTS provee la comunicación entre la estación móvil y la red mediante la interfaz aire. Sincroniza la operación y mantenimiento. Se encuentra conectado al BSC. Contiene el equipo para la transmisión y recepción de señales de radio, antenas y equipos de comunicación con la BSC. Sus principales funciones son:

- Codificación/decodificación de los canales.
- Diversidad en recepción.
- Búsqueda de las estaciones móviles.
- Recepción de las peticiones de canal desde las estaciones móviles.

- **BSC (Base Station Controller)**

La BSC es la entidad que conecta la estación base y el centro de conmutación móvil (MSC). Constituye el primer punto de concentración de tráfico hacia la red.

Se encarga de controlar y gestionar varias BTS's en función de su capacidad. Gestiona *handover*[4] entre BTS's. Puede encontrarse junto a una BTS, junto a una MSC o sola. Tiene como principales funciones las de control en el subsistema de estación base:

- Gestión de canales de radio.
- Supervisión de la estación base.
- Traspaso entre canales de la BSC.
- Localización de las estaciones móviles.
- Adaptador de velocidad.
- Gestión de las transmisiones hacia la estación base.
- Corrección de errores.
- Ejecuta algoritmos de control de potencia y cifrado.

1.2.3. El subsistema de conmutación y de red (NSS)

Se encarga de administrar las comunicaciones que se realizan entre los diferentes usuarios de la red. Para poder hacer este trabajo la NSS se divide en 7 partes:

[4] Procesos de transferencia

- **MSC (*Mobile Services Switching Center*)**

Forma la parte central de la NSS y se encarga de las funciones de conmutación dentro de la red así como de proporcionar la conexión entre redes.

- **GMSC (*Gateway Mobile Services Switching Center*)**

Sirve de mediador entre las redes de telefonía fija y la red GSM.

- **HLR (*Home Location Register*)**

Es la base de datos que contiene información sobre los usuarios conectados a una determinada MSC.

- **VLR (*Visitor Location Register*)**

Contiene toda la información sobre un usuario necesaria para que dicho usuario pueda acceder a los servicios de red.

- **AuC (*Authentication Center*)**

Proporciona los parámetros necesarios para la autenticación de usuarios.

- **EIR (*Equipment Identy Register*)**

Contiene una base de datos con todos los terminales que son válidos para ser utilizados en la red.

- **GIWU (*GSM Interworking Unit*)**

Sirve como interfaz de comunicación entre diferentes redes para comunicación de datos.

1.2.4. OSS (*Operation and Maintenance Subsystem*)

La operación de la red es controlada por el subsistema de Operación y mantenimiento. Las funciones de control son monitoreadas desde el OMC *(Operation and Maintenance Center)*. La OMC tiene acceso tanto a la GMSC como a la BSC.

Entre las principales funciones se puede nombrar: operación comercial y de administración, manejo de la seguridad, configuración de la red, y tareas de mantenimiento.

1.3. Servicios de mensajes cortos (SMS)

1.3.1. Introducción

SMS apareció en escena en 1991 en Europa, donde la tecnología inalámbrica digital dio raíces. El Standard Europeo para la tecnología inalámbrica digital, es ahora conocido globalmente como el Standard para móviles (GSM), que incluye el servicio de mensajería corta desde el principio.

En Norte América, SMS estuvo disponible en las redes inalámbricas digitales construidas por los primeros pioneros tales como *BellSouth Mobility* y *Nextel*. En 1998, con el desarrollo de las redes basadas en GSM como el servicio de comunicación personal (PCS), código de acceso por división múltiple (CDMA), y acceso por división de tiempo (TDMA), estos métodos ayudaron a la completa implementación del SMS.

1.3.2. Definición

Servicio de Mensajes Cortos (SMS) es un servicio inalámbrico aceptado globalmente, este permite la transmisión de mensajes alfanuméricos entre clientes de teléfonos móviles y sistemas externos tales como correo electrónico, paging[5] y sistemas de mensajes de voz.

Cada mensaje puede tener hasta 160 caracteres cuando se usa el alfabeto latino y 70 caracteres si se usa otro alfabeto como el árabe o el chino.

1.3.3. Arquitectura

En la Figura 1.3, se puede observar la arquitectura de SMS:

Figura 1.3 Arquitectura SMS

- **SME (*Short Messaging Entity*)**

Entidad que puede ser un teléfono móvil, un computador o cualquier otro dispositivo capaz de enviar y recibir mensajes cortos en formato SMS.

- **MSC (*Mobile Switching Center*)**

Realiza funciones de conmutación. Recibe y transfiere los SMS, además puede suministrar información sobre errores en la transferencia de los mensajes de texto.

[5] Servicio de búsqueda de personas

- **SMSC (*Short Message Service Center*)**

Es un centro de almacenamiento y retransmisión, responsable de garantizar la entrega de los mensajes de texto a través de la red. Puede ser parte integrante del MSC o una entidad de red independiente.

Almacena los mensajes hasta que el destino se encuentre disponible y luego los retransmite. Los SMS no pueden enviarse directamente entre SME's sin pasar por este centro. Además el SMSC intercambia con la red mensajes de confirmación de recepción/ envío de los mensajes cortos.

- **SMS-GMSC(*SMS Gateway Mobile Switching Center*)**

Es un MSC capaz de recibir SMS desde un SMSC. El SMS-GMSC interroga al HLR sobre la información de encaminamiento, localiza la MSC actual del receptor y le entrega el SMS para ser enviado al SME destino.

- **SMS-IWMSC(*SMS inter-working Gateway Mobile Switching Center*)**

Es un MSC capaz de recibir un mensaje corto de la red móvil y enviarlo hacia el SMSC apropiado. El SMS-GMSC y SMS-IWMSC están normalmente integrados en el SMSC.

1.3.4. Modelo de Capas

El *stack*[6] de protocolos SMS está compuesto de cuatro capas: Aplicación, Transferencia, Retransmisión y Enlace.

[6] Estructura de datos en la que el modo de acceso a sus elementos es de tipo LIFO

- **Capa de Aplicación**

Consiste en las aplicaciones (editor de mensajes, juegos, etc.) implementadas en el SME para enviar, recibir e interpretar el contenido de los mensajes. Esta capa es también conocida como SM-AL (*Short Message Application Layer*).

- **Capa de Transferencia**

En esta capa el mensaje es considerado como una secuencia de octetos que contiene información como la longitud del mensaje, creador y destinatario del mensaje, fecha de recepción, etc. Esta capa es también conocida como SM-TL (*Short Message Transfer Layer*).

- **Capa de Retransmisión**

Permite el transporte de mensajes entre varios elementos de red. Un elemento de red puede almacenar temporalmente un mensaje si el siguiente elemento en la cadena no está disponible para recibir el mismo. Esta capa es también conocida como SM-RL (*Short Message Relay Layer*).

- **Capa de Enlace**

Realiza la transmisión del flujo de bits del mensaje a través del medio físico, entre las entidades de la red SMS. Esta capa es también conocida como SM-LL (*Short Message Link Layer*).

1.3.5. Elementos del SMS

El SMS para la recepción y transferencia de mensajes comprende siete elementos que son:

- ***Validity Period***

Período durante el cual puede estar almacenado un SMS en el SMSC mientras no pueda ser entregado a su destino, si se supera este tiempo el mensaje es eliminado.

- ***Service Centre Time Stamp***

Elemento que informa el tiempo al que el SMSC recibió el SMS para ser entregado al SME.

- ***Protocol Identifier***

Este elemento indica la forma en la que la aplicación receptora maneja los mensajes entrantes.

- ***More Messages to Send***

Elemento que le permite al SMSC informarle al SME que más mensajes están esperando para ser entregados. Utiliza un parámetro booleano para indicar si hay más mensajes para enviar.

- ***Priority***

Elemento provisto por el SMSC o SME que indica la importancia relativa de un mensaje.

- ***Messages Waiting***

Cuando un mensaje no puede ser entregado porque el SME no está disponible, permite indicar al HLR que notifique al SMSC cuando ya esté accesible el SME para realizar la entrega del mensaje.

- ***Alert SMSC***

Permite avisar al SMSC de que un SME al que se le había intentado entregar un mensaje sin éxito ya está disponible.

1.3.6. Características

Las características generales de SMS son:

- **Concatenación**

Se pueden concatenar algunos SMS estándar para formar un mensaje largo. Se pueden concatenar hasta 255 mensajes. Cuando esta característica es usada se debe incluir información adicional para que la aplicación puede reensamblar correctamente los mensajes cortos concatenados.

Existe una versión mejorada que también permite concatenar hasta 255 mensajes pero utiliza un campo de referencia de 16 bits en vez de 8 bits que utiliza la versión normal. El campo de referencia de 16 bits reduce la probabilidad de errores en el proceso de concatenación.

- **Compresión**

Permite comprimir los datos de usuario del mensaje. Esta característica es opcional, y se basa en un algoritmo donde la longitud de la secuencia de salida es inversamente proporcional a la frecuencia con que el caracter ocurre en la secuencia de entrada.

- **Mensajería binaria**

El SMS puede ser configurado en modo caracter o binario. El modo binario permite mejorar la eficiencia de los datos transmitidos.

- **Facturación**

Cada mensaje tiene una referencia de facturación asociada, ésta le dice al sistema de facturación la tarifa que se le debe cargar al mensaje.

1.3.7. Tipos de SMS

Los SMS pueden clasificarse según el número de destinatarios en:

- **Mensajes punto a punto**

En este tipo de mensajes el destinatario es único y se pueden clasificar según la dirección de envío en: *Mobile Originated y Mobile Terminated.*

- ***Mobile Originated*****:** El mensaje puede ser enviado a un número corto, que previamente ha sido contratado a las operadoras móviles por parte de las empresas que prestan servicios utilizando SMS. Este tipo de mensajes son los que se emplean para participación en concursos, votaciones, petición de alertas o de recepción de información en el móvil.

- ***Mobile Terminated*****:** El mensaje es enviado desde la fuente hasta el terminal móvil, la fuente puede ser otro usuario móvil o una aplicación. Una vez que el mensaje llega al terminal móvil un reporte confirma a la fuente que la entrega fue completada.

- **Mensajes punto multipunto**

En este tipo, el mensaje es enviado a un conjunto de usuarios. A este tipo corresponde *Cell Broadcast*[7]*.* El destino del mensaje está descrito en términos de identificadores de celda utilizados por la BSC para enrutar el contenido del mensaje a los usuarios de la BTS.

1.3.8. Beneficios de SMS

Los beneficios del servicio SMS para el proveedor son los siguientes:

- El aumento de llamadas gracias a las capacidades de notificación del SMS en las redes inalámbricas.

- Una alternativa al servicio de búsqueda de personas alfanumérico “Paging”.

- Activa el acceso inalámbrico a datos para usuarios de empresas.

- Provisiones de servicios con valor agregado como el e-mail, buzón de voz, la integración de fax, etc.

- Proporciona una herramienta administrativa para servicios como avisos de precios, descargas en forma inalámbrica.

Los beneficios del SMS a los clientes se centran en la conveniencia, flexibilidad y la integración de servicios de mensajes y acceso a datos.

[7] Permite el envío simultáneo de mensajes de hasta 93 bytes a múltiples usuarios en un área geográfica específica.

Desde esta perspectiva, el beneficio es ser capaz de usar un equipo móvil como una extensión del computador.

1.3.9. Aplicaciones

Las principales aplicaciones basadas en SMS son:

- **Mensajes de persona a persona**

Los usuarios de telefonía móvil utilizan comúnmente el servicio de mensajería corto para comunicarse con otro usuario móvil de su misma operadora e incluso de una operadora diferente.

- **Alertas de E-mail**

Los SMS permiten notificar al usuario que tiene un nuevo e-mail. Este mensaje usualmente contiene la dirección de quien envía, el título y unas pocas palabras del inicio de E-mail.

- **Servicios de notificación**

Permite el envío de mensajes a ciertos usuarios que constan en una base de datos específica tales como: clientes de compañías de televisión, clubs deportivos, supermercados y otros minoristas, aerolíneas y bancos. Estos mensajes pueden ser publicitarios o de notificación entre otros.

- **Servicios de información**

Permite enviar al terminal móvil mensajes con pequeños contenidos de información periódica, de un amplio rango como reporte del clima, reportes financieros, información deportiva.

- **Servicios de localización**

Aplicado a la localización de vehículos, integra GPS[8]. Los datos de longitud y latitud son transferidos a un terminal móvil. El terminal por medio de un SMS envía estos datos a un servidor donde se procesan para indicar la localización actual del vehículo en un mapa geográfico.

- **Supervisión Remota**

El servicio de mensajería corta puede usarse para gestionar máquinas en ambientes de supervisión remota. Esta aplicación proporciona valiosa información sobre el estado o el suceso de algún evento ocurrido sobre la máquina, que el usuario precisa saber.

- **Comercio electrónico**

Se pueden llevar a cabo transacciones financieras a través del terminal móvil, para la cual será necesario tener convenios con algunas instituciones bancarias.

[8] Global Position System

CAPÍTULO 2

2. ANÁLISIS DEL SISTEMA DE ORDEÑO MECÁNICO

2.1. Introducción

El sistema de ordeño mecánico consiste en la extracción de leche sin dañar al pezón y al tejido mamario, que se realiza mediante el empleo de elementos mecánicos que generan de manera discontinua y cíclica vacío a nivel del pezón, extrayendo la leche y conduciéndola a un recipiente, simulando la acción del becerro mediante la aplicación de vacío cuando este es alimentado con biberones y mamila de hule.

El elemento que se pone en contacto con el pezón de la vaca se llama pezonera, el cual simula la acción de succión del becerro. La pezonera está incluida en un casco metálico o acrílico a la cual está ajustada, llamado copa (concha). La pezonera se abre y se cierra a consecuencia de la acción de un pulsador, el cual provoca de forma intermitente, vacío parcial y presión atmosférica al espacio entre la pezonera y la copa.

Cuando el pulsador abre el espacio entre la copa y la pezonera al vacío, se igualan las presiones que existen entre el interior y el exterior de la pezonera, tomando la posición de apertura normal; en este periodo fluye la leche del pezón al interior de la pezonera (Ver Figura 2.2). Cuando el aire se introduce entre el casco y la pezonera, la presión fuera de la pezonera aumenta causando la contracción de esta; durante este periodo se proporciona un masaje al pezón (Ver Figura 2.1). Una pulsación comprende la apertura y la contracción de la pezonera.

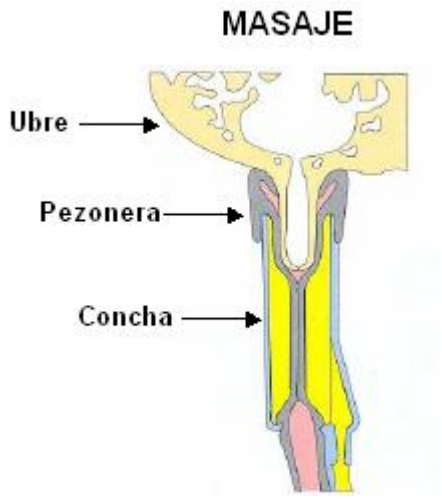


Figura 2.1 Proceso de Masaje

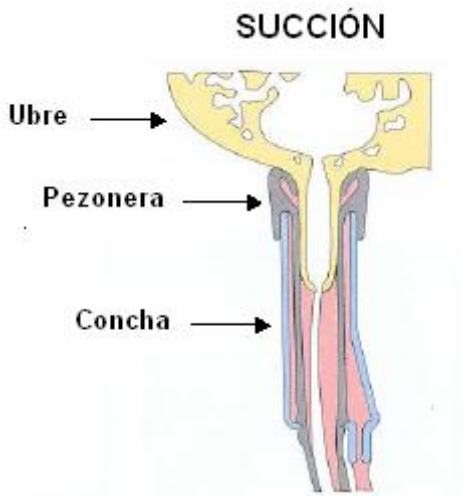


Figura 2.2 Proceso de succión

2.2. Tipos de sistemas de ordeño mecánico

2.2.1. Ordeño Mecánico Móvil

Este método consiste en una pequeña máquina móvil que dispone de todos los elementos necesarios para el ordeño y que se desplaza por el establo hasta los animales a ordeñar. Puede llevar incorporado el sistema de generación de vacío, o bien puede acoplarse en los diferentes puestos de ordeño a una conducción fija de vacío. Es un sistema sólo válido para explotaciones muy pequeñas, muy raras ya en la actualidad.

2.2.2. Ordeño Mecánico Fijo

Este método consiste en que el equipo está ubicado de forma fija en la sala de ordeño y son los animales los que se desplazan para el ordeño. Su eficacia es mucho mayor, son los que se van a encontrar en las explotaciones comerciales.

Dentro del ordeño fijo se puede encontrar con “Sistemas de ordeño a cantara”, donde la leche es recogida en este recipiente en la sala, y que hay que vaciar cada vez que se llena y los “Sistemas de ordeño directo”, donde la leche llega a un recipiente (Unidad final) que va de forma automática mandando la leche

al tanque de refrigeración. Este último sistema es más costoso pero lógicamente mucho más eficaz e higiénico, y es el que nos encontramos ya en la mayoría de las haciendas.

Dentro de los sistemas de ordeño fijo el último avance son los sistemas de ordeño voluntario, conocidos como “robots de ordeño”, que son unidades de ordeño donde la vaca acude voluntariamente a ser ordeñada, realizándose todas las operaciones de forma automática. Este sistema es aún muy minoritario por su elevado coste y limitado rendimiento (50 vacas / robot).

2.3. Elementos de la unidad de ordeño mecánico

En la Figura 2.3 se observa los componentes del ordeño mecánico, los cuales serán descritos a continuación:

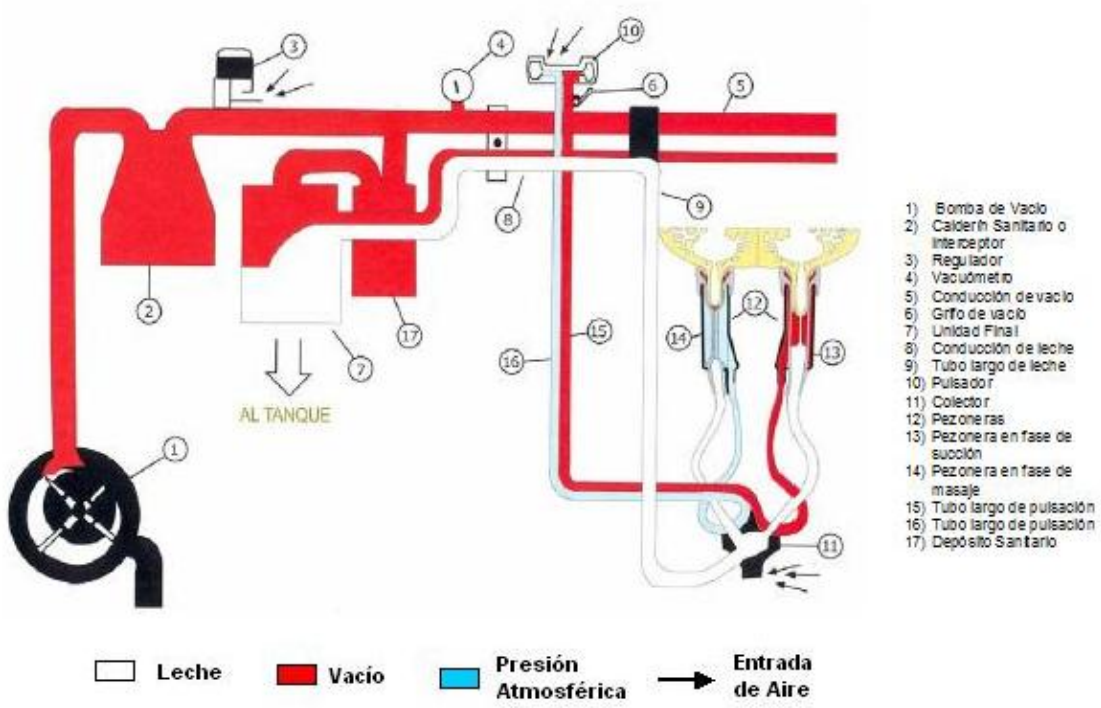


Figura 2.3 Elementos de la unidad de ordeño mecánico

2.3.1. Bomba de vacío

La bomba de vacío, en general, se compone de:

- **Cuerpo de Bomba**

El cuerpo de la bomba consiste en un cilindro hueco de fundición con un rotor excéntrico que aloja, normalmente cuatro paletas, que tienen un movimiento longitudinal. Al girar el rotor aspira las moléculas de aire del interior de la instalación, las comprime y lanza al exterior por el escape.

- **Escape**

Es importante que las bombas dispongan de un escape adecuado con la finalidad de amortiguar el ruido de la bomba, recuperar parte del aceite y llevar una válvula de no retorno para evitar que la bomba funcione a la inversa cuando se para el grupo motobomba.

- **Sistema de Lubricación**

Para que se desarrolle de correcta forma el proceso, la cámara en donde se comprime el aire es necesario que haya una fina película de aceite entre las paletas y las paredes de la cámara de compresión; esto se consigue mediante un sistema de lubricación que introduce el aceite en el interior de la cámara mediante diferentes sistemas, utilizando normalmente el propio vacío que se genera en la cámara de compresión.

Los depósitos de aceite normalmente están en la parte superior de la bomba y deben tener una parte transparente para vigilar el nivel del aceite. El aire cargado de aceite sale al exterior y es muy normal disponer en el escape de ciertas capas de láminas alternadas que hace que el aire choque contra ella y recupere parte del aceite por gravedad.

Debido a la mejora del comportamiento de los materiales de las paletas, actualmente hay bombas que no requieren lubricación, lo que antes sólo era habitual en bombas de un caudal muy bajo.

2.3.2. Calderín de vacío

Este componente también se llama interceptor y tiene varias misiones importantes como la de evitar que líquidos o cuerpos sólidos puedan entrar en la bomba, para lo que está equipado con una válvula de flotador guiada. Usualmente esta válvula consiste en una bola de goma o plástico que se ajusta al diámetro de la conducción de entrada de la bomba y que está dentro de una guía.

Si agua o leche entran en la instalación irían directamente al interceptor; cuando el nivel del agua, que mueve la bola hacía arriba, hiciera que la bola esté cerca de la entrada de la bomba, ésta sería succionada por el vacío de la bomba y cerraría el conducto de entrada impidiendo que el agua accediera a la bomba. El funcionamiento del interceptor se complementa con una válvula de drenaje automática que al cerrase el conducto de entrada de la bomba evacuaría el interceptor y a veces con una válvula de seguridad para proteger la bomba de los vacíos elevados que se pueden producir al cerrase la entrada de la misma.

La entrada de la conducción de aire al interceptor está situada de tal manera que las partículas sólidas que puedan venir por la instalación caigan a la parte inferior del interceptor, por lo que este componente deberá tener un sistema para que se pueda limpiar con facilidad.

2.3.3. Regulador

El regulador es un dispositivo automático diseñado para mantener un nivel vacío constante en la instalación durante el ordeño.

Hay que tener en cuenta que por el regulador pasa una gran cantidad de aire por lo que las entradas deben de llevar un filtro que hay que limpiar frecuentemente. También su correcto funcionamiento depende de que las piezas interiores estén limpias y en aquellos casos que tienen un dispositivo de regulación para diferentes niveles de vacío, este correctamente ajustado.

El regulador debe estar lo más cerca posible de las unidades de ordeño para que pueda equilibrar el nivel de vacío lo más rápidamente posible, por ello, se monta entre el interceptor y receptor, en éste o incluso en el depósito sanitario o entre ambos; en estos últimos casos sólo pueden colocarse sensores que cumplan los requisitos higiénicos.

Según sea el sistema de funcionamiento del regulador puede ser:

- **De muelle**

  Los reguladores de muelle mantienen el nivel de vacío mediante la fuerza que ejerce un muelle y son los que primero se utilizaron. Tiene el inconveniente que cuando va pasando el tiempo las características mecánicas del muelle cambian, y por ello, acaban por no regular muy bien el vacío; suelen actuar rápidamente cuando hay variaciones de vacío, pero tardan bastante tiempo en estabilizar otra su funcionamiento. Por ello, actualmente son muy poco usados.

- **De peso muerto**

  Los reguladores de peso muerto se han utilizado mucho, ya que son muy estables al utilizar un peso para regular el vacío. Tienen el problema, a diferencia de los de muelle y de los servo, de que para que funcionen bien tienen que estar equilibrados, lo que es difícil en una instalación como la de ordeño sometida a vibraciones. Se utilizaron mucho, aunque ahora han sido sustituidos por los de tipo servo.

- **De tipo servo con motor**

Los reguladores de tipo servo son los que más se utilizan a ser muy fiables, actúan muy rápido con las variaciones de vacío y no es trascendente que estén equilibrados. Funcionan con una combinación de membranas y pesos, y suelen tomar el vacío en otro punto diferente al que está montado el regulador mediante un sensor para evitar interferencias por las turbulencias que se producen en la entrada de aire al regulador.

2.3.4. Vacuómetro

Es un instrumento, normalmente mecánico, que mide el vacío (depresión) a que está sometido el aire en el interior de la instalación. Los vacuómetros son medidores de presión diferencial (vacío), es decir, de diferencia de presiones entre el interior y el exterior de la instalación. Los vacuómetros utilizados en las instalaciones de ordeño mecánico miden siempre el vacío en kilopascales y generalmente llevan una indicación del nivel de vacío recomendado y de cuándo el nivel de vacío pudiera ser peligroso.

2.3.5. Conducción de aire o Tubería de vacío

El aire que aspira la bomba procedente de toda la instalación viene por las conducciones de aire y vacío que conectan la bomba con el resto de componentes de una instalación. Hay varias tuberías de aire y una de vacío, que se llama “Conducción de vacío de ordeño”; en ambos casos, por estas tuberías sólo circula aire y nunca la leche.

Estas conducciones pueden ser de hierro fundido galvanizado, acero o, modernamente, de PVC. En algún caso especial, tal como en la conducción de aire del receptor, ésta puede ser de cristal o plástico transparente.

Las conducciones de aire son las siguientes:

- **Conducción principal de aire**

Situada entre la bomba de vacío y el depósito sanitario (Ver Figura 2.4).

- **Conducción de aire de pulsación**

Conecta la conducción principal de aire y los pulsadores (Ver Figura 2.5).

- **Conducción de aire del receptor**

Conecta el depósito sanitario con el receptor.

La conducción de vacío de ordeño es la conducción situada entre el depósito sanitario y las unidades de ordeño en determinadas instalaciones (máquinas con depósito sanitario y con conducciones de aire y leche independientes).

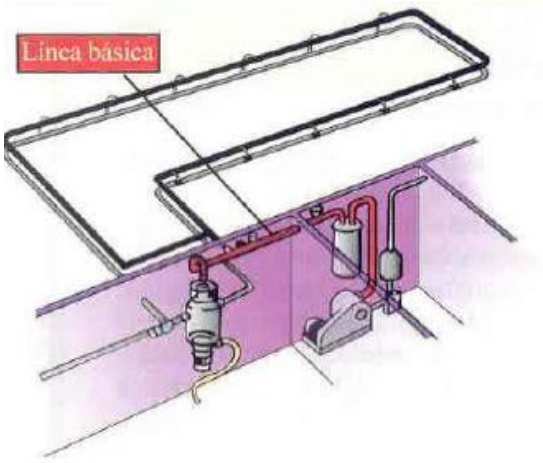


Figura 2.4 Conducción principal de vacío

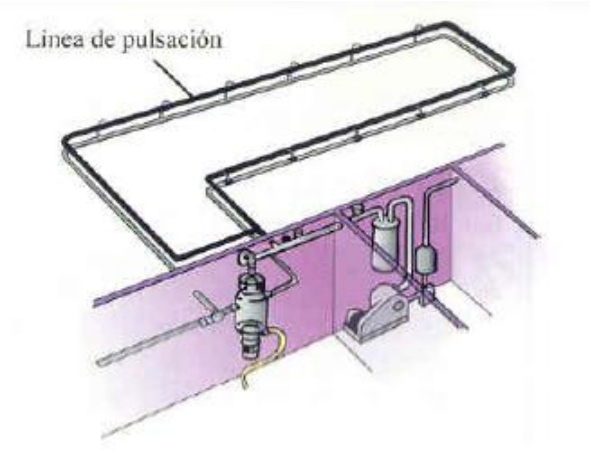


Figura 2.5 Conducción de Vacío de pulsación

2.3.6. Grifo de Vacío

Es una válvula, manual o automática, que permite la conexión y desconexión de las unidades de ordeño y otros dispositivos de que funcionan con vacío, al sistema de vacío. Estos grifos están montados en las conducciones de aire y vacío.

2.3.7. Pulsador

El pulsador es el dispositivo que alternadamente, según el ciclo de pulsación[9], dejar entrar aire o vacío en la cámara de pulsación.

Las partes básicas que tienen todos los pulsadores son:

- Conexión con las conducciones de aire de pulsación por las que se extrae el aire de la cámara de pulsación y se llega al nivel de vacío deseado en la fase de ordeño.

- Una válvula o corredera que se mueve mediante el vacío o corriente eléctrica.

- Conexión con el aire exterior para pasar éste a la cámara de pulsación en la fase de masaje.

- Conexión con el tubo largo de pulsación hacia la unidad de ordeño.

El funcionamiento del pulsador consiste en una corredera que se mueve alternadamente conectando el tubo largo de vacío y la cámara de pulsación con el sistema de vacío o con el aire atmosférico, de tal manera que el aire que está en la cámara de pulsación está a la presión atmosférica (masaje) o bajo vacío (succión).

[9] Comprende el periodo de succión y de masaje

El número de veces por minuto que se repite este ciclo se llama frecuencia de pulsación y la relación de pulsación es el tiempo de succión / tiempo de masaje, en tanto por ciento. Las frecuencias de pulsación más normales en vacuno están comprendidas entre 50 y 60 ciclos/min.

Hay dos maneras de mover la corredera o válvula que da lugar a dos tipos de pulsadores diferentes:

- **Pulsadores neumáticos**

  Los pulsadores neumáticos utilizan el propio vacío de la instalación para funcionar y son de funcionamiento relativamente simple. Estos pulsadores tienen la gran ventaja de que no necesitan elementos externos para su funcionamiento externo diferente del vacío, aunque se utilizan cada vez menos.

- **Pulsadores eléctricos**

  Los pulsadores eléctricos accionan la válvula mediante energía eléctrica de bajo voltaje y su funcionamiento es muy preciso ya que puede programarse electrónicamente. Su gran problema es que dependen del suministro de energía eléctrica o baterías que puede fallar en el primer caso o descargarse en el segundo.

Los pulsadores llevan entradas de aire con filtros que hay que limpiar periódicamente para garantizar su buen funcionamiento.

2.3.8. Pezoneras

En la Figura 2.6 se puede observar el esquema de una pezonera, el cual comprende de una copa rígida, de metal o material plástico o una combinación de ambos materiales, y un mango de ordeño flexible que esta ajustado en ambas partes de la copa. Entre el mango y la copa queda una cámara que se llama de pulsación, aquella que alternadamente está con presión atmosférica y vacío.

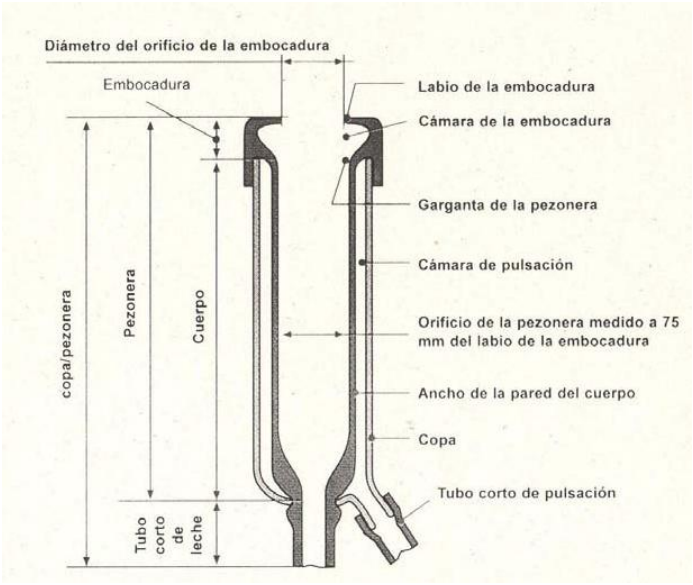


Figura 2.6 Esquema de pezonera[10]

El mango de ordeño puede ser de goma, caucho o silicona. En todos los caso es flexible para permitir el paso de la leche en la fase de succión y dar masaje al pezón en la fase de masaje.

Como el mango debe ajustar perfectamente en la copa existe una compatibilidad entre estos dos componentes, es decir que no todos los mangos valen para cualquier copa. La pezonera frecuentemente incluye al tubo corto de leche.

[10] Componentes del conjunto copa/pezonera (ISO 3918)

2.3.9. Colector

Pieza, generalmente de acero, plástico o un combinación de estos dos materiales (Ver Figura 2.7), que reúne las pezoneras y que tiene una cámara interior, con más o menos volumen, para recoger la leche y enviarla a través del tubo largo de leche.

Además, el colector es el asiento de la cámara de distribución de la pulsación.

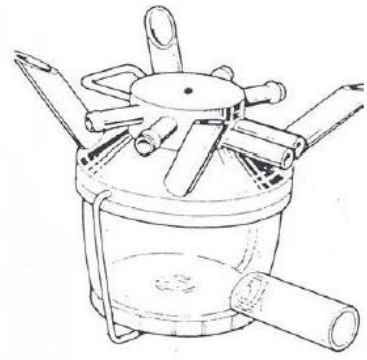


Figura 2.7 Colector

2.3.10. Tubos cortos de leche

Conecta la boquilla del colector al cuerpo del mango de ordeño, a un conector o a un visor.

2.3.11. Tubos cortos de pulsación

Tubo que conecta la cámara de pulsación de la copa y el distribuidor de pulsación.

2.3.12. Tubos largos de leche

Es un tubo de goma o material plástico que evacua la leche desde el colector a la conducción de leche, al depósito medidor.

2.3.13. Tubos largos de pulsación

Tubo que conecta el pulsador con la boquilla de aire del colector para transmitir el vacío.

2.3.14. Conducción de leche

Conducción que trasporta leche y aire durante el ordeño, y que tiene la doble misión de proporcionar vacío para el ordeño y de llevar la leche al receptor. Estas conducciones suelen ser de cristal, materiales plásticos alimentarios o acero inoxidable.

Las conducciones de leche pueden ser:

- **Simple**

La Figura 2.8 muestra la conducción de leche simple, que consiste en una conducción que está cerrada en su extremo más alejado por un obturador o tapón y con el extremo más próximo conectado al receptor.

Figura 2.8 Conducción de leche simple

- **En anillo**

La Figura 2.9 muestra la conducción de leche en anillo, que consiste en una conducción que forma un circuito cerrado con dos entradas al receptor.

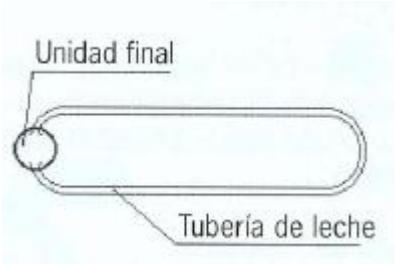


Figura 2.9 Conducción de leche en anillo

Todas las conducciones de leche deben tener una pendiente descendente hacia el receptor facilitando el movimiento de la leche.

2.3.15. Unidad final

Recoge la leche de la conducción y la va mandando al tanque de frío. Está compuesta por el receptor que es un depósito de vidrio o acero inoxidable, donde desemboca la conducción de leche y está conectado con la tubería de vacío.

2.3.16. Depósito sanitario

Colocado entre el receptor y la tubería de vacío, es un dispositivo de seguridad que evita que la leche llegue a la tubería de vacío.

2.3.17. Bomba de Impulsión

Se emplea para vaciar el receptor y enviar la leche al tanque de refrigeración

2.4. Frecuencia de Ordeño

En ganado vacuno, en función de su capacidad de producción, es necesario ordeñar como mínimo 2 veces al día con un intervalo lo más similar posible. Lo más usual es:

- 12h/12h (por ejemplo ordeño de mañana 6 h y ordeño de tarde 18 h)
- 10h./14h (por ejemplo ordeño de mañana 8 h y ordeño de tarde 18 h)

En caso del triple ordeño diario los intervalos se deben acercarlo máximo a 8h./8h./8h.

Cuando se realizan los cuatro ordeños en los lotes de alta producción el intervalo entre 1º y 2º ordeño, y entre 3º y 4º para estos animales deben ser como mínimo de 2h, mientras que el intervalo entre los dos ordeños principales de mañana y tarde de 12 h.

2.5. Tipos de Sala de Ordeño

2.5.1. Tandem

Se trata de un sistema de reducida difusión que se caracteriza por disponerse las vacas, una detrás de otra en jaulas individuales. Cada una de estas posee entrada y salida independiente. Las jaulas se pueden disponer a ambos lados de una fosa central (Ver Figura 2.10).

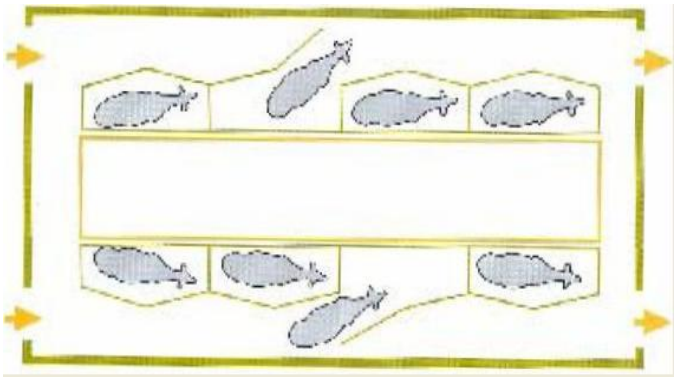


Figura 2.10 Sala de Ordeño Tandem

Este sistema fue concebido para brindar un ordeño caracterizado por un trato individual a cada vaca. Un operador puede trabajar con 2 o 3 grupos de

pezoneras y el rendimiento es de 4 a 6 vacas ordeñadas por punto de ordeño y por hora. Se recomienda la colocación de un punto de ordeño o bajada por jaula.

Presenta el inconveniente de un elevado número de metros cuadrados cubiertos por punto de ordeño y un elevado costo de las jaulas de ordeño. La desventaja primordial es que se ocupa mucho espacio y la instalación y/o construcción de la jaula forzosamente debe ser de tubería con un pasillo de ingreso y otro para la salida del animal terminado de ordeñar.

Tiene la ventaja de dar tratamiento individual a cada vaca lo cual representa que no es necesario homogeneizar los grupos de vacas por velocidad de ordeña.

2.5.2. Espina de Pescado

La sala de ordeño más empleada hoy día es la “espina de pescado”, en donde las vacas entran en grupos y se paran ligeramente en ángulo (Ver Figuras 2.11 y 2.12), de manera que solamente su parte posterior queda expuesta al ordeñador, que está en el foso. Esto acorta la distancia entre las ubres y reduce el tiempo de traslado de los ordeñadores.

El manejo en grupos facilita el movimiento de las vacas, pero una sola vaca que sea ordeñada lentamente retrasará a todo el grupo. Esto es particularmente perjudicial cuando se trata de espinas de pescado de gran tamaño, de 10 o más vacas por lado. El tiempo de salida de un grupo grande, consume también mucho tiempo.

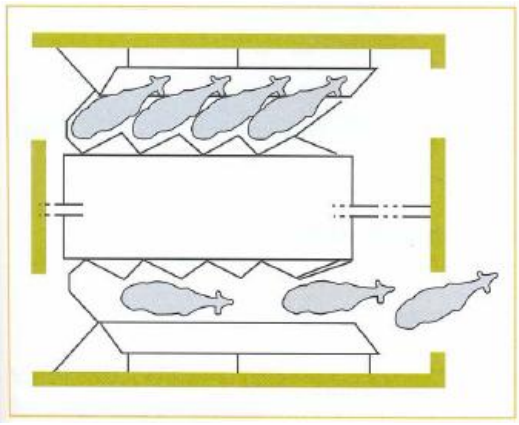


Figura 2.11 Sala de ordeño espina de pescado con barra trasera de contención semi-sinuosa

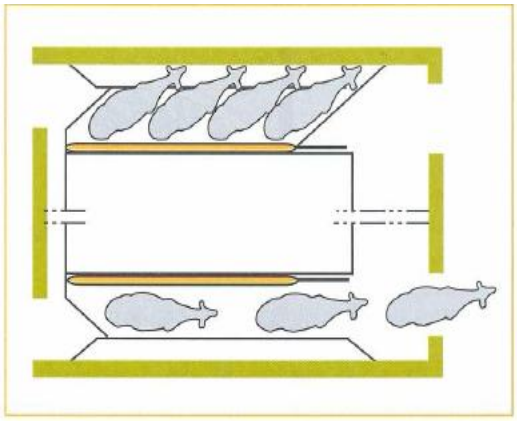


Figura 2.12 Sala de ordeño espina de pescado con barra trasera de contención recta

Este sistema es recomendable para establos grandes, mayores a las 100 vacas.

Sus principales ventajas son las siguientes:

- Mayor rendimiento al producirse la entrada y salida de las vacas en forma colectiva.

- Mayor comodidad del operador, ya que trabaja parado.

- Posibilidad de ampliación.

Sus principales desventajas son las siguientes:

- Construcción más costosa que los sistemas anteriores, dado que se debe fabricar una fosa en desnivel para que trabaje el operador.

- Es más difícil adaptar alguna construcción ya existente.

- La velocidad de ordeño está limitada por la vaca más lenta de la tanda.

- Trato colectivo (es necesario homogeneizar en velocidad de ordeño el grupo de vacas que ingresa en la línea).

2.5.3. Paralelo

Las vacas se colocan perpendiculares al foso (Ver Figura 2.13), haciéndose el ordeño por detrás, suelen ser de línea doble con un punto de ordeño por puesto y salida rápida de los animales; también es común que dispongan de retirada automática de pezoneras. Se recomiendan en rebaños grandes de más de 100 vacas

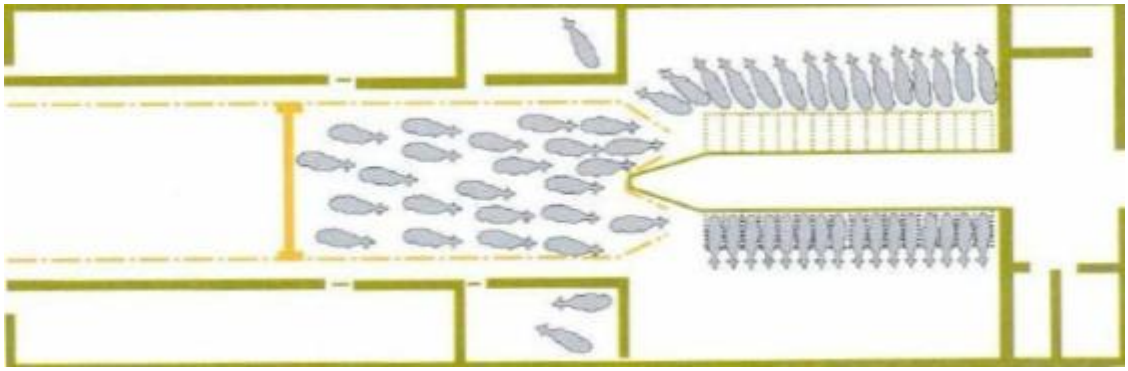


Figura 2.13 Sala de Ordeño en Paralelo

2.5.4. Rotativas

Las vacas entran y el operario no se desplaza ya que disponen de una plataforma circular rotatoria que acerca las vacas al ordeñador, requieren al menos dos operarios, uno pone pezoneras y otro sella pezones tras el ordeño.

Existen salas rotativas con ordeño exterior (Ver Figura 2.14), más comunes, y con ordeño interior (Ver Figura 2.15). Pueden ser de distintas dimensiones que van desde los 20 a los 48 puestos y puntos de ordeño; siempre van equipadas con retirada automática. Debido a su coste sólo son recomendables en explotaciones de gran tamaño (más de 250 vacas en ordeño)

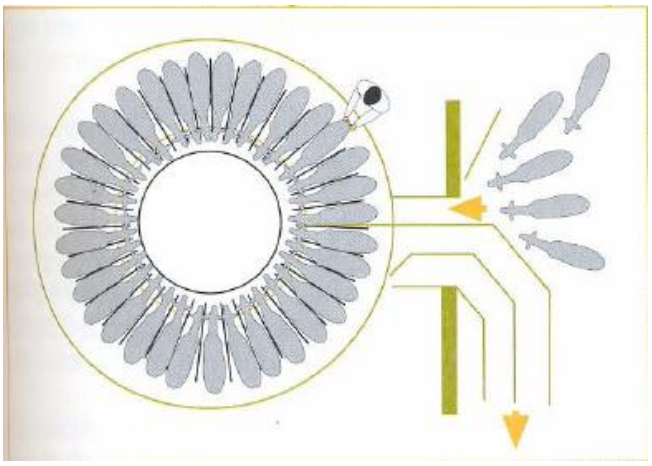


Figura 2.14 Sala de Ordeño Rotativa con ordeño exterior

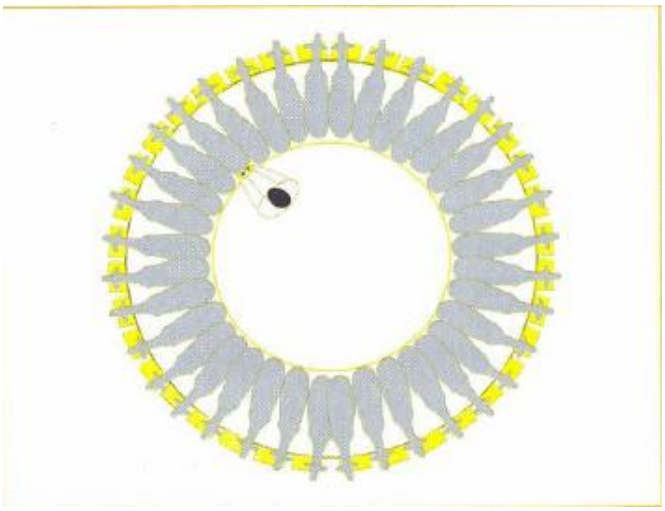


Figura 2.15 Sala de Ordeño Rotativa con ordeño interior

2.6. Ventajas y Desventajas del Ordeño Mecánico

2.6.1. Ventajas

- Mayor eficiencia de la mano de obra, se ordeñan más vacas por hora hombre en comparación al ordeño manual. Siendo esto de importancia donde existe escasez de personal.

- Se reducen los requerimientos de personal debido a la mayor eficiencia de la mano de obra, obteniéndose más kilogramos de leche por hombre al año.

- Se reducen los problemas de personal. El ausentismo no causa problemas tan serios como en el caso del ordeño manual, puesto que el trabajo del ordeñador ausente es fácilmente realizable por otra persona familiarizada con las máquinas para ordeño.

- Mejores condiciones para controlar la higiene de la leche. Se evita el contacto de la leche con el medio ambiente, lo que reduce las posibilidades de contaminación.

- Ofrece condiciones más favorables para los ordeñadores puesto que el esfuerzo físico es menor

2.6.2. Desventajas

- Se requiere una inversión elevada en equipos y obra civil.

- Si los equipos adolecen de fallas mecánicas y no son manejados con cuidado, el sistema puede resultar contraproducente y afectar seriamente la salud de la glándula mamaria.

- Se requiere capacitar al personal para manejar en forma cuidadosa y eficiente el equipo.

- Cierto porcentaje de animales con defectos anatómicos de la ubre, no puede adaptarse a esta forma de ordeño.

CAPÍTULO 3

3. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE RECOLECCIÓN Y ENVÍO DE LA CANTIDAD DE LECHE (SISTEMA TRANSMISOR)

3.1. Introducción

El sistema de recolección y envío de la cantidad de leche consiste en sensar la cantidad de leche que extrae el sistema de ordeño mecánico a través de sensores; los valores obtenidos de los sensores serán procesados por un sistema de control (microcontrolador), el cuál mediante un circuito integrado reloj-calendario obtendrá los valores de los sensores en dos horas específicas del día, posteriormente, estos valores serán trasmitidos mediante un modem GSM en forma de SMS a un centro de gestión (Sistema Receptor). La Figura 3.1 muestra el esquema del Sistema Transmisor:

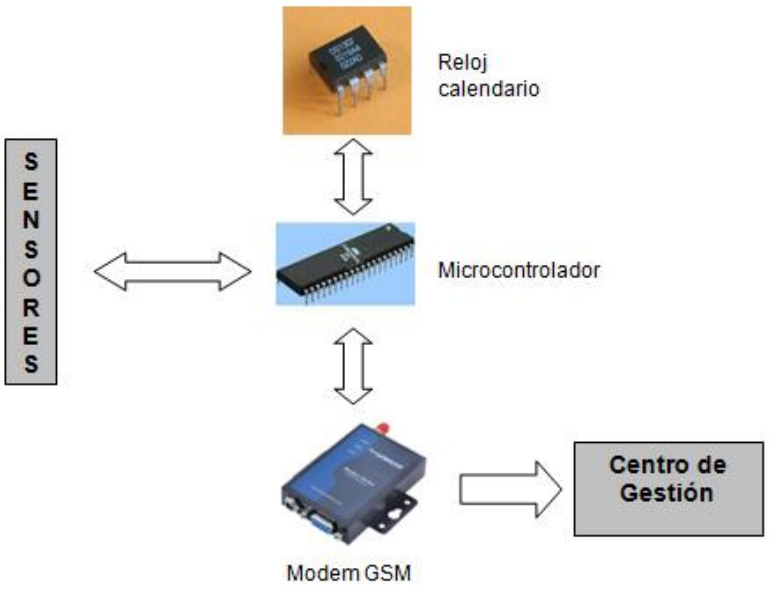


Figura 3.1 Esquema del Sistema Transmisor

3.2. Diagrama de Bloques del sistema transmisor

En la Figura 3.2, se puede observar el Diagrama de Bloques del Sistema Transmisor:

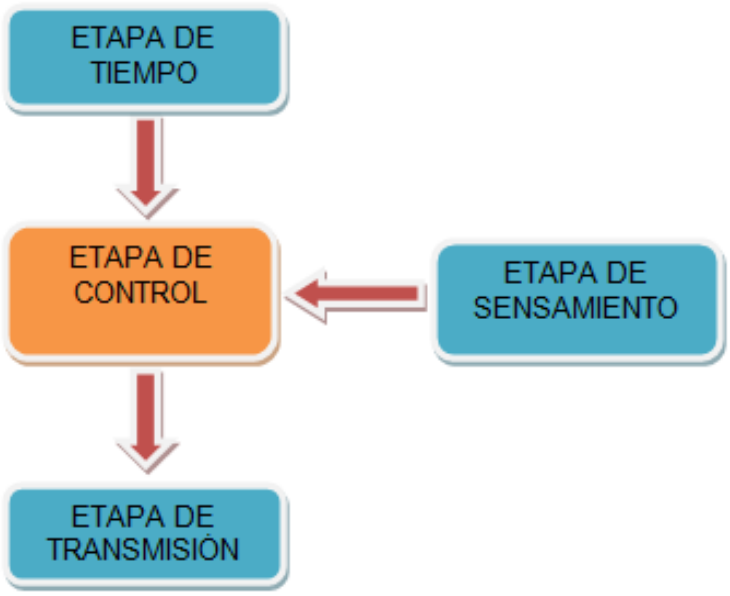


Figura 3.2 Diagrama de Bloques del Sistema Transmisor

3.2.1. Etapa de Tiempo

Esta etapa consiste en un reloj-calendario, el cual proporciona la hora y fecha en tiempo real. Este dispositivo de tiempo real es capaz de mantener en forma automática un conteo completo de hora y fecha, incluso con compensación de año bisiesto. El reloj de tiempo real que se utilizará para esta etapa será el modelo DS1307, el cual es fabricado por Dallas semiconductor y se comunica a través del protocolo I2C. El RTC es un dispositivo de uso muy común en aplicaciones con microcontroladores.

A continuación se describirá los principales parámetros del Reloj Calendario DS1307 (RTC, Reloj en tiempo real):

- **Características**

  - El DS1307 es un RTC serial que procesa la información de los segundos, minutos, horas, día del mes, día de la semana, mes y año.

  - El ajuste para la duración de los meses, incluso en los años bisiestos, es realizado por el propio circuito y es válido hasta el año 2100.

  - Contiene 56 bytes de NVRAM (memoria RAM no volátil)

  - Permite la operación en modo de 24 o 12 horas.

  - Al presentarse fallas en la alimentación (Vcc) del DS1307 cambia automáticamente al modo de operación con la batería para no perder su configuración. Su consumo, en esta condición, es menor de 500 nA.

  - Puede generar una señal de onda cuadrada de frecuencia programable.

- **Esquema**

La Figura 3.3 muestra el esquema del reloj calendario DS1307, a continuación se describirá los pines:

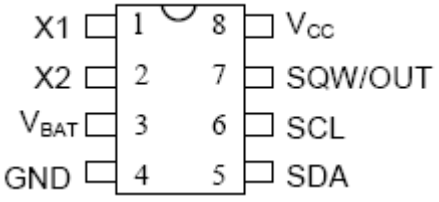


Figura 3.3 Esquema del reloj calendario DS1307

- Los terminales Vcc (Pin 8) y GND (Pin 4), proveen la tensión necesaria para el funcionamiento del dispositivo. Generalmente los niveles de voltaje de estos pines son: +5V DC y 0V.

- V_{BAT} (Pin 3): Proporciona un medio para garantizar que el DS1307 mantenga su configuración ante una pérdida de alimentación de Vcc. Funciona a través de una batería externa de 2V a 3,5V DC.

- Cristal X1, X2 (Pin 1 y Pin 2): Para el funcionamiento correcto el dispositivo necesita de un oscilador de 32,768kHz el mismo que va en los Pines 1 y 2.

- SDA (Pin 5): Es el pin por donde van a fluir los datos desde el reloj hacia el microcontrolador.

- SCL (Pin 6): Es el pin por el cual se va a sincronizar con el microcontrolador para poder enviar los datos de acuerdo a la señal de reloj que el circuito master genere.

- **Registros**

Los registros de función específica se encuentran ubicados en los ocho primeros bytes de una memoria total 256 bytes. De estos ocho registros, los siete primeros (dirección 00H a 06H), se utilizan para almacenar la información del rejo/calendario en formato BCD, mientras que el registro 07H se emplea como registro de control. En la Tabla 3.1 se muestra la distribución de los registros internos del DS1307.

Tabla 3.1 Registros del reloj Calendario DS1307

Direccionamiento	Registro
00	Segundos (0 a 59)
01	Minutos (0 a 59)
02	Horas (0 a 23)
03	Día de la semana (01 a 07)
04	Día (01 a 31)
05	Mes (01 a 12)
06	Año (01 a 99)
07	Control
08	Propósito General RAM

Todos los datos de tiempo y fecha están en formato BCD, lo cual hace muy fácil su lectura y escritura usando notación hexadecimal. Por ejemplo 07:30 a.m. va a contener $07 en el registro de horas y $30 en el registro de minutos.

- **Interfaz de comunicación**

El reloj calendario DS1307 se comunica con los microcontroladores a través de la interfaz serial I2C, a continuación se describirá las características y el funcionamiento de la interfaz I2C:

El bus I2C, es un estándar que facilita la comunicación entre microcontroladores, memorias y otros dispositivos electrónicos. Requiere de dos líneas de señal y un común (Tierra). Fue diseñado a este efecto por Philips y permite el intercambio de información entre muchos dispositivos a una velocidad aceptable, de unos 100 kbits por segundo, aunque hay casos especiales en los que el reloj llega hasta los 3,4 MHz.

Descripción de las señales

- SCL (*System Clock*): Es la línea de los pulsos de reloj que sincronizan el sistema.

- SDA (*System Data*): Es la línea por la que se mueven los datos entre los dispositivos.

- GND (Masa): Común de la interconexión entre todos los dispositivos conectados al bus.

Las líneas SDA y SCL son del tipo drenaje abierto, es decir, un estado similar al de colector abierto, pero asociadas a un transistor de efecto de campo (FET). Se deben polarizar en estado alto[11], lo que define una estructura de bus que permite conectar en paralelo múltiples entradas y salidas.

Protocolo de comunicación del bus I2C

Habiendo varios dispositivos conectados sobre el bus, es lógico que para establecer una comunicación a través de él se debe respetar un protocolo. En primer lugar, lo más importante: existen dispositivos maestros y dispositivos esclavos. Sólo los dispositivos maestros pueden iniciar una comunicación.

La condición inicial (Ver Figura 3.4), de bus libre, es cuando ambas señales están en estado lógico alto. En este estado cualquier dispositivo maestro puede ocuparlo, estableciendo la condición de inicio (*start*). Esta condición se presenta cuando un dispositivo maestro pone en estado bajo la línea de datos (*SDA*), pero dejando en alto la línea de reloj (*SCL*).

[11] Conectando a la alimentación por medio de resistencias "*pull-up*"

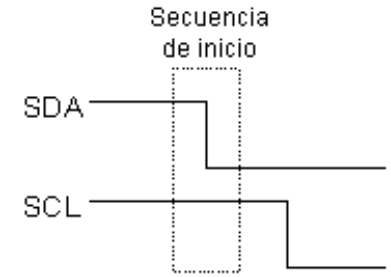


Figura 3.4 Condición de Inicio

La Figura 3.5 muestra los bits del Bus I2C, el primer byte que se transmite luego de la condición de inicio contiene siete bits que componen la dirección del dispositivo que se desea seleccionar, y un octavo bit que corresponde a la operación que se quiere realizar[12].

Si el dispositivo cuya dirección corresponde a la que se indica en los siete bits (A0-A6) está presente en el bus, éste contesta con un bit en bajo, ubicado inmediatamente luego del octavo bit que ha enviado el dispositivo maestro. Este bit de reconocimiento (*ACK*) en bajo le indica al dispositivo maestro que el esclavo reconoce la solicitud y está en condiciones de comunicarse. Aquí la comunicación se establece en firme y comienza el intercambio de información entre los dispositivos.

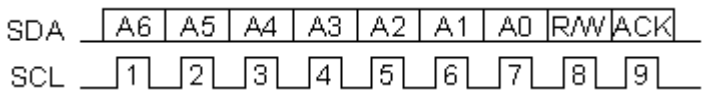


Figura 3.5 Bits del Bus I2C

Si el bit de lectura/escritura (*R/W*) es puesto en esta comunicación a nivel lógico bajo (escritura), el dispositivo maestro envía datos al dispositivo esclavo. Esto se mantiene mientras continúe recibiendo señales de reconocimiento, y el contacto concluye cuando se hayan transmitido todos los datos.

[12] Operación de lectura o escritura.

Si el bit de lectura/escritura (R/W) fue puesto en esta comunicación a nivel lógico bajo (escritura), el dispositivo maestro envía datos al dispositivo esclavo. Esto se mantiene mientras continúe recibiendo señales de reconocimiento, y el contacto concluye cuando se hayan transmitido todos los datos.

En el caso contrario, cuando el bit de lectura/escritura esta en nivel lógico alto (lectura), el dispositivo maestro genera pulsos de reloj para que el dispositivo esclavo pueda enviar los datos. Luego de cada byte recibido el dispositivo maestro (quien está recibiendo los datos) genera un pulso de reconocimiento.

El dispositivo maestro puede dejar libre el bus generando una condición de parada (Ver Figura 3.6).

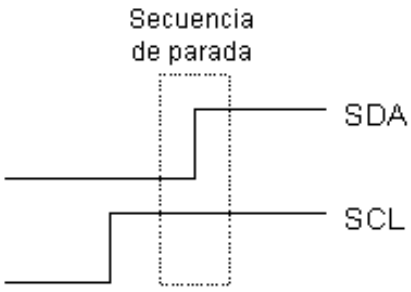


Figura 3.6 Condición de Parada

3.2.2. Etapa de Sensamiento

Esta etapa consiste en utilizar sensores, los cuales se encarga de medir la cantidad exacta de leche recolectada en el ordeño mecánico. El Sistema Transmisor puede manejar varios sensores (flujo, caudal, nivel, temperatura, etc.), En el diseño del Sistema Transmisor se procedió a utilizar un sensor de nivel ultrasónico, a continuación se explicará que clase de sensor se utilizará:

- **Sensor de Nivel**

Para la obtención del volumen de leche se ha determinado la utilización de un sensor de nivel ultrasónico, mediante este sensor se obtiene el nivel del líquido y conociendo la forma del Tanque de Almacenamiento se puede obtener el volumen de leche. Este sensor será ubicado en el tanque de almacenamiento final (depósito de refrigeración), en donde se obtendrá la medida de la leche contenida en el tanque.

En aplicaciones de medición de nivel en las que no existe contacto entre el instrumento con el líquido del proceso, una buena opción son los dispositivos de sonido o ultrasonido. Estos instrumentos miden la distancia entre un punto en el recipiente (usualmente un punto de referencia) y la interfaz de nivel del fluido. En general, los principios de operación de los dispositivos sónicos y ultrasónicos son similares: La vibración de un dispositivo causa que los objetos cercanos también vibren y esta transferencia de vibración o movimiento a través de un medio es sonido, que viaja en un medio dado en forma de onda con una frecuencia y velocidad características. Una medida de nivel ultrasónica aprovecha estas propiedades.

Debido a que el sonido viaja a una velocidad constante, el tiempo entre la ráfaga de sonido transmitida y la detección del eco de retorno será proporcional a la distancia entre el sensor y el dispositivo reflector. Por esto, se puede calcular la distancia entre ambos por la relación:

Distancia = Velocidad x Tiempo

Donde Velocidad es la del sonido en el aire y Tiempo es la mitad del tiempo desde la transmisión hasta la detección del eco.

En la figura 3.7 se puede observar como el sensor de nivel ultrasónico actúa sobre el tanque o depósito sobre el cual se desea obtener la medida de nivel.

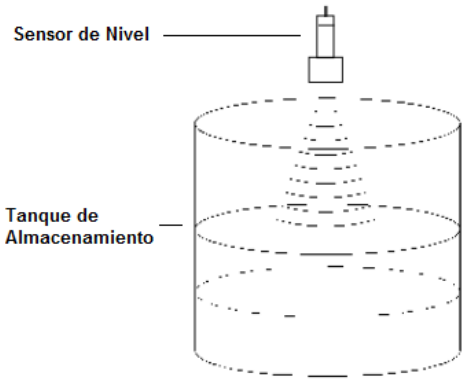


Figura 3.7 Tanque de almacenamiento y sensor de nivel ultrasónico

Para obtener una correcta medida con el sensor de nivel ultrasónico es muy importante la forma de colocación del sensor sobre el tanque o depósito. En la figura 3.8 se explica la correcta e incorrecta forma de colocación del sensor.

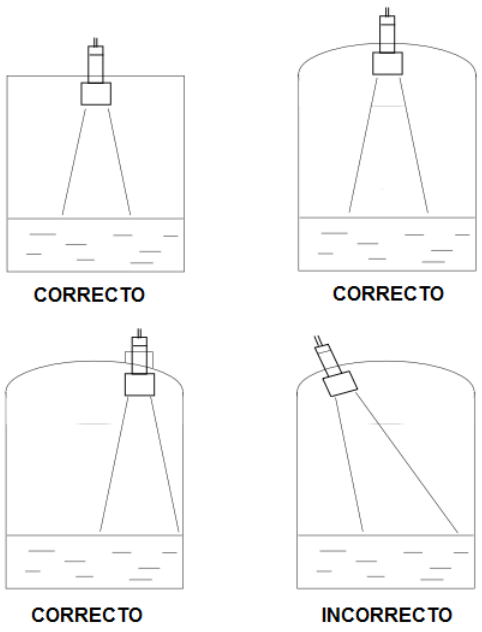


Figura 3.8 Correcta e Incorrecta forma de colocación del sensor de nivel

Para la implementación de la etapa de sensamiento se ha determinado la utilización del sensor de nivel ultrasónico PING, a continuación se describirá las características del sensor:

SENSOR PING (Ultrasonido)

Figura 3.9 Sensor PING

El sensor ultrasónico de distancia PING (Figura 3.9) permite efectuar la medición de distancia de objetos colocados entre 3 cm y 3 m, es fácil de conectar y requiere únicamente para su operación un terminal de entrada /salida del microcontrolador.

El funcionamiento del sensor se basa en la utilización de ondas ultrasónicas, las cuales se caracterizan porque su frecuencia supera la capacidad de audición de los seres humanos. El oído humano es capaz de detectar ondas sonoras de frecuencias comprendidas entre unos 20 y 20000 Hertz, a esto se le conoce como espectro audible. Toda señal sonora que se encuentre por encima de este rango, se cataloga como ultrasónica.

El sensor ping transmite una ráfaga ultrasónica y mide el tiempo que demora el eco en ser recibido. Este eco se produce cuando las ondas sonoras golpean un objeto que se encuentra dentro del rango de medición del sensor PING.

El sensor PING entrega una salida en forma de un pulso digital que es proporcional al tiempo requerido por el ultrasonido para ir desde el módulo emisor, golpear contra un objeto y regresar hasta el receptor. Para lograr que el microcontrolador obtenga la medición de distancia de un objeto colocado frente al sensor, basta con medir la duración de este pulso y aplicar un cálculo para obtener el resultado. Este sensor es una buena elección para aplicaciones donde se requiera efectuar la medición de distancia entre objetos fijos o móviles.

Características Técnicas

- Voltaje de Alimentación = 5 VDC.
- Consumo de Corriente = 30 – 35 mA (máx).
- Rango de medición = 3 cm hasta 3 m.
- Entrada de disparo = Pulso ascendente TTL con duración mínima de 5us.
- Pulso de salida = Pulso ascendente TTL comprendido entre 115 us y 18.5 ms.
- Tiempo de espera para la medición = 750 us luego del pulso de disparo.
- Frecuencia del ultrasonido = 40 kHz.
- Tiempo de emisión del ultrasonido = 200 us.
- Diodo LED indicador de actividad.
- Tiempo mínimo de espera entre medidas = 200 us.
- Dimensiones = 22x46x16 mm.

Terminales de Conexión del Sensor

Este dispositivo cuenta únicamente con tres terminales de conexión separados por una distancia estándar de 0,1”, lo cual hace su inserción en protoboards o en circuitos impresos.

La Figura 3.10, muestra la función de los terminales del sensor. GND se utiliza como referencia o tierra, Vcc (5V) provee la alimentación para los circuitos internos del sensor y PIN E/S es el terminal de entrada/salida utilizado para producir el pulso de activación o disparo y recibir la medición efectuada.

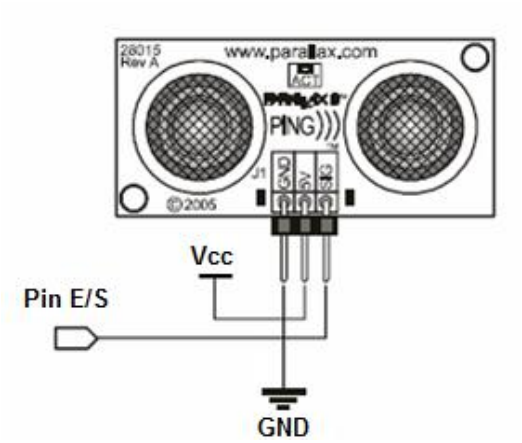


Figura 3.10 Terminales de conexión

Funcionamiento del Sensor

El microcontrolador debe garantizar que exista un estado bajo en el pin de señal del sensor antes de comenzar la operación. Seguidamente se genera un pulso de activación o de disparo. Al concluir ese pulso, el terminal de E/S del microcontrolador conectado a pin de señal debe convertirse en una entrada para permitir que el sensor PING tome el control del mismo. El sensor activa al transmisor de ultrasonido durante unos 200us enviando una ráfaga a 40 kHz. Esta ráfaga viaja en el aire a una velocidad aproximada de 1239,93 Kms/hora golpea al objeto en frente del sensor y se genera una señal rebote que es “escuchada” por el micrófono para ultrasonidos del módulo. El terminal E/S se colocará en estado alto luego de ser enviada la ráfaga de 40 kHz y permanecerá de esa manera por un tiempo comprendido entre 115 us y 18,5ms.

Para efectuar el cálculo de la distancia debe considerarse que el pulso recibido tiene una duración proporcional al doble de la distancia recorrida por la onda sonora. Para comprender esto hace falta analizar la figura 3.11, en esta se observa que la señal demora un tiempo T1 en alcanzar al objeto y posteriormente toma un tiempo T2 en llegar de regreso al sensor PING. Ya que ambas ondas se propagan por el mismo medio (aire) los tiempos T1 y T2 serán iguales.

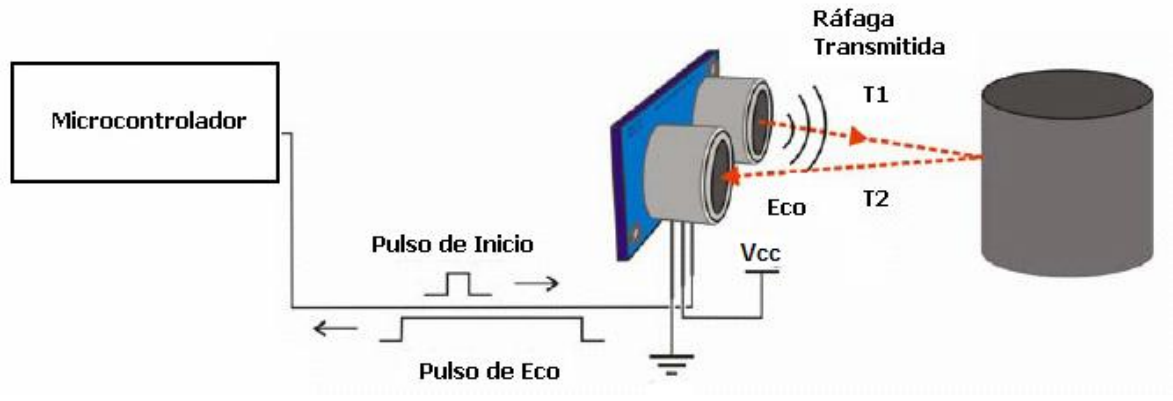


Figura 3.11 Funcionamiento del Sensor

Calculo para generar el pulso de inicio del sensor mediante el software BASCOM

Para conocer el pulso de inicio del sensor PING es necesario realizar los siguientes cálculos:

- Se conoce:

$$Fclock = 11{,}059200MHz \rightarrow cristal\ del\ microcontrolador$$
$$Pulso = T(us) \times Fclock\,(MHz)$$

- El pulso de inicio del sensor necesita un periodo de 5 uS, por lo tanto:

$$T(uS) = 5us$$
$$Pulso = 5us \times 11{,}059200MHz$$
$$Pulso = 55{,}29 \rightarrow Pulso\ a\ ser\ generado\ por\ el\ microcontrolador$$

Calculo del pulso recibido del Sensor PING para obtener la distancia mediante el Software BASCOM

Una vez obtenido el pulso de respuesta del sensor PING, se tiene que realizar los siguientes cálculos:

- Antes de calcular la distancia es necesario calcular la velocidad del sonido a 25°C:

$$V_{sonido} = 331{,}5 + (0{,}6 \times T_{ambiente})m/s$$
$$T_{ambiente} = 25°C$$
$$V_{sonido} = 331{,}5 + (0{,}6 \times 25)m/s = 346{,}5m/s = 34650cm/s$$

- El cálculo de la distancia se lo realiza de la siguiente manera:

La función que permite obtener el pulso es *Pulsein*, la cual trabaja en un intervalo de 10 uS, para el cálculo del pulso se asume que la función trabaja a 4 MHz.

$$Para\ 10us\ y\ 4MHz \rightarrow pulso_{(4MHz\ y\ 10us)} = 40xT = \frac{40}{Fclock}$$

$$T = pulsein \times pulso_{(4MHz\ y\ 10us)} = pulseinx\ \frac{40}{Fclock}$$

$$D = vxt$$

$$T1 = T2$$

$$T = T1 + T2$$

$$t = \frac{T}{2}$$

$$D = v \times \frac{T}{2}$$

$$D = v \times \frac{40 \times pulsein}{2 \times Fclock}$$

$$D = \frac{20 \times v \times pulsein}{Fclock}$$

$$D = \frac{20 \times 34650 \times pulsein}{11{,}059200x10^{6}} \qquad D = \frac{\frac{cm}{s}}{\frac{1}{s}} = cm$$

$$D = \frac{Pulsein}{15{,}95}\ cm \rightarrow Calculo\ \ para\ obtener\ la\ lectura\ del\ sensor$$

Linealización de la respuesta del sensor.

Al obtener las lecturas del sensor con los datos calculados se obtuvo el número de pulsos generados por el sensor en las lecturas (Ver Tabla 3.2), a continuación se procedió a graficar la respuesta del sensor (Ver Figura 3.12), mediante la gráfica se obtuvo la ecuación de la linealización de la respuesta del sensor:

Tabla 3.2 Lecturas del sensor con los datos calculados

Pulsos	Distancia (cm)
31	5
64	10
96	15
121	20
153	25
182	30

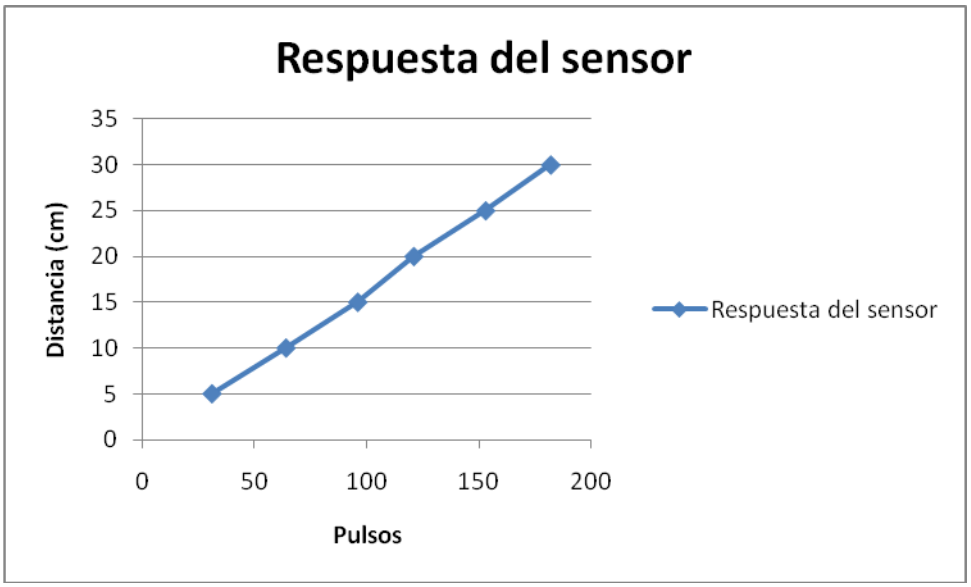


Figura 3.12 Gráfico de la respuesta del Sensor

Para la linealización es necesario obtener la respuesta lineal en el gráfico, en la Figura 3.13 se puede observar la ecuación de la linealización de la respuesta:

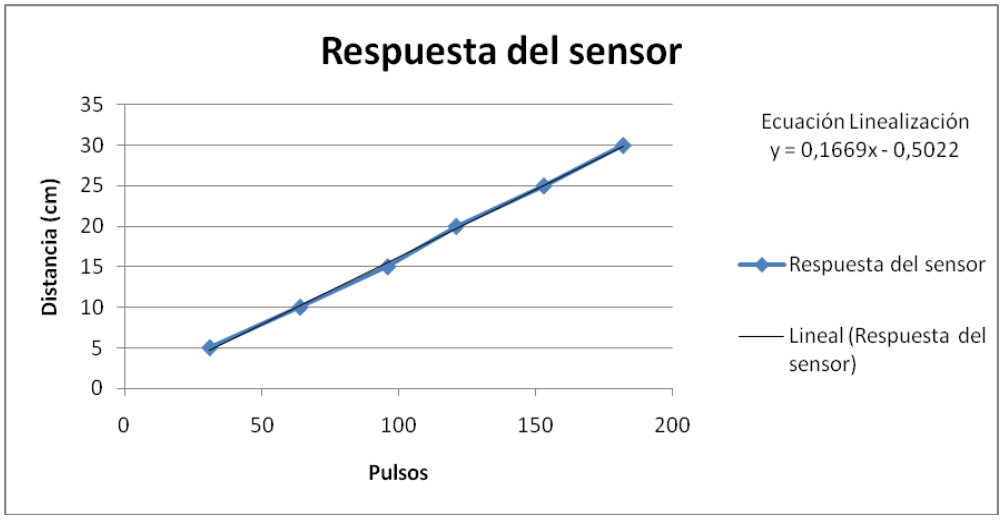


Figura 3.13 Calculo de la distancia proporcionada por el sensor

Mediante la ecuación obtenida en la Figura 3.13 se obtiene la respuesta linealizada del sensor, las respuestas del sensor se muestran en la Tabla 3.3:

Tabla 3.3 Datos de la respuesta linealizada del Sensor

Pulsos	Distancia (cm)	Linealización
31	5	4,6717
64	10	10,1794
96	15	15,5202
121	20	19,6927
153	25	25,0335
182	30	29,8736

A continuación se obtendrá el cálculo del error:

$$E_{\%} = \left| \frac{Vreal - Vmedido}{Vreal} \right| \times 100$$

$$E_{\%} = \left| \frac{5 - 4{,}67}{5} \right| \times 100 = 6{,}6\%$$

$$E_{\%} = \left| \frac{10 - 10{,}18}{10} \right| \times 100 = 1{,}8\%$$

$$E_{\%} = \left| \frac{15 - 15{,}52}{15} \right| \times 100 = 3{,}4\%$$

$$E_{\%} = \left| \frac{20 - 19{,}69}{20} \right| \times 100 = 1{,}5\%$$

$$E_{\%} = \left| \frac{25 - 25{,}03}{25} \right| \times 100 = 1{,}2\%$$

$$E_{\%} = \left| \frac{30 - 28{,}87}{30} \right| \times 100 = 3{,}7\%$$

Como se puede observar en el cálculo de los errores, estos son mínimos lo que no afectaría en las medidas de nivel, para obtener el volumen de la cantidad de leche.

Cálculo de volumen de leche del tanque almacenador con el sensor PING

Para el cálculo de la cantidad de leche del tanque de almacenamiento, se debe tomar en cuenta las distancias especificadas en la figura 3.14, mediante la obtención de las distancias y la forma del tanque se puede obtener el volumen de leche que contiene el tanque:

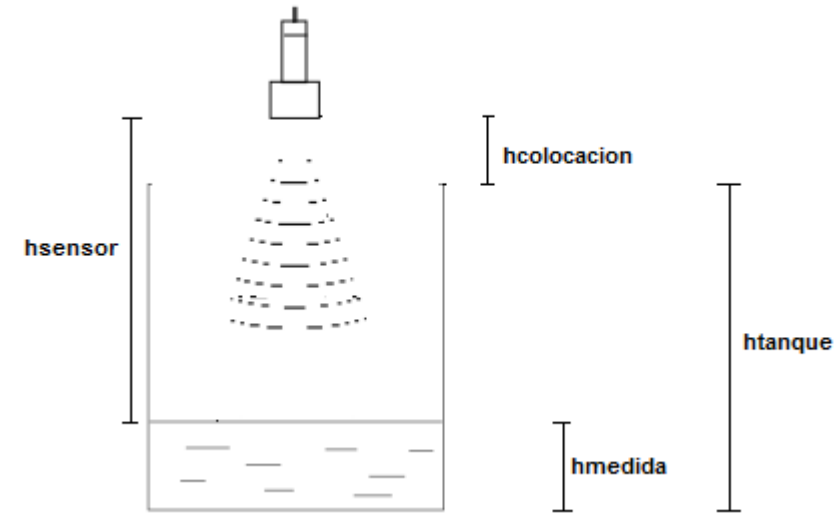


Figura 3.14 Distancias para el cálculo de volumen de leche que contiene el tanque

$$h_{medida} = (h_{\tan que} + h_{colocación}) - h_{sensor}$$

$$Volumen = Area_{\tan que} \times h_{medida}$$

$$Area_{\tan quecilindrico} = \pi \times r^2$$

$$Volumen_{\tan quecilindrico} = \pi \times r^2 \times h_{medida}$$

En este caso se asume que el tanque de almacenamiento es de forma cilíndrica por lo que se obtendrá el volumen a partir de la ecuación del volumen de un tanque cilíndrico.

3.2.3. Etapa de Transmisión

Esta etapa consiste en trasmitir los datos obtenidos de la cantidad de leche hacia un centro de gestión (Sistema Receptor) y recibir peticiones de estado por parte del centro de gestión; la transmisión y recepción de datos se la realiza a través de un Modem GSM, el cual envía y recibe los datos mediante el servicio de SMS.

La comunicación con el Modem se realiza a través de un puerto serial, y dependiendo del módem, se pueden usar los niveles definidos por la norma RS232 (Modems para PC) o niveles TTL (Modems para Circuito impreso).

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.

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

Los comandos AT con cadenas ASCII que comienzan por los caracteres AT y terminan con un retorno. Cada vez que el módem recibe un comando, lo procesa y devuelve un resultado, que normalmente es una cadena ASCII salvo que hayamos indicado lo contrario.

Para tener acceso a todos los servicios del modem GSM, 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+.

Para la implementación de la etapa de transmisión se utilizó el Modem ZTE MG3006 (Ver Figura 3.15), a continuación se indicará las principales características del Modem:

- **MODEM ZTE MG3006**

Figura 3.15 Modem ZTE MG3006

Características

A continuación se describirá las principales características del modem:

- Diseño industrial con capacidades de software inteligente, por lo que es fiable en soluciones celulares para la recolección de datos y transmisión.

- *Plug-and-play*, con la interfaz de software fácil de usar para una fácil integración.

- Incorpora Watch-dog.

- Posee Reloj en Tiempo Real (RTC).

- Control y monitoreo de datos remotamente.

- Fiable conectividad de red GSM, proporcionando un rápido y amplio rango de comunicación inalámbrica.

- Diseño industrial con protección contra sobrecarga.

- Configuración local y remota.

Aplicaciones

A continuación se describirá las principales aplicaciones del modem:

- Control y Monitoreo de datos remotamente.

- Medición de flujo de agua, petróleo y gases.

- AMR (lectura automática de contadores).

- Monitoreo y control de estaciones eléctricas.
- Terminales de punto de venta remotos.
- Monitoreo y control de señales de tránsito.
- Administración de flotas.
- Supervisión de distribución de redes de energía.
- Supervisión de sistemas centrales de calefacción.
- Transmisión de datos estaciones climáticas.
- Adquisición de datos hidrológicos.
- Maquinas de distribución.
- Guía de información de tráfico.
- Parquímetros.
- Supervisión de equipos de Telecomunicaciones (estación base móvil, microondas).
- Adquisición de datos en campos petroleros.
- Supervisión de seguridad de locales.

Especificaciones

Entre las principales especificaciones se encuentran las siguientes:

- Bandas de Frecuencias: En la Tabla 3.4 se puede observar las Bandas de Frecuencias a la que trabaja el Modem.

Tabla 3.4 Bandas de Frecuencias del Modem ZTE MG3006

Especificación	Modem	Banda de Frecuencias
Redes de Telefonía Móvil	M12H111	GPRS phase2/2+,EGSM 900MHz/DCS 1800MHz GSM850MHz/PCS 1900MHz
	M12Z111	Mobile Networks GSM 850MHz/EGSM 900MHz/DCS 1800MHz/PCS 1900MHz
	M12S211,	GPRS phase2/2+,GSM 900MHz/GPRS 1800MHz
	M12O111,	GSM/GPRS EGSM 900MHz/DCS 1800MHz/PCS 1900 MHz

- Transferencia de Datos: En la Tabla 3.5 se puede observar las Tazas de Transferencias de Datos del Modem.

Tabla 3.5 Transferencia de Datos del Modem ZTE MG3006

Especificación	Transferencia de Datos
GPRS	Downlink up to 85.6kbps Uplink up to 21.4kbps
GSM	GSM phase 2/2+
CSD	Downlink/Uplink up to 14.4kbps

- Interfaces: En la Tabla 3.6 se puede observar las Características de las Interfaces del Modem.

Tabla 3.6 Características de las interfaces del Modem ZTE MG3006

Especificación	Característica
Antena	50 dBi, conector SMA
Puerto Serial	DB9 (RS-232)
Led	Power Ring Data
UIM/SIM	1.8V/3V

- Energía: En la Tabla 3.7 se puede observar las Características del Consumo de Energía del Modem.

Tabla 3.7 Consumo de Energía del Modem ZTE MG3006

Especificación	Características
Fuente de energía	DC5V-25V , recomendado 9V a 1 A
Consumo de Energía	Peek: 2.5mA+9VDC Speech: 300mA+9VDC Sleep: 3.5mA+9VDC

- Físicas: En la Tabla 3.8 se puede observar las Características Físicas del Modem.

Tabla 3.8 Especificaciones físicas del Modem ZTE MG3006

Especificación	Característica
Temperatura	Temperatura de trabajo: -20 - 55 Temperatura de almacenamiento: -25 - 70
Humedad	95% Máximo (Sin condensación)
Dimensiones	Item (L x B x H): 75mm x 50mm x 16mm Empaquetado (L x B x H): 260mm x 190mm x 65mm
Peso	Item: 200g Empaquetado: 2.0 lbs

Panel

Figura 3.16 Descripción del Panel del Modem ZTE MG3006

La Figura 3.16 muestra el panel del Modem, este panel posee tres leds de estado, la tabla 3.9 describe el funcionamiento de los leds:

Tabla 3.9 Funcionamiento de los Leds del Modem ZTE MG3006

	Led Alimentación	**Led Ring**	**Led Datos**
Puesta en marcha	Encendido3s,intermitente 0.5s, parpadea 0.5s, encendido 0.5s	Parpadea	Encendido 0.5s
Inicio de sesión de red	Intermitente	Parpadea	Intermitente
Estado de no trabajo	Encendido 0.5s, parpadea 0.5s	Parpadea	Parpadea
Datos transferidos	Encendido 0.5s, parpadea 0.5s	Parpadea	Intermitente
Datos no transferidos	Encendido 0.5s, parpadea 0.5s, encendido 1s	Parpadea	Parpadea
Llamada de Voz	Encendido 0.5s, parpadea 0.5s	Encendido 1s	Parpadea
Reinicio	Después de 5s parpadea	Parpadea 4s	Parpadea

- **Formas de Conexión del Modem**

  - Conexión con Pc: La Figura 3.17 muestra la conexión del Modem a una Pc a través del Cable Serial RS232.

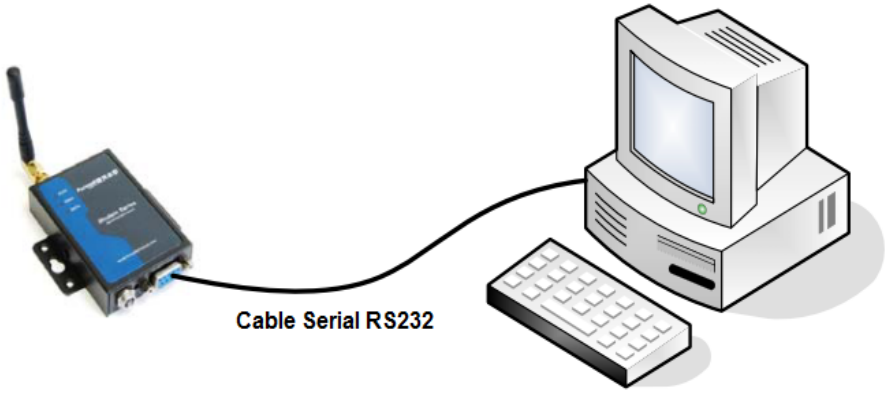


Figura 3.17 Conexión del Modem con una Pc

  - Conexión con Microcontrolador: La Figura 3.18 muestra la conexión del Modem a un Microcontrolador a través del Cable Serial RS232.

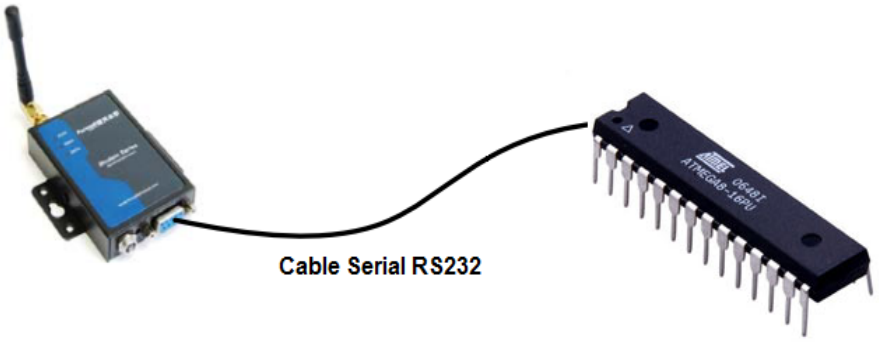


Figura 3.18 Conexión del Modem con un Microcontrolador

- **Comandos AT utilizados por el Modem**

  - ATE: La Tabla 3.10 muestra los principales parámetros del comando ATE.

Tabla 3.10 Comando ATE

Descripción	Este comando se utiliza para habilitar el eco.	
Formato	ATE<n>	
Ejemplo	ATE0 OK OK	
	ATE1 OK ATE1 OK	
Parámetros	<n>=0 Deshabilitado. <n>=1 Habilitado.	

  - AT+IPR: La Tabla 3.11 muestra los principales parámetros del comando AT+IPR.

Tabla 3.11 Comando AT+IPR

Descripción	Este comando se utiliza para establecer la velocidad de transmisión del módulo.	
Formato	AT+IPR=<velocidad de transmisión>	
Ejemplo	AT+IPR? +IPR: 115200 OK	Consulta al módulo la velocidad de transmisión actual
	AT+IPR=?	Consulta las velocidades de transmisión que son soportadas
	AT+IPR=115200 OK	Establece la velocidad de transmisión como 115200
Comentario	La velocidad de transmisión más alta (115200bps) sólo puede utilizarse en EDGE y la plataforma 3G. El uso de AT&W sirve para guardar el la velocidad de transmisión, caso contrario, volverá a reajustarse 115200bps si el módulo se apaga.	

- AT+CMGF: La Tabla 3.12 muestra los principales parámetros del comando AT+CMGF.

Tabla 3.12 Comando AT+CMGF

Descripción	Este comando se utiliza para establecer el modo de entrada de SMS.	
Formato	AT+CMGF=<n>	
Ejemplo	AT+CMGF=1 OK AT+CMGF? +CMGF:1 AT+CMGF=? +CMGF=(0-1) OK	Establece el modo de entrada de SMS como entrada de texto. Consulta modo de entrada de la configuración actual. Configuración actual como modo texto. Consulta el intervalo de la configuración actual.
Parámetros	<n>=0 Modo PDU. <n>=1 Modo Texto.	

- AT+CNMI: La Tabla 3.13 muestra los principales parámetros del comando AT+CNMI.

Tabla 3.13 Comando AT+CNMI

Descripción	Este comando se utiliza para configurar el formato de SMS.	
Formato	AT+CNMI=<mode>,<mt>,<bm>,<ds>,<bfr>	
Ejemplo	AT+CNMI=? +CNMI:(0-3),(0-3),(0,2,3),(0-1),(0) OK	Consulta el rango de los valores actuales
	AT+CNMI=3,1,0,0,0 OK +CMTI: "SM",19	Establece modo de recepción de SMS en memoria: +CMTI: memoria, posición
	AT+CNMI=3,2,0,0,0 OK AT+CMGF=1 OK +CMT: "+86130********","","07/02/14,10:29:04+32"	Establece el modo de SMS, recepción y eliminación. Establece la configuración actual como en modo texto Recibir SMS de texto a partir de 130
Resultados Retornados	+CMTI:<mem>,<index>: indica la recepción de un mensaje nuevo. +CMT:,<length><CR><LF><pdu>: salida directa de recepción de mensajes (Modo PDU) +CBM:<length><CR><LF><pdu>: salida directa en modo broadcast (Modo PDU)	

- AT+CSQ: La Tabla 3.14 muestra los principales parámetros del comando AT+CSQ.

Tabla 3.14 Comando AT+CSQ

Descripción	Este comando se utiliza para informar el indicador de la fuerza de la señal recibida (RSSI) y la tasa de error (BER)	
Formato	AT+CSQ	
Ejemplo	AT+CSQ +CSQ:<rssi>,<ber>	
Parámetros	<rssi>: 0–113dbm; 1-111dbm; 2..30–109..-53dbm; 31-51dbm; 99: red no disponible. <ber>: 0~7: normal; 99: red no disponible.	

- AT&W: La Tabla 3.15 muestra los principales parámetros del comando AT&W.

Tabla 3.15 Comando AT&W

Descripción	Este comando se utiliza para guardar la configuración actual.	
Formato	AT&W	
Ejemplo	AT&W	Guarda configuración

- AT+CMGS: La Tabla 3.16 muestra los principales parámetros del comando AT+CMGS.

Tabla 3.16 Comando AT+CMGS

<table>
<tr><td>Descripción</td><td colspan="2">Este comando se utiliza para originar el mensaje desde el módulo a la red.
Retorna con parámetros a la terminal después de que el mensaje se originó con éxito.</td></tr>
<tr><td>Formato</td><td colspan="2">Modo Texto: (AT+CMGF=1)
AT+CMGS=<de><CR>
<data><Ctrl-Z/ESC>

Modo PDU: (AT+CMGF=0)
AT+CMGS=<longitud><CR>
<pdu><Ctrl-Z/ESC></td></tr>
<tr><td rowspan="3">Ejemplo</td><td>AT+CMGF=1
OK</td><td>Establece como modo texto.</td></tr>
<tr><td>AT+CMGS="13316538879"<CR>
ABC<ctrl/Z>
OK

AT+CMGF=0
OK</td><td>Envia el texto de “ABC” a 13316538879

Establece como modo PDU.</td></tr>
<tr><td>AT+CMGS=17<CR>
0891683108705505f011000b81312
0882624f700f1ff0361f118<Ctrl-Z>
+CMGS:2
OK</td><td>Envia el texto de “ABC” a 13028862427</td></tr>
<tr><td>Parámetros</td><td colspan="2"><de>: número al cual se envía el mensaje en modo Texto.
<longitud>: carácter de la longitud del texto TPDU en modo PDU.
<data>: texto en modo Texto</td></tr>
</table>

3.2.4. Etapa de Control

Esta etapa consiste en controlar las etapas de tiempo, de sensamiento y de transmisión, a través de un microcontrolador, el cual se encargará de realizar las correspondientes funciones para que el sistema de recolección y envió de la cantidad de leche funcione correctamente.

Para la ejecución de esta etapa se analizó el microcontrolador que más se ajuste a las necesidades y requerimientos del sistema, por lo que se decidió elegir el microcontrolador Atmega8 de la familia AVR. Para crear el programa de control que maneje el microcontrolador es necesario utilizar un compilador donde se

diseñará todas las instrucciones que debe cumplir el sistema de transmisión, el compilador a utilizar es *BASCOM-AVR*, este compilador genera un archivo con extensión .hex que será cargado en el microcontrolador a través de un programador, en este caso el programador a utilizar será PROGISP 1.6.7.

- **Microcontrolador Atmega8**

El Atmega8 es un microcontrolador CMOS de bajo consumo de potencia basado en la arquitectura AVR RISC, permitiendo diseñar un sistema óptimo de consumo de potencia frente a la velocidad de procesamiento.

El microcontrolador ATMEGA8 es de 8 bits, su procesador presenta características avanzadas de tipo RISC, segmentado y arquitectura Harvard.

La tecnología RISC (*Reduced Instruction Set Computing*), o sea presentan instrucciones con complejidad reducida, a diferencia de otros que tienen tecnología CISC (*Complex Instrucción Set Computing*), permite una rápida ejecución de las instrucciones que se ejecutan en un solo ciclo de reloj, el ATMEGA 8 consigue obtener 1 MIPS por MHz, permitiendo al diseñador del sistema optimizar su consumo de energía versus la velocidad de procesamiento. Las instrucciones en la memoria de programas son ejecutados con estructura segmentada (*pipeling*), al mismo tiempo que una instrucción es ejecutado, se realiza la búsqueda de la próxima instrucción. Este concepto permite habilitar instrucciones paras ser ejecutados con cada ciclo de reloj.

Las características principales del microcontrolador ATMEGA8 son:

- Tiene 32 registros de 8 bits de propósito general. Todos estos registros están conectados a la unidad aritmética lógica (ALU) para un rápido acceso, una instrucción es ejecutada con uno solo ciclo de reloj.

- Tipos de Memoria:
  Memoria flash de 8 kbytes
  EEPROM de 512 bytes
  SRAM de 1 kbytes

- 2 temporizador/contador de 8 bits con pre-escalador y comparador.

- 1 temporizador/contador de 16 bits con pre-escalador, comparador y capturador.

- 8 canales de entrada para cada convertidor A/D (en TQFP y MLF).

- 6 canales A/D de 10 bits y 2 canales A/D de 8 bits.

- 6 canales de entrada para cada convertidor A/D (tipo PDIP).

- 4 canales A/D de 10 bits.

- 2 canales A/D de 8 bits.

- 1 USART (módulo programable para comunicación serial).

- 1 módulo SPI, para interface serial (*master/slave*).

- 1 *watch dog*.

- 1 comparador analógico.

- Puertos programables de entrada/salida.
  Puerto B, con 8 líneas
  Puerto C, con 7 líneas
  Puerto D, con 8 líneas

- Velocidad de operación:
  0-16 MHZ (ATMEGA8)

- Voltaje de alimentación:
  4.5 a 5.5 voltios (ATMEGA8)

- Tipo de empaque:
  PDIP de 28 pines
  TQFP de 32 pines
  MLF de 32 pines

- Otras características:
  Fuentes de interrupción internas y externas
  Oscilador interno de 1, 2, 4 y 8 MHz.

Se puede configurar su frecuencia de trabajo a través de su oscilador interno a 1, 2, 4 y 8 MHz, también el microcontrolador puede hacer uso de un cristal externo.

La memoria de programas y de datos están separadas (arquitectura *Harvard*). Para la memoria de programas, el microcontrolador maneja un bus de direcciones de 14 bits, por lo tanto puede direccionar hasta 4kbx16, dado que la memoria de programas (Flash de 8kbx8) está organizado en 4kbx16.

El Atmega8 además de poseer los 32 registros de propósito general, también dispone de 3 registros índices de 16 bits, X, Y, Z, un registro contador de programa PC y un puntero de pila, SP (*stack pointer*), también

de 16 bits. El registro de estado, contiene los 8 indicadores: C (bandera de acarreo), V (bandera de desbordamiento), Z (bandera de resultado cero), N (bandera negativo), H (bandera de acarreo a la mitad), I (habilitador de interrupciones), T (copia, almacena un bit), y S (bit de signo).

A continuación se explicará la descripción de los pines del microcontrolador ATMEGA8:

- VCC y GND. Son los pines de alimentación (+5 v) y tierra (0 v).

- XTAL1 y XTAL2. Conectores del cristal de reloj externo.

- RESET. Corresponde a la línea de *reset* (entrada).

- AVCC. Es el pin para conectar la fuente de alimentación al convertidor A/D.

- AREF. Para conectar una tensión de referencia para el conversor A/D interno.

- Puerto B (PB7…PB0). Compuesto de 8 bits, a cada pin le corresponde un bit, son bidireccionales con resistencia interna *pull up* para cada bit. Alternativamente, cada pin tiene otras funciones alternativas, por ejemplo PB6 y PB7, permiten la conexión a un cristal externo.

- Puerto C (PC0…PC5). Tiene 7 bits, bidireccionales con resistencias internas *pull up*, para cada bit.

- PC6/RESET. El bit 6, puede ser usado como entrada/salida si los fusibles han sido programados, en caso contrario, PC6 es usado para la entrada *Reset*. Un bajo nivel en este pin generará un *reset.*

- Puerto D (PD0…PD7). Son 8 líneas bidireccionales de entrada/salida con resistencias internas *pull up*, para cada uno. Los pines del puerto D, por ejemplo PD0 y PD1 permiten usar el periférico de comunicación serial USART, PD1 permite la transmisión de datos y PD0, permite la recepción de datos.

- **Compilador Bascom**

En las figuras 3.19 y 3.20 se puede observar el compilador Bascom en donde se realizo la programación del Sistema Transmisor:

Figura 3.19 Presentación del compilador BASCOM-AVR 1.11.9.5

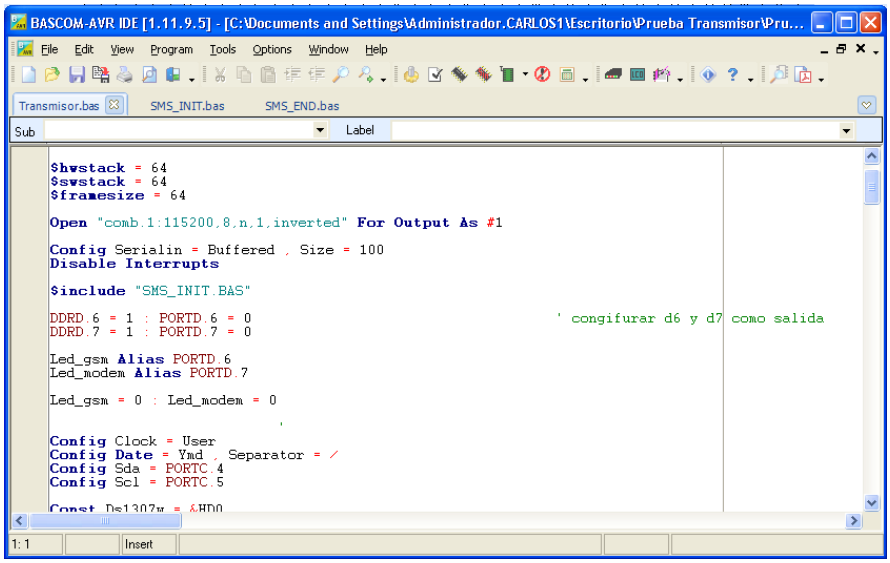


Figura 3.20 Programa BASCOM-AVR

- **Programador de Microcontroladores (PROGISP 1.6.7)**

En la Figura 3.21 se puede observar el Software 1.6.7, que es un programador que consta de dos partes, un *software* para el computador y un dispositivo físico (AVR USBASP) que se encargan de traspasar el archivo hexadecimal creado en un compilador, al microcontrolador que debe ejecutar el programa.

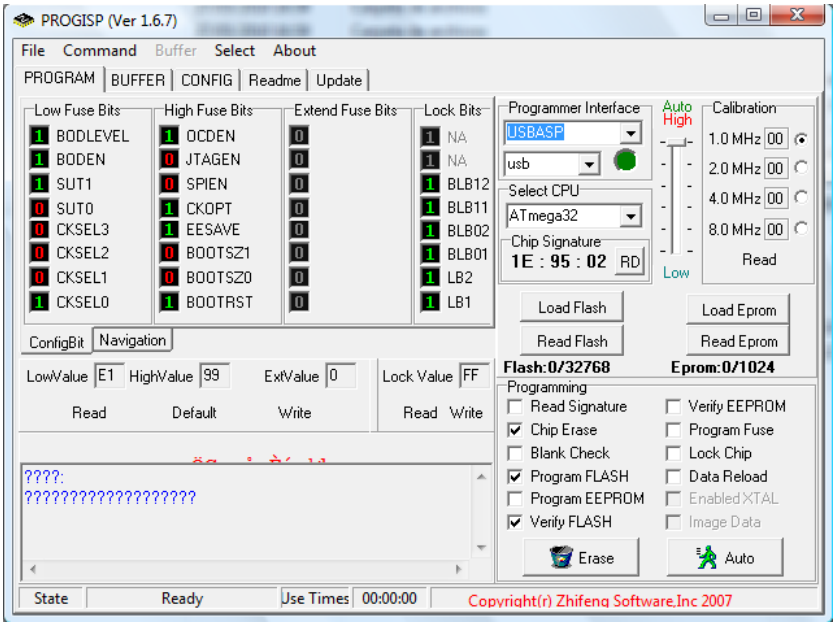


Figura 3.21 Software PROGISP 1.6.6

En este *software* se configura los fuses. Los *fuses* son bits que determinan la frecuencia a la que el microprocesador trabajara. Los *fuses bits*, están divididos en *fuses* alto y *fuses* bajo. A continuación una breve descripción de estos bits:

En el registro de *bits de fuse altos* se configuran los siguientes:

- ***OCDEN:*** Habilita algunos osciladores a pesar de estar en modo *sleep*.
- ***JTAGEN:*** Habilita el *JTAG*, interfaz que cumple con el estándar 1149.1 de la IEEE.
- ***SPIEN:*** Habilita o deshabilita el uso del ISP.
- ***CKOPT:*** Su funcionalidad depende de los bits de *CKSEL*.
- ***EESAVE:*** Indica si se borra la memoria Eeprom o no durante el ciclo de borrado.
- ***BOOTSZ1:*** Configura el tamaño del arrancador.
- ***BOOTSZ0:*** Cargador.
- ***BOOTRST:*** Selecciona donde comienza el vector del *reset*.

En el registro de *bits de fuses bajos* se configuran los siguientes:

- ***BODLEVEL:*** Indica el nivel en el que se detecta el nivel de bajo voltaje.
- ***BODEN:*** Habilita el detector de nivel de bajo voltaje.

- ***SUT1, SUT0:*** Indica el tiempo que debe de esperar antes iniciar el programa dentro del microcontrolador.

- ***CKSEL3, CKSEL2, CKSEL1, CKSEL0:*** Se utiliza para seleccionar los tipos de reloj a utilizar desde el oscilador interno de 1 MHz hasta 8 MHz internos, o los osciladores externos que alcanzan hasta los 16MHz.

3.3. Diagramas de Flujo del sistema

A continuación se describirá el funcionamiento de cada etapa en lo referente a su programación en el compilador BASCOM, mediante diagramas de flujo y codificación:

3.3.1. Etapa de Sensamiento

- **Diagrama de Flujo**

En la Figura A.1 del Anexo A2, se puede observar el Diagrama de Flujo del funcionamiento de la Etapa de Sensamiento.

- **Codificación**

Se configura el pin B0 como salida, el siguiente código:

```
Config PINB.0 = Output
DDRB.0 = 1 : PORTB.0 = 0
```

Se realiza un retardo 50 milisegundos y se genera un pulso para que trabaje el sensor:

```
Waitms 50
```

```
Pulseout PORTB, 0 , 55
```

Se realiza un retardo de 700 microsegundos y se configura el puerto B0 como entrada, se vuelve a realizar un retardo de 50 microsegundos; estas líneas de código sirven para esperar la respuesta del sensor:

```
Waitus 700
DDRB.0 = 0 : PORTB.0 = 0
Waitus 50
```

Se realiza la lectura de la respuesta del sensor mediante la función *pulsein*:

```
Pulsein Lectura_ultrasonido, PINB, 0, 1
```

La función *pulsein*, nos permite almacenar la respuesta del sensor en la variable Lectura_ultrasonido; el último parámetro de la función (1), indica la respuesta de la función; retorna cero si existe un dato de la lectura es válido o retorna 1 en caso de que exista un error.

Se realiza una condición mediante la función *if*; si la respuesta de la función *pulsein* es 1 (error) no es medida valida, caso contrario nos calcula el nivel de la cantidad de la leche en el tanque almacenador:

```
If Err = 1 Then
Else
  Dist = Range * 0.1669
  Dist = Dist - 0.5022
End If
```

3.3.2. Etapa de Tiempo

La etapa de tiempo está compuesta por la etapa principal y subrutinas de temporización (settime, setdate y getdatetime):

- **Diagramas de Flujo**

Etapa Principal

En la Figura A.2 del Anexo A2, se puede observar el Diagrama de Flujo del funcionamiento de la Etapa de Principal.

Subrutina Settime

En la Figura A.3 del Anexo A2, se puede observar el Diagrama de Flujo del funcionamiento de la Subrutina Settime.

Subrutina Setdate

En la Figura A.4 del Anexo A2, se puede observar el Diagrama de Flujo del funcionamiento de la Subrutina Setdate.

Subrutina Getdatetime

En la Figura A.5 del Anexo A2, se puede observar el Diagrama de Flujo del funcionamiento de la Subrutina Getdatetime.

- **Codificación**

Etapa Principal

Se configura los puertos para la comunicación I2C, para la línea de señal Sda se configura el puerto C4 y para la línea de señal Scl se configura el puerto C5:

```
Config Sda = PORTC.4
Config Scl = PORTC.5
```

Se configura las direcciones para escritura y lectura de los datos del reloj calendario DS1307:

```
Const Ds1307w = &HD0
Const Ds1307r = &HD1
```

Se configura el reloj para utilizar las variables Time$ y Date$; se utiliza el argumento *User* para utilizar un propio código de lectura y escritura del microcontrolador en combinación con el puerto de comunicación I2C del reloj calendario y establecer el formato de la fecha:

```
Config Clock = User
Config Date = Ymd , Separator = /
```

Cada vez que se necesite trabajar con el tiempo y fecha en tiempo real, el compilador Bascom utiliza subrutinas de temporización que serán descritas más adelante, las variables para mostrar el tiempo y la fecha son las siguientes:

```
Time$    -> Tiempo
Date$    ->  Fecha
```

Subrutina Settime

Se transforma las variables establecidas de segundos, minutos y horas a BCD, debido a que para la comunicación I2C es necesario que los datos estén en BCD:

```
_sec = Makebcd(_sec) : _min = Makebcd(_min) : _hour = Makebcd(_hour)
```

Se inicia la comunicación I2C mediante la función I2cstart y se envía la dirección de escritura desde el microcontrolador hacia el DS1307:

I2cstart
I2cwbyte Ds1307w
I2cwbyte 0

Se envía los datos de día, mes y año, mediante la función I2cwbyte, hacia el Ds1307:

I2cwbyte _sec
I2cwbyte _min
I2cwbyte _hour

Se detiene la comunicación I2C mediante la función I2cstop:

I2cstop

Subrutina Setdate

Se transforma las variables establecidas de día, mes y año a BCD, debido a que para la comunicación I2C es necesario que los datos estén en BCD:

_day = Makebcd(_day) : _month = Makebcd(_month) : _year = Makebcd(_year)

Se inicia la comunicación I2C y se envía la dirección de escritura desde el microcontrolador hacia el DS1307:

I2cstart
I2cwbyte Ds1307w
I2cwbyte 4

Se envía los datos de día, mes y año, hacia el Ds1307:

I2cwbyte _day
I2cwbyte _month
I2cwbyte _year

Se detiene la comunicación I2C:

I2cstop

Subrutina Getdatetime

Se inicia la comunicación I2C y se envía la dirección de escritura al DS1307:

I2cstart
I2cwbyte Ds1307w
I2cwbyte 0

Se inicia la comunicación I2C y se envía la dirección de lectura al Ds1307:

I2cstart
I2cwbyte Ds1307r

Leer los datos de tiempo y fecha, indicando siempre si existe más datos para leer (Ack) y el último dato de lectura (Nack); se detiene la comunicación I2C:

I2crbyte _sec , Ack
I2crbyte _min , Ack
I2crbyte _hour , Ack
I2crbyte Weekday , Ack
I2crbyte _day , Ack
I2crbyte _month , Ack
I2crbyte _year , Nack
I2cstop

Transformar los datos leídos a decimales mediante la función Makedec() para poder trabajar con ellos:

_sec = Makedec(_sec) : _min = Makedec(_min) : _hour = Makedec(_hour)
_day = Makedec(_day) : _month = Makedec(_month) : _year = Makedec(_year)

3.3.3. Etapa de Transmisión

La etapa de transmisión está compuesta por 2 archivos:

- Archivo donde se declara las variables y funciones para el control del Modem.
- Archivo donde se desarrolla las funciones para el control del Modem.

Al momento de desarrollar la etapa de control es necesario cargar los dos archivos antes mencionado para que se ejecuten.

A continuación se describirá las funciones de control del modem GSM:

- **Diagramas de Flujo**

Función Configuración Inicial del Modem GSM

En la Figura A.6 del Anexo A2, se puede observar el Diagrama de Flujo del funcionamiento de la Función Configuración Inicial del Modem GSM.

Función de Envio de Mensajes del Modem GSM

En la Figura A.7 del Anexo A2, se puede observar el Diagrama de Flujo del funcionamiento de la Función de Envio de Mensajes del Modem GSM.

Función de espera de respuesta OK del Modem GSM

En la Figura A.8 del Anexo A2, se puede observar el Diagrama de Flujo del funcionamiento de la Función de espera de respuesta OK del Modem GSM.

Función Limpiar Buffer del puerto de comunicaciones del Modem GSM

En la Figura A.9 del Anexo A2, se puede observar el Diagrama de Flujo del funcionamiento de la Función Limpiar Buffer del puerto de comunicaciones del Modem GSM.

Función Recibir Mensaje del Modem GSM

En la Figura A.10 del Anexo A2, se puede observar el Diagrama de Flujo del funcionamiento de la Función Recibir Mensaje del Modem GSM.

Función Validar Mensaje del Modem GSM

En la Figura A.11 del Anexo A2, se puede observar el Diagrama de Flujo del funcionamiento de la Función Validar Mensaje del Modem GSM.

Función Automensaje

En la Figura A.12 del Anexo A2 se puede observar el Diagrama de Flujo del funcionamiento de la Función Automensaje del Modem GSM.

- **Codificación**

Función Configuración Inicial del Modem GSM

Esta función permite configurar los principales parámetros del Modem para su funcionamiento, a continuación se explicará la codificación:

Se limpia el buffer de comunicaciones y se limpia la variable que contiene la respuesta del buffer de comunicaciones:

```
Clear Serialin
Modem_answer = ""
```

Se verifica si existe la tarjeta SIM y la correcta conexión a la red GSM del modem; para la verificación de estos dos parámetros la codificación es la misma por lo que se explicará el código una sola vez:

Mediante la función Iscarwaiting() se verifica si existe datos en el buffer de comunicaciones, si existe datos en el buffer se utiliza la función Inkey() para capturar el valor decimal del primer caracter ASCII que se encuentra en el buffer; se utiliza la función select-case para verificar si la función Inkey() devuelve un dato válido; si el dato es válido, esta verificado que los parámetros se encuentran correctamente, caso contrario se vuelve a utilizar la función Inkey() hasta encontrar un dato válido. Si los parámetros están correctamente verificados se limpia el buffer de comunicaciones y la variable que contiene la respuesta del buffer:

```
Do
   If Iscarwaiting() = 1 Then
      Set Led_gsm
      Do
        B = Inkey()
        Select Case B
          Case 10 : If Modem_answer <> "" Then Exit Do
          Case 13 :
          Case Else:
             Modem_answer = Modem_answer + Chr(b)
        End Select
      Loop
      Exit Do
   End If
Loop
Clear Serialin
Modem_answer = ""
```

Se llama a la función Limpiarbuffer para verificar que el buffer de comunicaciones se encuentra vacio:

```
Limpiarbuffer
For I = 0 To 25
    Toggle Led_modem
    Waitms 25
Next
```

Se envía el comando AT por el puerto de comunicaciones serial seguido del caracter decimal 13 que representa salto de línea; se envía este comando hasta que el modem responda OK, para obtener la respuesta OK se llama a la función Getok(); el comando AT permite iniciar la configuración del Modem:

```
Do
    Print "AT" ; Chr(13);
    Getok Modem_answer
Loop Until Modem_answer = "OK"
```

Se llama a la función Limpiarbuffer para verificar que el buffer de comunicaciones se encuentra vacio:

```
Limpiarbuffer
For I = 0 To 5
    Toggle Led_modem
    Waitms 25
Next
```

Se envía el comando ATEO por el puerto de comunicaciones serial seguido del caracter decimal 13 que representa salto de línea; se envía este comando hasta que el modem responda OK, para obtener la respuesta OK se llama a la función Getok(); el comando ATEO sirve para eliminar el eco producido por el Modem:

```
Do
    Print "ATE0" ; Chr(13);
    Getok Modem_answer
Loop Until Modem_answer = "OK"
```

Se llama a la función Limpiarbuffer para verificar que el buffer de comunicaciones se encuentra vacio:

```
Limpiarbuffer
For I = 0 To 5
    Toggle Led_modem
    Waitms 25
Next
```

Se envía el comando AT+IPR por el puerto de comunicaciones serial seguido del caracter decimal 13 que representa salto de línea; se envía este comando hasta que el modem responda OK, para obtener la respuesta OK se llama a la función Getok(); el comando AT+IPR sirve para establecer la velocidad de transmisión del Modem:

```
Do
    Print "AT+IPR=115200" ; Chr(13);
    Getok Modem_answer
Loop Until Modem_answer = "OK"
```

Se llama a la función Limpiarbuffer para verificar que el buffer de comunicaciones se encuentra vacio:

```
Limpiarbuffer
For I = 0 To 5
    Toggle Led_modem
    Waitms 25
Next
```

Se envía el comando AT+CMGF por el puerto de comunicaciones serial seguido del caracter decimal 13 que representa salto de línea; se envía este comando hasta que el modem responda OK, para obtener la respuesta OK se llama a la función Getok(); el comando AT+CMGF sirve para establecer el modo de SMS como texto:

```
Do
    Print "AT+CMGF=1" ; Chr(13);
    Getok Modem_answer
Loop Until Modem_answer = "OK"
```

Se llama a la función Limpiarbuffer para verificar que el buffer de comunicaciones se encuentra vacio:

```
Limpiarbuffer
For I = 0 To 5
    Toggle Led_modem
    Waitms 25
Next
```

Se envía el comando AT+CNMI por el puerto de comunicaciones serial seguido del caracter decimal 13 que representa salto de línea; se envía este comando hasta que el modem responda OK, para obtener la respuesta OK se llama a la función Getok(); el comando AT+CNMI sirve para establecer el formato de SMS para ser enviados por el Modem:

```
Do
    Print "AT+CNMI=3,2,0,0,0" ; Chr(13);
    Getok Modem_answer
Loop Until Modem_answer = "OK"
```

Se llama a la función Limpiarbuffer para verificar que el buffer de comunicaciones se encuentra vacio:

```
Limpiarbuffer
For I = 0 To 5
    Toggle Led_modem
    Waitms 25
Next
```

Se envía el comando AT+CSQ por el puerto de comunicaciones serial seguido del caracter decimal 13 que representa salto de línea; se envía este comando hasta que la respuesta del Modem sea un valor adecuado de señal, para obtener la respuesta se llama a la función Getok(); el comando AT+CSQ sirve para verificar si el Modem tiene una señal adecuada para el correcto envío de SMS:

```
Do
    Modem_csq_rssi = ""
    Modem_csq_ber = ""
    Do
      Print "AT+CSQ" ; Chr(13);
      Getok Modem_answer
    Loop Until Modem_answer <> ""
    For I = 0 To 5
      Toggle Led_modem
      Waitms 20
    Next
    Modem_csq_rssi = Mid(modem_answer , 7 , 2)
    Modem_csq_ber = Mid(modem_answer , 10 , 2)
    Waitms 100
Loop Until Modem_csq_rssi <> "" And Modem_csq_ber <> "" And
Modem_csq_rssi <> "99" And Modem_csq_ber <> "99"
```

Se limpia el buffer del puerto de comunicaciones:

```
Clear Serialin
```

Se envía el comando AT+W por el puerto de comunicaciones serial seguido del caracter decimal 13 que representa salto de línea; se envía este comando hasta que el modem responda OK, para obtener la respuesta OK se llama a la función Getok(); el comando AT+W sirve para guardar la configuración actual del Modem:

```
Do
    Print "AT&W" ; Chr(13);
    Getok Modem_answer
Loop Until Modem_answer = "OK"
```

Se llama a la función Limpiarbuffer para verificar que el buffer de comunicaciones se encuentra vacio:

```
Limpiarbuffer
For I = 0 To 5
    Toggle Led_modem
    Waitms 25
Next

Limpiarbuffer
Set Led_modem
Reset Led_gsm
Clear Serialin
```

Función de Envió de Mensajes del Modem GSM

Esta función permite enviar SMS a través del Modem, a continuación se explicará la codificación:

Se limpia el buffer de comunicaciones, y se realiza un retardo de 200 ms:

```
Clear Serialin
Waitms 200
```

Se envía el comando AT+CMGS=, seguido de la variable N (entre comillas), la variable N almacena el número de celular al que se desea enviar el mensaje; se envía este comando hasta obtener como respuesta en el buffer el caracter “>”. Si la respuesta en el buffer es el caracter “>”, se llama a la función limpiar buffer, después se envía el texto del mensaje. Se llama a la función getok() hasta obtener la respuesta OK del modem. Finalmente se limpia el buffer de comunicaciones:

```
Do
    Print "AT+CMGS=" ; Chr(34) ; N ; Chr(34) ; Chr(13);
    Sret = ""
    Do
      If Iscarwaiting() = 1 Then
        B = Inkey()
        Select Case B
          Case 13 :
          Case 10 :
          Case Else
            Sret = Sret + Chr(b)
        End Select
      End If
    Loop Until Sret = "> "
Loop Until Sret = "> "
Limpiarbuffer
Print S ; Chr(26);
Do
    Getok Sret
Loop Until Sret = "OK"
Clear Serialin
Mensa = ""
```

Función de espera de respuesta OK del Modem GSM

Esta función permite obtener la respuesta OK del Modem, a continuación se explicará la codificación:

Mediante la función Iscarwaiting() se verifica si existe datos en el buffer de comunicaciones, si existe datos en el buffer se utiliza la función Inkey() para capturar el valor decimal del primer caracter ASCII que se encuentra en el buffer; se utiliza la función select-case para verificar si la función Inkey() devuelve el dato OK; si el dato es válido, esta verificado que

los parámetros se encuentran correctamente, caso contrario se vuelve a utilizar la función Inkey() hasta encontrar el dato OK, la función getok() devuelve una variable con el dato OK:

```
S = ""
Do
   If Iscarwaiting() = 1 Then
     B = Inkey()
     Select Case B
       Case 10 : If S <> "" Then Exit Do
       Case 13 :
       Case Else:
          S = S + Chr(b)
     End Select
   End If
Loop
```

Función Limpiar Buffer del puerto de comunicaciones del Modem GSM

Esta función permite asegurar que el buffer de comunicaciones este vació para ser utilizado, a continuación se explicará la codificación:

Mediante la función Inkey() se obtiene el valor decimal del caracter ASCII que se encuentra en el buffer de comunicaciones, se utiliza esta función hasta obtener el valor 0:

```
Do
   B = Inkey()
Loop Until B = 0
```

Función Recibir Mensaje del Modem GSM

Esta función permite recibir los SMS que llegan al Modem, a continuación se explicará la codificación:

Para ejecutar la función se debe conocer el formato con el que llega el SMS, en la figura 3.22 se muestra el formato del SMS:

```
+CMT: "+59392754351",,"10/07/15,21:27:16-20"
ESTADO EQUIPO
```

Figura 3.22 Formato de SMS para Sistema Transmisor

Se crea una variable de tipo string que almacenará los datos del buffer. Mediante la función Inkey() obtenemos los valores ASCII de los caracteres en el buffer. Mediante la función select-case se almacena en la variable creada los caracteres que se encuentran en el buffer. Se almacena los caracteres hasta que la función Inkey() devuelva el valor ASCII 10(representa salto de línea). Se almacena los datos de la variable creada en la variable Datos_sms_recibido.

```
Do
  B = Inkey()
  Select Case B
    Case 10 : If Modem_answer <> "" Then Exit Do
    Case 13:
    Case Else
      Modem_answer = Modem_answer + Chr(b)
  End Select
Loop
Datos_sms_recibido = Modem_answer
```

La función Mid() permite separar una cadena de caracteres, en subcadenas. Mediante esta función se obtiene el número, fecha y hora del SMS recibido.

```
Numero_sms_recibido = Mid(datos_sms_recibido , 8 , 12)
Fecha_sms_recibido = Mid(datos_sms_recibido , 24 , 8)
Hora_sms_recibido = Mid(datos_sms_recibido , 33 , 8)
```

Se limpia la variable para almacenar los caracteres del buffer de comunicaciones, Mediante la función Inkey() obtenemos los valores ASCII de los caracteres en el buffer. Mediante la función select-case se almacena en la variable creada los caracteres que se encuentran en el buffer. Se almacena los caracteres hasta que la función Inkey() devuelva el valor ASCII 10. Se almacena los datos de la variable de caracteres en la variable de tipo string mensaje. Finalmente se limpia el buffer.

```
Modem_answer = ""
Do
  B = Inkey()
  Select Case B
    Case 10 : If Modem_answer <> "" Then Exit Do
    Case 13:
    Case Else
      Modem_answer = Modem_answer + Chr(b)
  End Select
Loop

Mensaje = Modem_answer
Clear Serialin
```

Función Validar Mensaje del Modem GSM

Esta función permite validar los SMS, dependiendo del número del que son enviados y el texto que poseen, a continuación se explicará la codificación:

Se limpia el buffer y se crea la variable que almacenará el texto del SMS a enviar.

```
Clear Serialin
Mensa = " "
```

Si el número del SMS recibido es igual a uno de los dos números validos, se utiliza la función select-case para validar el SMS dependiendo del texto que tenga. Si el texto es “ESTADO EQUIPO” o “Estado Equipo”, se procede a actualizar las variables Time$ y Date$ con la fecha y hora del SMS recibido, también se envía un SMS indicando el estado en que se encuentra el sensor “ACTIVO” o “PASIVO”. Si el texto es “AUTOMENSAJE”, se actualiza las variables Time$ y Date$ con la fecha y hora del SMS recibido.

```
If Numero_sms_recibido = Numero Or Numero_sms_recibido = Numero_auto Then
   Select Case S

      Case "ESTADO EQUIPO":
        Time$ = Hora_sms_recibido
        Date$ = Fecha_sms_recibido
        If Flag_estado = 1 Then
          Mensa = "ACTIVO"
        Else
          Mensa = "PASIVO"
        End If
        Enviarmensaje Mensa , Numero
```

```
Case "Estado equipo":

  Time$ = Hora_sms_recibido
  Date$ = Fecha_sms_recibido

  If Flag_estado = 1 Then
    Mensa = "ACTIVO"
  Else
    Mensa = "PASIVO"
  End If
  Enviarmensaje Mensa , Numero

Case "AUTOMENSAJE
   Time$ = Hora_sms_recibido
   Date$ = Fecha_sms_recibido
End Select
```

Caso contrario si el número no es igual a los dos números validos se almacena en una variable de tipo string el número del SMS y el mensaje del SMS, para ser enviado a un número de respaldo.

```
Else
  Mensa = "NUMERO DESCONOCIDO - MENSAJE: " + S + "
  NUMERO: " + Numero_sms_recibido
   Enviarmensaje Mensa , Numero_dos
End If
```

Función Automensaje

Esta función permite enviar un automensaje, para que de esta forma se actualice las variables Date$ y Time$, a continuación se explicará la codificación:

Se envía un SMS con el texto “AUTOMENSAJE” al mismo número del equipo transmisor, se limpia el buffer de comunicaciones.

```
Mensa = "AUTOMENSAJE"
Enviarmensaje Mensa , Numero_auto
Clear Serialin
```

Se espera hasta verificar mediante la función Iscarwaiting() que existe datos en el buffer, se llama a la función recibirmensaje y finalmente a la función validarmensaje.

```
Do
  Waitms 500
Loop Until Iscarwaiting() = 1

Recibirmensaje
Validarmensaje , Mensaje
```

3.3.4. Etapa de Control

- **Diagrama de Flujo**

Etapa de Control

En la Figura A.13 del Anexo A2, se puede observar el Diagrama de Flujo del funcionamiento de la Etapa de Control del Modem GSM.

Bucle Principal de la Etapa de Control

En la Figura A.14 del Anexo A2, se puede observar el Diagrama de Flujo del funcionamiento del Bucle Principal de la Etapa de Control del Modem GSM.

3.4. Diagrama Circuital del Sistema Transmisor

La Figura 3.24 muestra del diagrama circuital del Sistema Transmisor, para su realización se utilizo el Software Proteus 7.5, específicamente el programa ISIS mostrado en la Figura 3.23, en donde se puede encontrar todo tipo de componentes electrónicos para la realización de diagramas electrónicos y simulaciones.

Figura 3.23 Programa ISIS (Proteus 7.5)

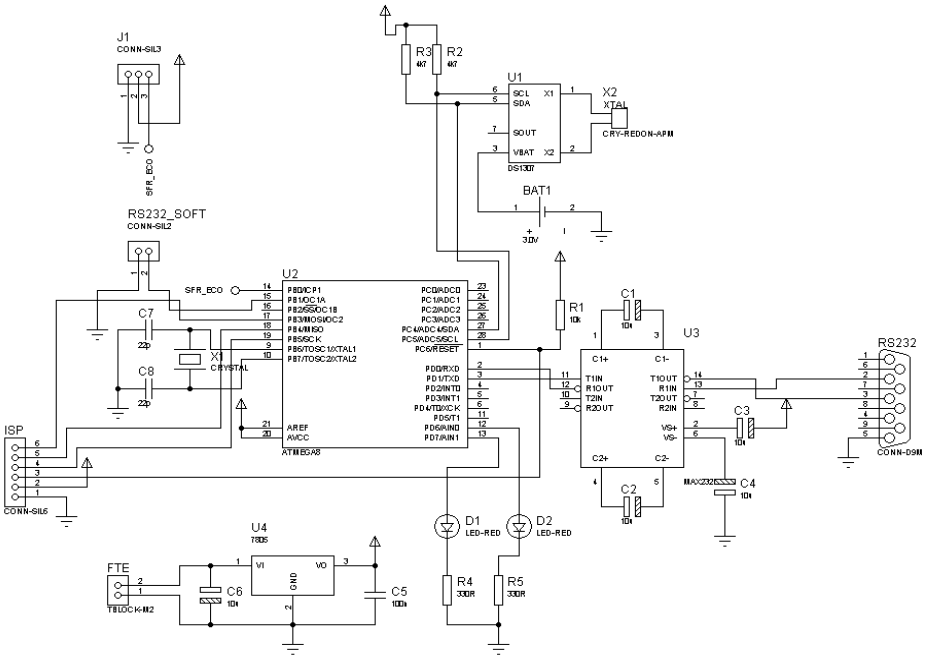


Figura 3.24 Diagrama Circuital del Sistema de Recolección y envío de cantidad de Leche

3.5. Implementación Placa Impresa del Sistema Transmisor

Para la realización de la placa impresa se utilizó el Software Proteus 7.5 mostrado en la Figura 3.25, específicamente el programa ARES, el cual permite realizar el ruteo de circuitos electrónicos mostrados en las Figuras 3.26, 3.27, 3,28. El Software Proteus es de gran ayuda ya que permite exportar todos los componentes electrónicos de un circuito determinado, desde el programa ISIS hacia el programa ARES, de esta forma permite tener grandes ventajas al momento de realizar el ruteo de circuitos. El programa Ares también permite realizar un diagrama electrónico virtual mostrado en la Figura 3,29, donde se puede observar la ubicación y diseño de los elementos electrónicos.

Figura 3.25 Programa ARES (Proteus 7.5)

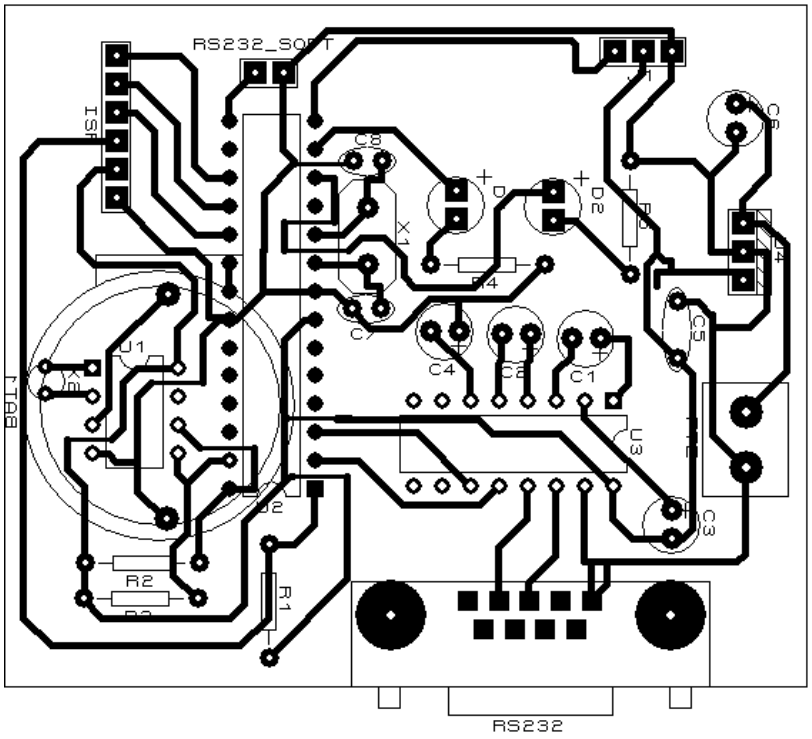


Figura 3.26 Diagrama de ruteo y elementos electrónicos

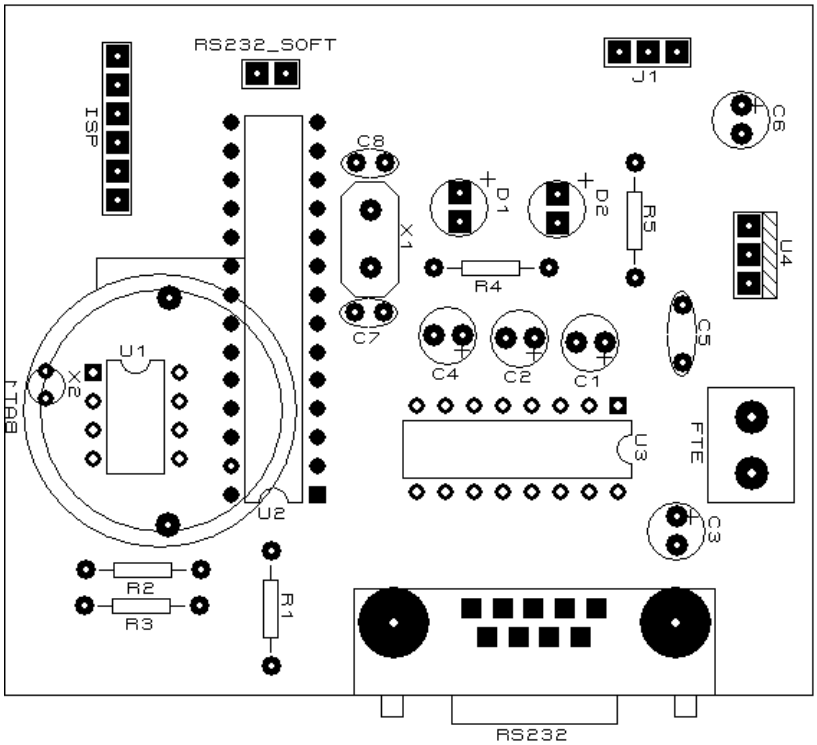


Figura 3.27 Diagrama de elementos electrónicos

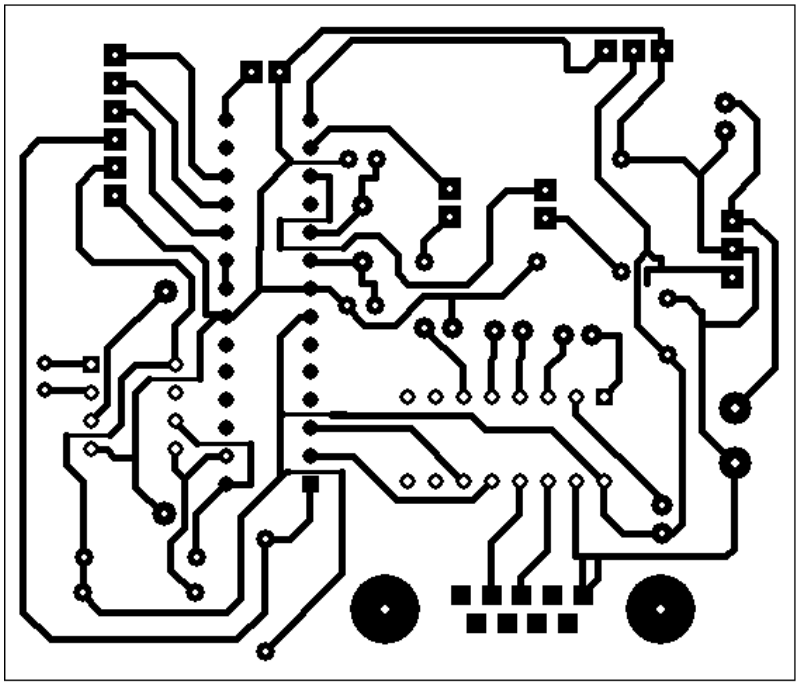


Figura 3.28 Diagrama de ruteo de la placa

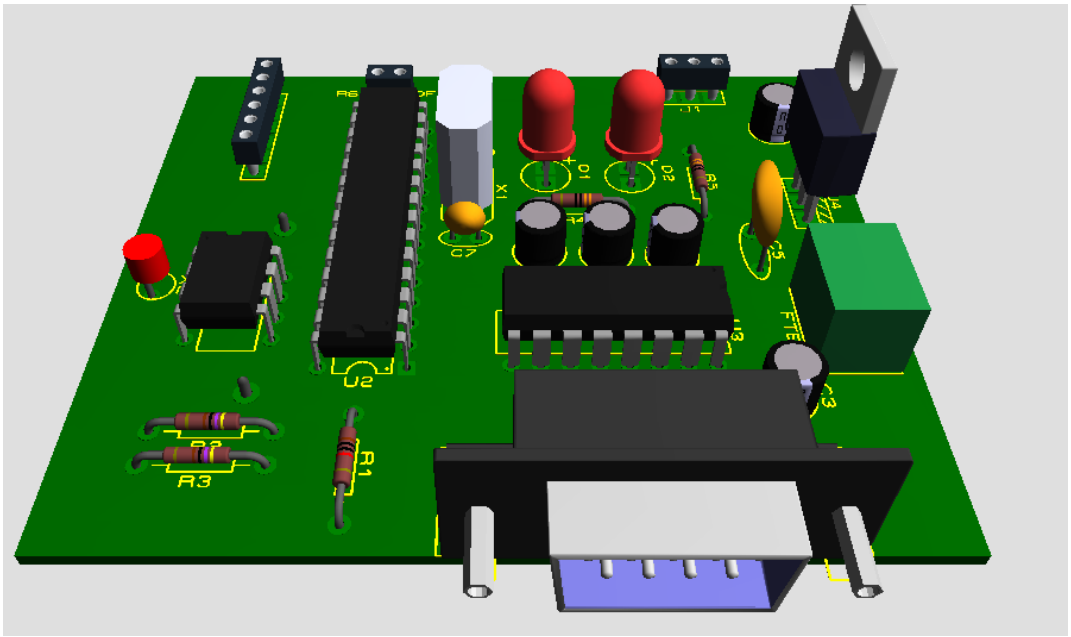


Figura 3.29 Diagrama virtual de elementos en placa impresa

3.6. Costo de los Elementos Electrónicos utilizados en el Sistema Transmisor

La Tabla 3.17 muestra los Costos de los Elementos Electrónicos utilizados en el Sistema Transmisor:

Tabla 3.17 Costos de la implementación del sistema de transmisión

Cantidad	Elementos Electrónicos	Precio unitario (USD)	Precio Total (USD)
1	MODEM	130	130
1	SENSOR ULTRASONIDO	54,76	54,76
1	ATMEGA8	3,75	3,75
1	MAX232	2,5	2,5
1	DS1307	3,5	3,5
1	CRY-32	0,6	0,6
1	ZOC PILA	0,95	0,95
1	PILA CR-2032	1	1
1	CRY-110592	0,5	0,5
2	C22P	0,08	0,16
5	C10U	0,1	0,5
1	ZOC28P	0,15	0,15
1	ZOC16P	0,12	0,12
1	BORN-2P	0,25	0,25
1	DB9-MACHO PLACA	0,65	0,65
1	ZOC8P	0,1	0,1
1	LM7805	0,5	0,5
1	LED-VE	0,08	0,08
1	LED-RO	0,08	0,08
2	R330	0,02	0,04
1	R10K	0,02	0,02
2	R4K7	0,02	0,04
1	REGLETA HEMBRA	0,5	0,5
1	GRABADOR ATMEL	20	20
		TOTAL	220,75

CAPITULO 4

4. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE RECEPCIÓN DE LA CANTIDAD DE LECHE (SISTEMA RECEPTOR)

4.1. Introducción

El sistema de recepción consiste en una aplicación que recibe los mensajes SMS a través del Modem, este SMS contiene los datos de la recolección de la leche del sistema transmisor, que serán decodificados por la aplicación. Para la decodificación del mensaje se utiliza el software SharpDevelop. Una vez decodificado el mensaje el software SharpDevelop almacenará los datos en una base de datos, esta base de datos es previamente creada mediante el software Microsoft SQL Server 2000. Los datos almacenados podrán ser consultados y presentados en un reporte, para obtener este reporte se ha utilizado los software Crystal Report XI y Visual .Net 2005.

4.2. Diagrama de Bloques del sistema receptor

La Figura 4.1 muestra el diagrama de bloques del Sistema Receptor:

Figura 4.1 Diagrama de Bloques del sistema receptor

4.2.1. Etapa de Recepción

La etapa de recepción es realizada por el Modem GSM, mediante comandos AT se programa al Modem para que almacene todos los SMS recibidos en su memoria interna.

4.2.2. Etapa Decodificadora

La etapa decodificadora es realizada por la aplicación creada en el Software Sharp Develop, esta aplicación revisa la memoria del Modem y lee los SMS. Después de leer los SMS, los decodifica para obtener los datos de fecha, hora y cantidad de leche. Una vez decodificados los mensajes los almacena en una base de datos.

4.2.3. Etapa de Almacenamiento

La etapa de almacenamiento es realizada por la aplicación creada en Sharp Develop y el Software Microsoft SQL Server 2000 (Sistema para la gestión de base de datos). La aplicación manipula la información de la base de datos a través de lenguaje SQL.

4.2.4. Etapa de Consulta

La etapa de consulta es realizada por la aplicación en Sharp Develop, esta aplicación nos permite escoger las fechas y horas específicas para la consulta de los datos de la cantidad de leche.

4.2.5. Etapa de Reporte

La etapa de reporte es realizada por una aplicación creada por el software Visual .Net 2005 y Crystal Repot XI, esta aplicación consulta los datos específicos

en la base de datos y los muestra en un informe (Generado y diseñado en Crystal Report).

4.3. Diseño del Sistema Receptor de Datos

El diseño del Sistema Receptor consiste básicamente en tres partes:

- Creación de la Base de Datos (Microsoft SQL server 2000).
- Manejo de Modem y Base de Datos (Sharp Develop).
- Creación del Reporte (Visual.Net 2005, Crystal Report XI y Microsoft SQL Server 2000).

4.3.1. Creación de la Base de Datos (Microsoft SQL server 2000)

Para la creación de la base de datos debemos realizar los siguientes pasos:

- Instalar el Software Microsoft SQL Server 2000, haciendo que el computador actúe como un servidor local.

- La Figura 4.2 muestra como abrir el Administrador Corporativo de Microsoft SQL Server.

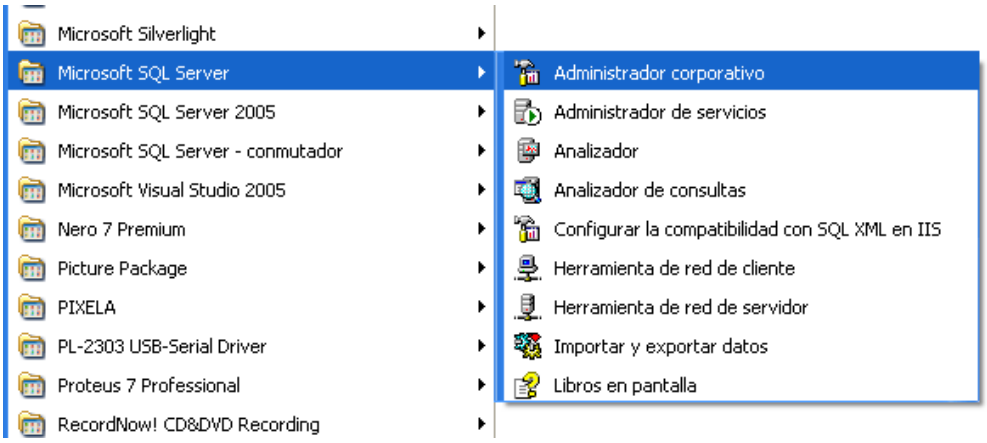


Figura 4.2 Administrador Corporativo

- La Figura 4.3 muestra como abrir el Servidor Local (en este caso CARLOS1), que contiene las Bases de Datos.

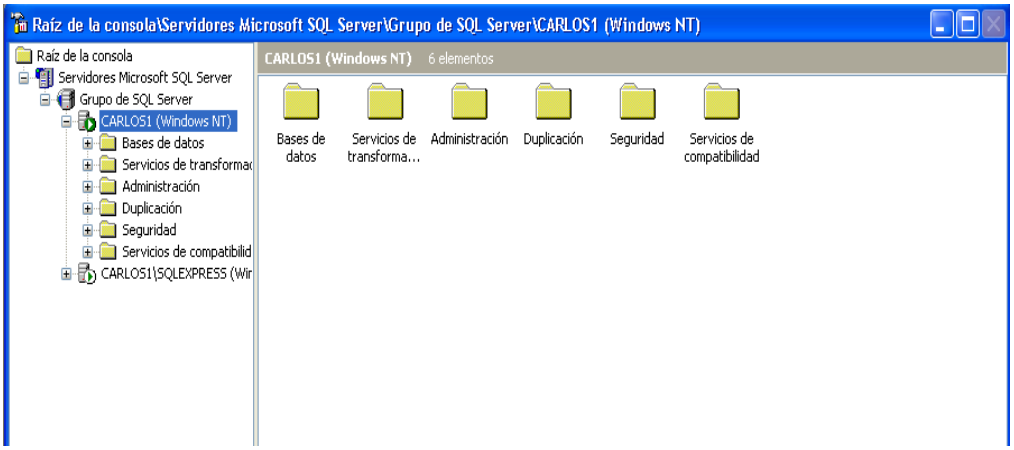


Figura 4.3 Servidor Local

- Dar click derecho en Bases de Datos, aparecerá un menú en donde se escoge la opción Nueva Base de Datos. Después aparecerá la ventana de la Figura 4.4 en donde se ingresa el nombre de la Base de Datos a crear.

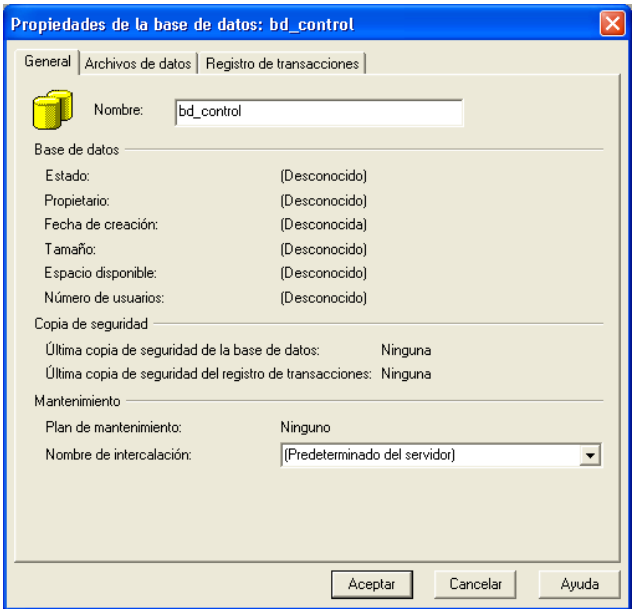


Figura 4.4 Creación Nueva Base de Datos

- Una vez creada la nueva Base de Datos, se escoge la Base de Datos y se da click derecho en el elemento tablas. Aparece el menú de la Figura 4.5 en donde se escoge Nueva Tabla.

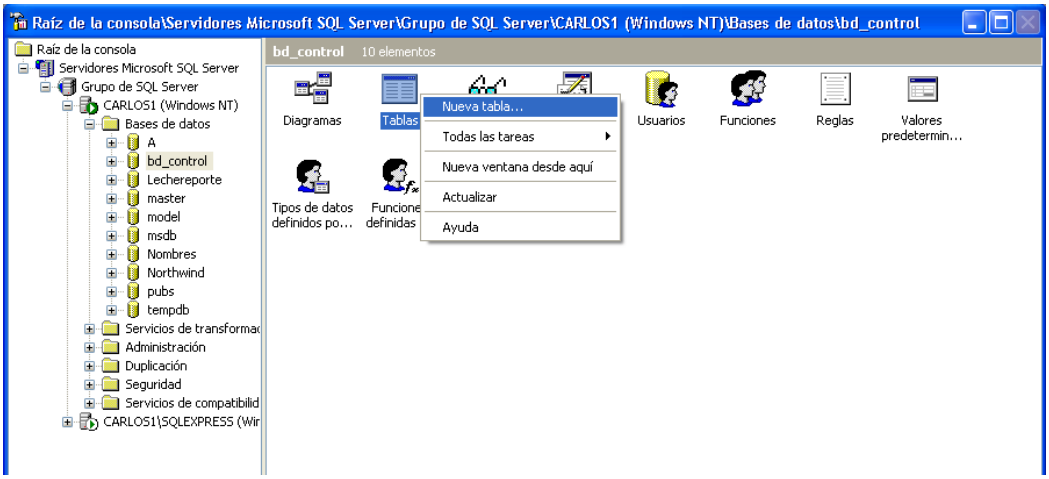


Figura 4.5 Creación de una Nueva Tabla

- Aparece la ventana de la Figura 4.6 donde se crea las columnas que se necesitan en la tabla. En la ventana se especifica el nombre de la columna, el tipo de dato que va a contener la columna y la longitud de los datos.

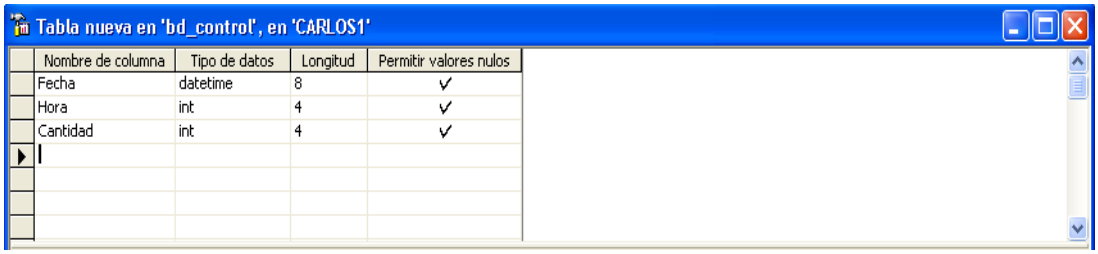


Figura 4.6 Creación de las columnas que contiene la Tabla

- Una vez realizados estos pasos, la Base de Datos se encuentra lista para ser usada por los Software Sharp Develop y Visual .Net 2005.

- Para la aplicación de recepción de datos se creó 2 tablas en la base de datos, para ingreso de datos y para validación de usuarios.

Para la seguridad de los datos es necesario crear un usuario de acceso a las bases de datos, de esta forma al momento de ingresar a la base de datos se solicitará ingresar el usuario y contraseña:

- Abrir el Administrador Corporativo de Microsoft SQL Server, abrir el servidor local, abrir la carpeta seguridad, dar click derecho en inicios de sesión y seleccionar nuevo inicio de sesión, como se muestra en la Figura 4.7.

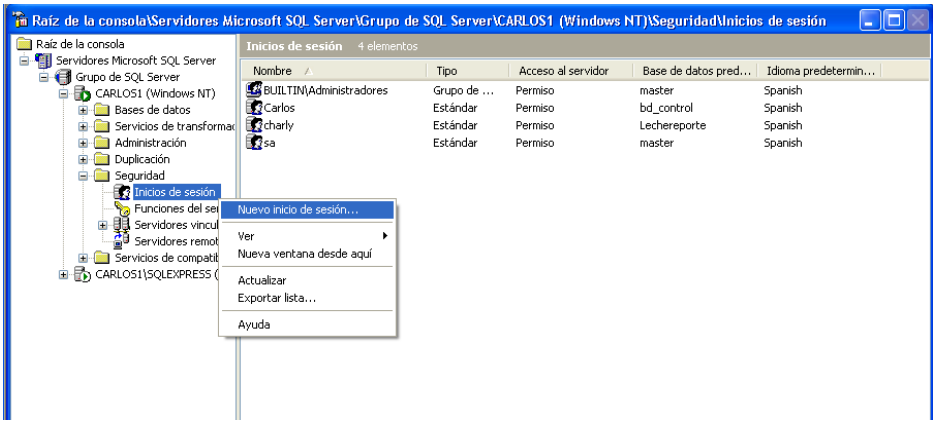


Figura 4.7 Administrador Corporativo de Microsoft SQL Server

- En la ventana propiedades de inicio de sesión, se selecciona la pestaña general y se especifica los campos mostrados en la Figura 4.8.

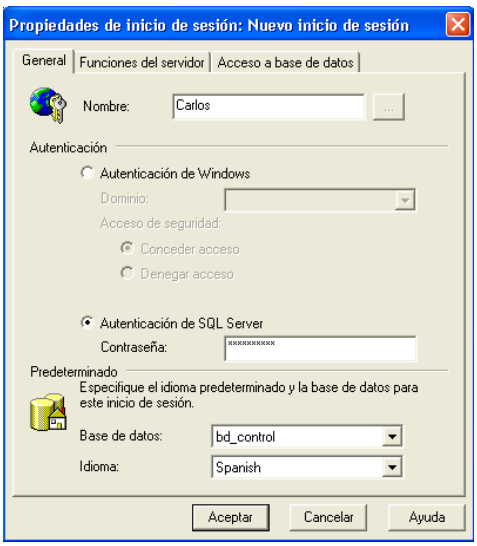


Figura 4.8 Creación de nombre de Sesión de Usuario

- La Figura 4.9 muestra como se selecciona la pestaña Acceso a Base de Datos en donde se selecciona la Base de Datos que tendrá acceso el usuario de acceso creado.

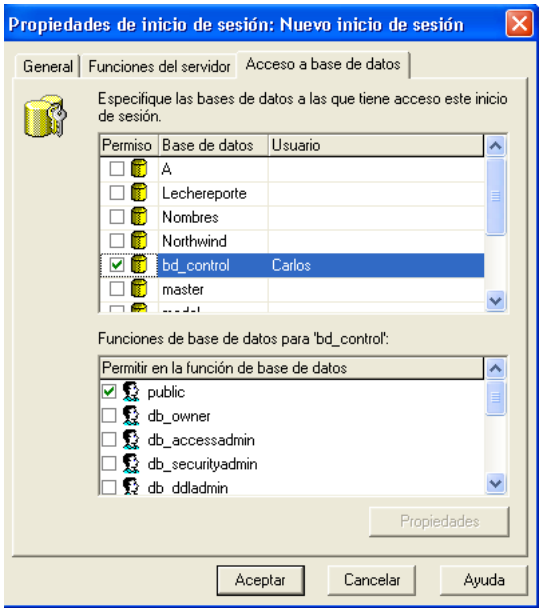


Figura 4.9 Selección de Base de Datos que tendrá acceso el Usuario creado

- De esta forma cada vez que se abra el servidor, el administrador corporativo pide nombre de usuario y contraseña; el usuario que ingrese solo tendrá acceso a las Bases de Datos establecidas en la creación de inicio de sesión.

4.3.2. Manejo del Modem y Base de Datos (Sharp Develop)

El diseño del manejo del Modem y Base de Datos se lo realizó de la siguiente manera:

- Creación de 2 clases: ClsBD (para manejo de Base de Datos) y ClsGSM (para manejo del Modem). Cada una de las clases tiene métodos (funciones) que son llamados por los diferentes formularios de la aplicación del Sistema Receptor.

- Creación de los siguientes formularios:
  - Formulario Principal
  - Formulario Acceso
  - Formulario Usuarios
  - Formulario de Ingreso Manual

- **Clase ClsBD**

Para desarrollar la clase siempre al inicio se declara la cabecera para la ejecución de la clase, la siguiente cabecera usada en esta clase permite el manejo de bases de datos:

```
using System;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Data;
```

Una vez definida la cabecera de la clase se procede a desarrollar la clase:

Definición de variables

Se define las siguientes variables:

```
string cadena  = "server=(local): database = bd_control; uid=Carlos;
password=ciml";
string consulta = "";
SqlCommand command;
Public SqlConnection conexion;
```

La variable cadena permite el acceso a la Base de Datos con la que se va a trabajar (en el caso de la aplicación la Base de Datos es bd_control), también contiene el usuario y contraseña de seguridad para el acceso a la Base de Datos. Las variables command y conexión permiten el manejo de la Base de Datos**.**

Método ClsBD

Este método permite a la clase crear la conexión SQL para el acceso a la Base de Datos, mediante la siguiente línea de código:

```
public clsBD()
{
conexion = new SqlConnection(cadena);
}
```

Método Contar Igual

Este método permite validar el ingreso de usuarios y contraseña, devolviendo una variable con el valor de 1 si existe la coincidencia de usuario o contraseña en la base de datos.

```
public int contar_igual(string tabla, string registro, string valor)
   {
   int num = 0;
   try
   {
         string cadena;
         cadena = "select count(*) from " + tabla + " where " + registro
         + " = '" + valor + "'";
         conexion.Open();
         command = new SqlCommand(cadena, conexion);
         //Ejecutar un conteo, devuelve entero
```

```
            num = Convert.ToInt16(command.ExecuteScalar());
            conexion.Close();

    }
    catch(Exception ex)
    {
            MessageBox.Show(ex.Message);
            Application.Exit();
    }
    return num;
    }
```

El comando SQL *select count* permite realizar consultas en la base de datos. En este caso el comando SQL revisa si existen coincidencias de usuario y contraseña en la base de datos, para permitir el ingreso a la aplicación de recepción de datos.

La variable cadena contiene el formato de la búsqueda en lenguaje SQL, donde se especifica las condiciones de búsqueda (tablas y campos).

Método Agregar

Este método permite el ingreso de datos a la Base de Datos.

```
    public void agregar(string tabla, string [] campos, string [] datos)
    {
      try
      {
            string dato = "";
            string campo = "";

            for (int i=0; i<campos.Length; i++)
            {
```

```
            campo += campos[i] + ",";
            dato += "'" + datos[i] + "',";
            }
            if (dato.Length > 0) dato = dato.Remove(dato.Length - 1);
            if (campo.Length > 0) campo = campo.Remove(campo.Length
            - 1);
            string cadena = "insert into " + tabla + " (" + campo + ") values
            (" + dato + ")";

            conexion.Open();
            command = new SqlCommand(cadena, conexion);
            //Ejecuta consulta de acción
            command.ExecuteNonQuery();
            conexion.Close();
            MessageBox.Show("Dato Guardado");
        }
        catch(Exception ex)
        {
            conexion.Close();
            MessageBox.Show(ex.Message);
        }
    }
```

El comando SQL *insert into* permite el ingreso de datos a una Base de datos, para ello se debe especificar la tabla, el campo y el dato a ser ingresado.

Es importante ejecutar el método ExecuteNonQuery(), cuando se modifica la Base de Datos.

Este método devuelve un mensaje indicando que el dato ha sido guardado.

Metodo Eliminar

Este método permite borrar un dato de la Base de Datos.

```
public void eliminar(string tabla, string registro, string valor)
{
    string cadena = "delete from " + tabla + " where " + registro + " = '"
    + valor + "'";
    conexion.Open();
    command = new SqlCommand(cadena, conexion);
    //Ejecuta consulta de acción
    command.ExecuteNonQuery();
    conexion.Close();
}
```

El comando SQL *delete from* permite borrar datos de una Base de datos, para ello se debe especificar la tabla, el campo y el dato a ser borrado.

Método Grid_todo

Este método permite consultar y mostrar todos los datos de una tabla de la Base de datos.

```
public DataTable grid_todo(string tabla)
{
  DataTable dt = new DataTable();
  try
  {
        string cadena = "select * from " + tabla;
        conexion.Open();
```

```
            command = new SqlCommand(cadena, conexion);
            SqlDataAdapter da = new SqlDataAdapter(command);
            da.Fill(dt);
            conexion.Close();
    }
    catch (Exception ex)
    {
            MessageBox.Show(ex.Message);
    }
    return dt;
}
```

Este método devuelve una variable de tipo DataTable, esta variable contiene todos los datos consultados de una tabla de la Base de Datos.

Método Grid

Este método permite consultar datos, en el caso de la aplicación del sistema receptor, permite consultar los datos en fechas y horas específicas.

```
public DataTable grid(string tabla, string []campos, string[] datos)
{
    DataTable dt = new DataTable();
    try
    {
            consulta = "select * from " + tabla + " where ";
            consulta += campos[0] + " between '" + datos[0] + "' and '" +
            datos[1] + "' AND ";
            consulta += campos[1] + " between '" + datos[2] + "' and '" +
            datos[3] + "';";

            conexion.Open();
            command = new SqlCommand(consulta, conexion);
```

```
            SqlDataAdapter da = new SqlDataAdapter(command);
            da.Fill(dt);
            conexion.Close();
        }
        catch (Exception ex)
        {
            conexion.Close();
            MessageBox.Show(ex.Message);
        }
        return dt;
    }
```

Este método devuelve una variable de tipo DataTable, esta variable contiene todos los datos consultados en las horas y fechas específicas de una tabla de la Base de Datos.

Método Validar_Ingreso_Manual

Este método permite validar el ingreso manual de datos, el método busca si existen datos similares en la tabla de la Base de Datos, si es así no almacena los datos, caso contrario, los almacena.

```
public DataTable validar_ingreso_manual(string tabla, string[] campos,
string[] datos)
{
    DataTable dt = new DataTable();
    try
    {
            consulta = "select * from " + tabla + " where ";
            consulta += campos[0] + " = '" + datos[0] + "' and ";
            consulta += campos [1] + " = '" + datos[1] + "';";

            conexion.Open();
```

```
            command = new SqlCommand(consulta, conexion);
            SqlDataAdapter da = new SqlDataAdapter(command);
            da.Fill(dt);
            conexion.Close();
        }
        catch (Exception ex)
        {

            conexion.Close();
            MessageBox.Show(ex.Message);
        }
    return dt;
    }
```

Este método devuelve una variable de tipo DataTable, que contiene los datos similares al ingreso manual, si es que no existen datos similares esta variable esta vacía.

- **ClsGSM**

Para trabajar con el Modem y procesar los SMS, es muy importante tener en cuenta como es el formato de los SMS, el cual se muestra en la Figura 4.10:

```
+CMGL: 1,"REC READ","+59395530889",,"10/06/26,06:06:29-20"
100
+CMGL: 2,"REC READ","+59395530889",,"10/06/26,06:10:53-20"
200
+CMGL: 3,"REC READ","+59392754016",,"10/07/21,19:48:11-20"
1500
```

Figura 4.10 Formato de SMS para sistema Receptor

Para desarrollar la clase siempre al inicio se declara la cabecera para la ejecución de la clase, la siguiente cabecera usada en esta clase permite el manejo de los puertos de comunicación:

```
using System;
using System.IO;
using System.IO.Ports;
using System.Windows.Forms;
```

Una vez definida la cabecera de la clase se procede a desarrollar la clase:

Definición de variables

Se define las siguientes variables:

```
SerialPort puerto = new SerialPort();
char c_z = '\x001a';
```

La variable puerto, permite crear un puerto de comunicaciones. La variable c_z, almacena la representación ascii de las teclas control Z.

Método Vel_com

Este método define la velocidad en baudios a la que va a atrabajar el puerto serial.

```
public void vel_com(int vel)
{
   puerto.BaudRate = vel;
}
```

Método Cargar_puerto

Este método define el puerto que se va a utilizar para la comunicación.

```
public void cargar_puerto(string com)
{
   puerto.PortName = com;
}
```

Método Abrir_puerto

Este método sirve para abrir el puerto de comunicaciones que se va a utilizar para la aplicación del sistema receptor.

```
public void abrir_puerto()
{
   try
   {
         puerto.Open();
   }
   catch(Exception ex)
   {
         MessageBox.Show(ex.Message);
   }
}
```

Método Cerrar_puerto

Este método sirve para cerrar el puerto de comunicaciones que se va a utilizar para la aplicación del sistema receptor.

```
public void cerrar_puerto()
{
```

```
    puerto.Close();
}
```

Método Leer

Este método sirve para leer todos los datos que se encuentran el buffer del puerto de comunicaciones.

```
public string leer()
{
    string msj = "";
    try
    {
            abrir_puerto();
            msj += puerto.ReadExisting();
            cerrar_puerto();
    }
    catch(Exception ex)
    {
    msj = ex.Message;
    }
return msj;
}
```

Este método devuelve una variable de tipo string que contiene los datos que se encuentran en el buffer de comunicaciones. Para leer el buffer de comunicaciones se utiliza el método ReadExisting(), que permite leer todos los datos que se encuentran en el buffer.

Método Leer_modem

Este método permite revisar y leer todos los mensajes que se encuentran en la memoria del Modem. Para leer todos los SMS de la memoria del Modem se utiliza le comando AT+CMGL, para ello es necesario configurar el Modem para que almacene los SMS en su memoria, mediante el comando AT+CNMI=3,1,0,0.

```
public string leer_modem()
{
   string msj = "";
   try
   {
         escribir("AT+CNMI=3,1,0,0,0");
         escribir("AT+CMGL=\"ALL\"");
         msj = leer();
   }
   catch(Exception ex)
   {
         msj = ex.Message;
   }
   return msj;
}
```

Este método devuelve una variable de tipo string que contiene los datos de todos los SMS almacenados en la memoria del Modem.

Método Existe_msj

Este método permite comprobar si existen SMS en la memoria del Modem.

```
public bool existe_msj()
{
   string msj = "";
   bool existe = false;
   try
   {
         escribir("AT+CNMI=3,1,0,0,0");
         escribir("AT+CMGL=\"ALL\"");
         msj = leer2();
         if (msj.Length > 0) existe = true;
   }
   catch (Exception ex)
   {
         msj = ex.Message;
   }
   return existe;
}
```

Este método devuelve una variable de tipo bool, si la variable es igual a true, existen SMS en la memoria, caso contrario si la variable es igual a false, no existen SMS en la memoria.

Método Escribir

Este método permite escribir en el puerto de comunicaciones.

```
public string escribir(string texto)
{
   string msj = "";
   try
   {
         abrir_puerto();
         puerto.Write(texto + "\r");
```

```
            cerrar_puerto();
        }
        catch (Exception ex)
        {
            msj = ex.Message;
        }
        return msj;
    }
```

Método Enviar_msj

Este método permite enviar SMS a través del Modem.

```
    public string enviar_msj(string numero, string texto)
    {
        string msj = "";
        try
        {
            escribir("AT+CMGF=1");
            escribir("AT+CMGS=\"" + numero + "\"");
            escribir(texto);
            escribir(c_z + " \r");
            msj = "Petición de estado enviada a " + numero;
        }
        catch (Exception ex)
        {
            msj = ex.Message;
        }
        return msj;
    }
```

Este método devuelve una variable de tipo string indicando que el SMS fue enviado.

Método Procesar_sms

Este método permite procesar el SMS y separarlo por fecha, hora, dato y número telefónico.

```
public string[] procesar_sms(string sms)
{
   string [] sms_procesado =  new string[4];
   int cont = 0;
   try
   {
         cont = sms.IndexOf("/");
         sms_procesado[0] = sms.Substring(cont - 2, 8);
         cont = sms.LastIndexOf(":");
         sms_procesado[1] = sms.Substring(cont - 5, 8);
         cont = sms.LastIndexOf("\"");

         if (sms.Contains("OK")) sms_procesado[2] =
         sms.Substring(cont + 2, sms.Length - cont -  5);
         else sms_procesado[2] = sms.Substring(cont + 2, sms.Length
         - cont - 3);
         if (sms.Contains("D\",\""))
         {
               cont = sms.IndexOf("D\",\"");
               sms_procesado[3] = sms.Substring(cont + 4, 9);
         }

         if (sms.Contains("D\",\"+"))
         {
               cont = sms.IndexOf("D\",\"+");
               sms_procesado[3] = "0" + sms.Substring(cont + 8, 8);
         }
```

```
    }
    catch(Exception ex)
    {
            MessageBox.Show(ex.Message);
    }
return sms_procesado;
}
```

Este método devuelve un vector que contiene la fecha, hora, dato y número telefónico del mensaje leído de la memoria del Modem. La función Substring permite separar una cadena de caracteres en diferentes cadenas basándose en la ubicación de los caracteres.

Método Borrar_sms

Este método permite borrar los SMS de la memoria del Modem, para ello se utiliza el comando AT+CMGD, donde se especifica la posición del mensaje almacenado.

```
public string borrar_sms(int pos)
{
    string msj = "";
    try
    {
            escribir("AT+CMGD=" + pos);
            msj = "Mensaje " + pos + " borrado";
    }
    catch(Exception ex)
    {
            msj = ex.Message;
    }
return msj;
}
```

Este método devuelve una variable de tipo string indicando la posición del SMS borrado.

- **Formulario Principal**

Diseño Gráfico del Formulario

La Figura 4.11 muestra el cuadro de diálogo del Formulario Principal:

Figura 4.11 Formulario Principal

Diseño del Código del Formulario

La explicación del diseño del Formulario principal se lo realizará a través del diagrama de flujo de la Figura A.15 del Anexo A3. Este formulario maneja el método Revisar Memoria del Modem (), el cual es descrito en la Figura A.16 del Anexo A3.

- **Formulario Acceso**

Diseño Gráfico del Formulario

La Figura 4.12 muestra el cuadro de diálogo del Formulario Acceso:

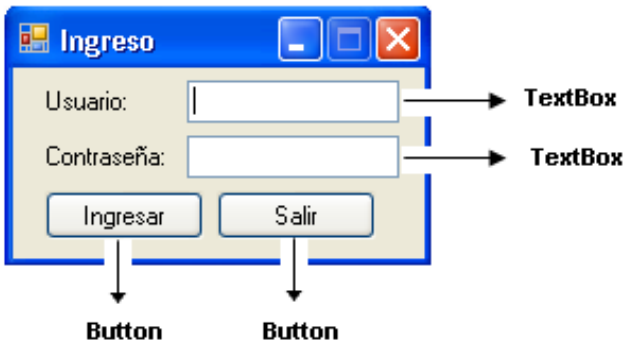


Figura 4.12 Formulario Acceso

Diseño del Código del Formulario

La explicación del diseño del Formulario Acceso se lo realizará a través del diagrama de flujo de la Figura A.17 del Anexo A3.

- **Formulario Usuarios**

Diseño Gráfico del Formulario

Las Figuras 4.13 y 4.14 muestran los cuadros de diálogo del Formulario Usuarios:

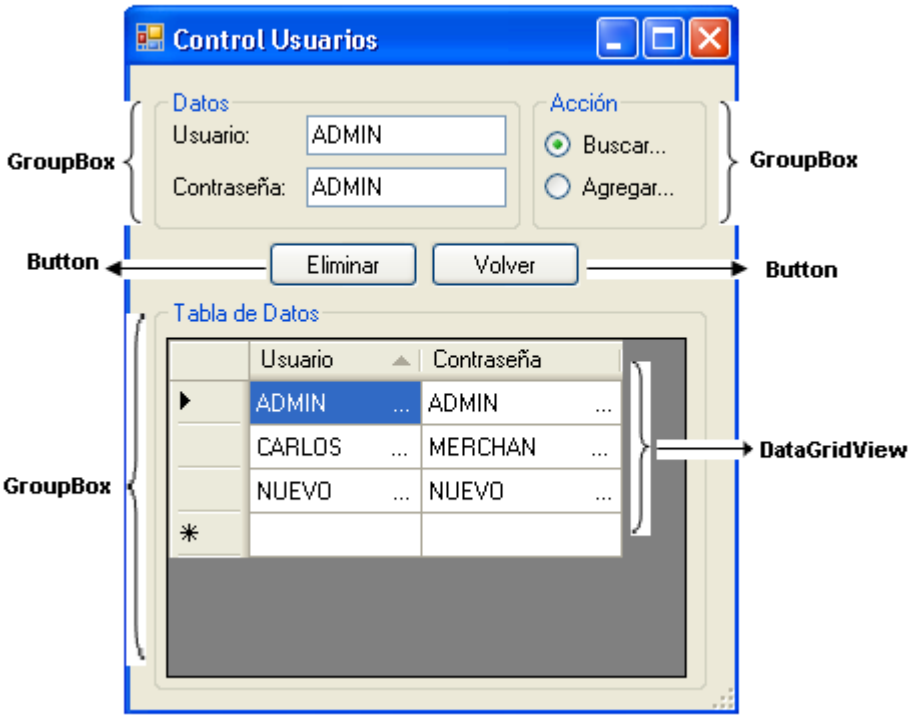


Figura 4.13 Formulario Usuario (Selección RadioButton Buscar)

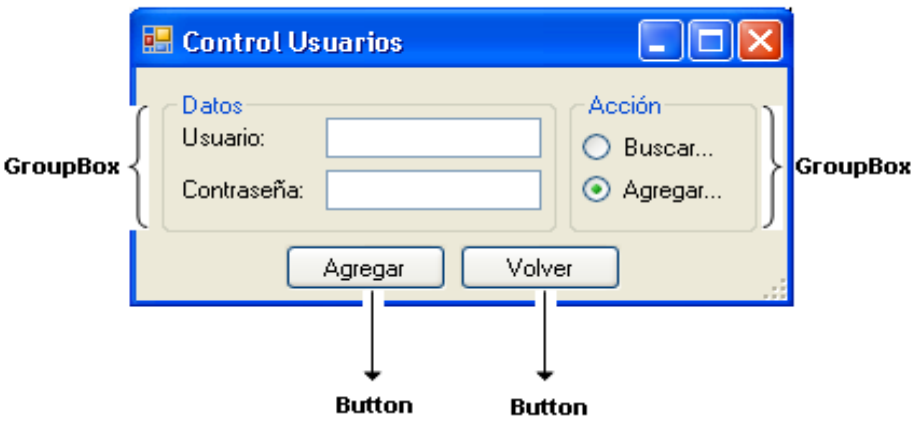


Figura 4.14 Formulario Usuario (Selección RadioButton Agregar)

Diseño del Código del Formulario

La explicación del diseño del Formulario Usuarios se lo realizará a través del diagrama de flujo de la Figura A.18 del Anexo A3.

- **Formulario de Ingreso Manual**

Diseño Gráfico del Formulario

La Figura 4.15 muestra el cuadro de diálogo del Formulario Ingreso Manual:

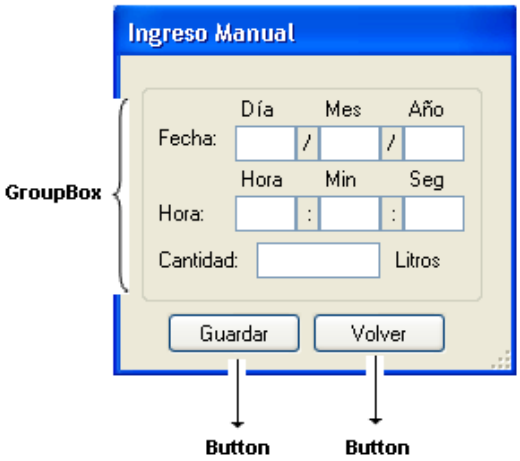


Figura 4.15 Formulario Ingreso Manual

Diseño del Código del Formulario

La explicación del diseño del Formulario Ingreso Manual se lo realizará a través del diagrama de flujo de la Figura A.19 del Anexo A3.

4.3.3. Creación del Reporte (Visual.Net 2005, Crystal Report XI y Microsoft SQL Server 2000)

Para poder realizar el reporte es necesario en primer lugar crear el acceso ODBC[13], mediante los siguientes pasos:

- Ingresar a Panel de Control de Windows.
- Ingresar a Herramientas Administrativas.
- Ingresar a Orígenes de Datos ODBC.
- En la pestaña DSN de sistema presionar el botón agregar.
- En la Figura 4.16, se puede observar la selección del controlador para establecer el origen de datos (en el caso de la aplicación del sistema receptor, se selecciona el controlador SQL server), y presionar el botón Finalizar.

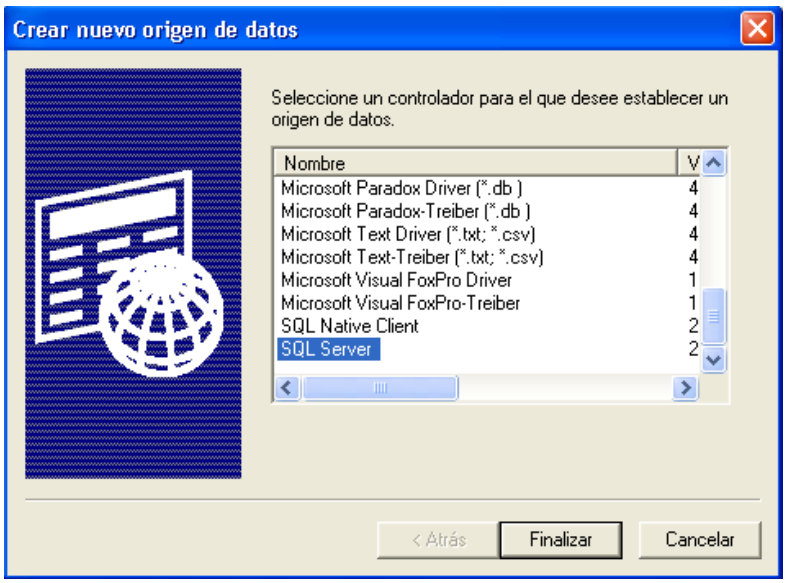


Figura 4.16 Controlador de Origen de Datos (SQL server)

[13] Open DataBase Connectivity

- En el cuadro de diálogo de la Figura 4.17, se establece el nombre del acceso ODBC y el nombre del servidor SQL Server al que va a conectarse, y se presiona siguiente.

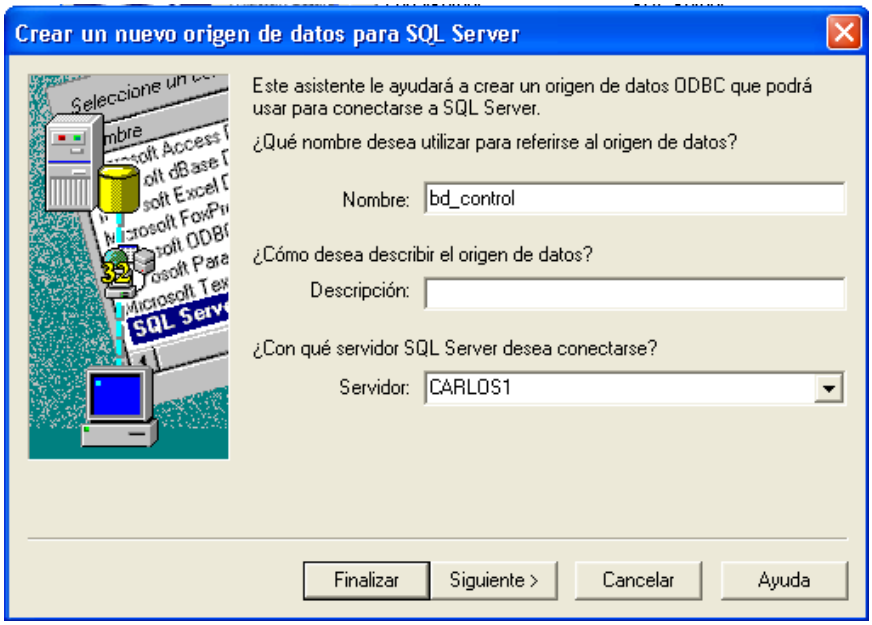


Figura 4.17 Creación de acceso ODBC para conectarse a un SQL Server

- En la Figura 4.18, se selecciona la Base de Datos a la que va conectarse el acceso ODBC.

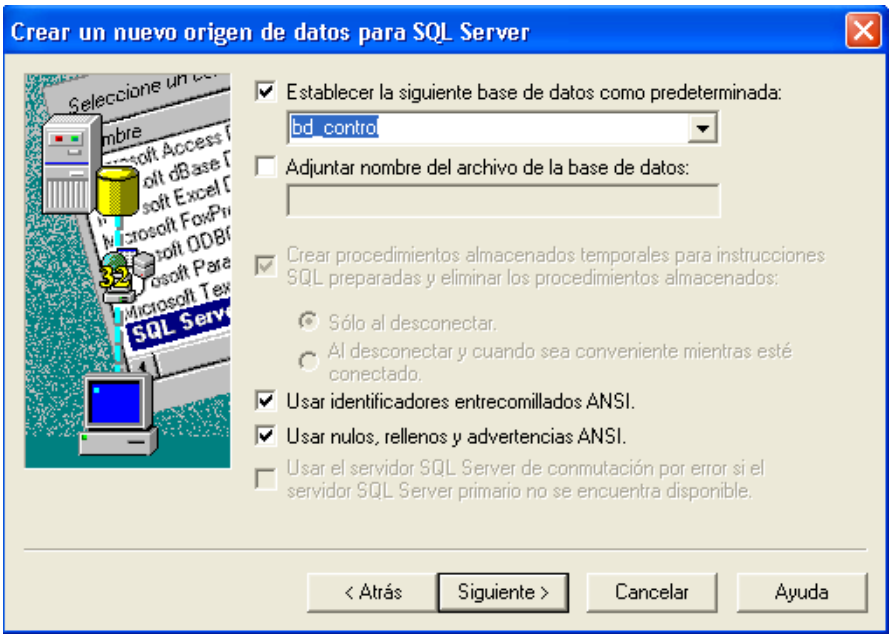


Figura 4.18 Selección de la Base de Datos para la conexión ODBC

- Presionar siguiente hasta que aparezca el botón finalizar, seleccionar finalizar para tener listo el acceso ODBC.

El acceso ODBC es de mucha importancia debido a que permite conectar la Base de Datos de la aplicación con el software Crystal Report XI para que de esta forma se pueda observar los datos consultados en un reporte.

Una vez realizado el acceso ODBC, para la creación del reporte es necesario manejar los siguientes formularios:

- Formulario de Consulta
- Formulario de Reporte

- **Formulario de Consulta**

Diseño Gráfico del Formulario

La Figura 4.19 muestra el cuadro de diálogo del Formulario Consulta:

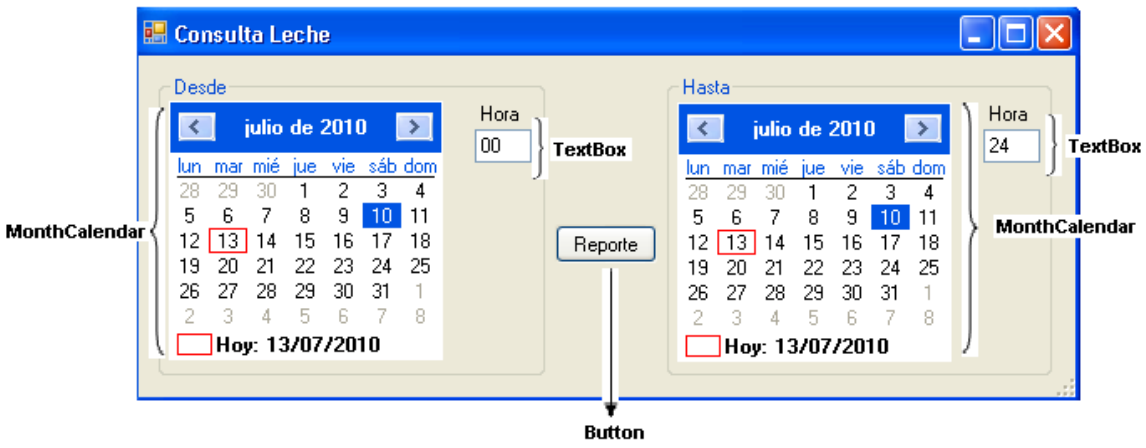


Figura 4.19 Formulario de Consulta

Diseño del Código del Formulario

La explicación del diseño del Formulario Consulta se lo realizará a través del diagrama de flujo de la Figura A.20 del Anexo A3.

- **Formulario de Reporte**

Diseño Gráfico del Formulario

El diseño de este formulario se lo realiza en Visual .Net 2005. Como primer paso se debe abrir el acceso de origen de datos (ODBC):

- La Figura 4.20 muestra, como abrir el menú Datos y seleccionar Agregar nuevo origen de datos.

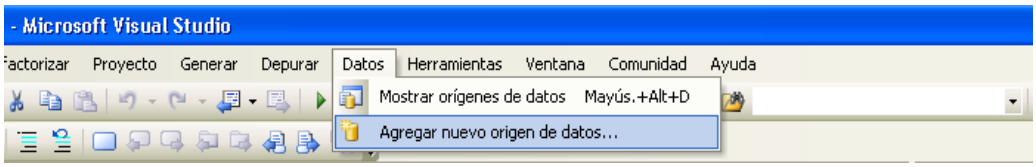


Figura 4.20 Agregar Nuevo Origen de Datos

- En la Figura 4.21 muestra como elegir el tipo de origen de datos, Base de datos.

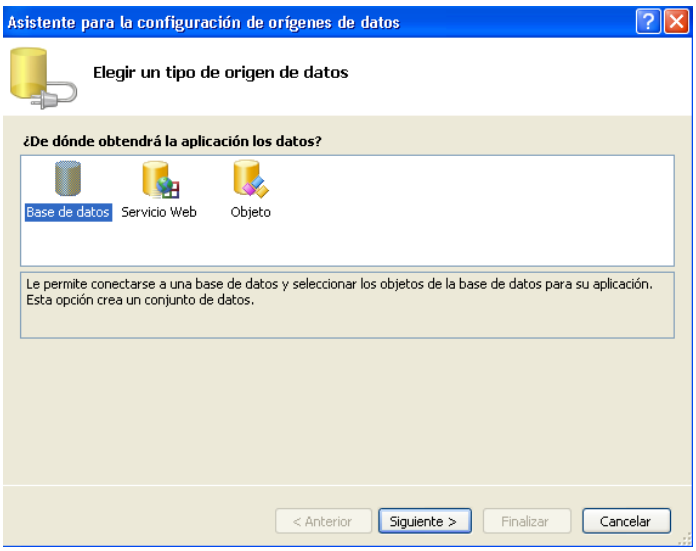


Figura 4.21 Selección del tipo de Origen de Datos

- En la ventana de la Figura 4.22 se muestra como elegir conexión de datos, seleccionar Nueva conexión:

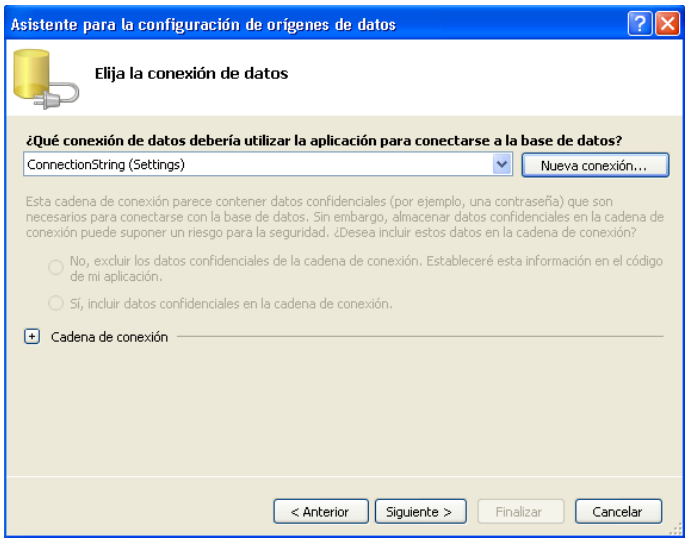


Figura 4.22 Selección Nueva Conexión

- En la ventana de la Figura 4.23, se selecciona el acceso ODBC creado anteriormente.

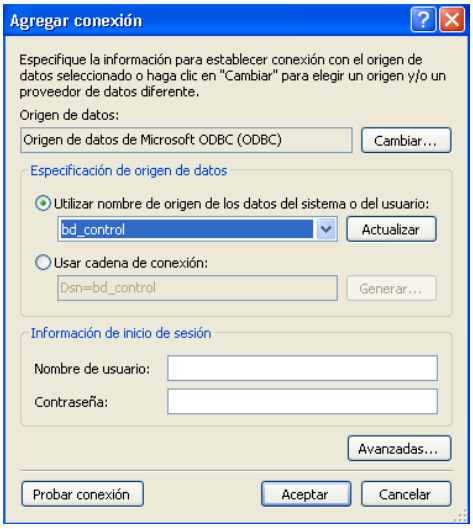


Figura 4.23 Selección de acceso ODBC

- Finalmente se guarda la cadena de conexión, como se muestra en la Figura 4.24:

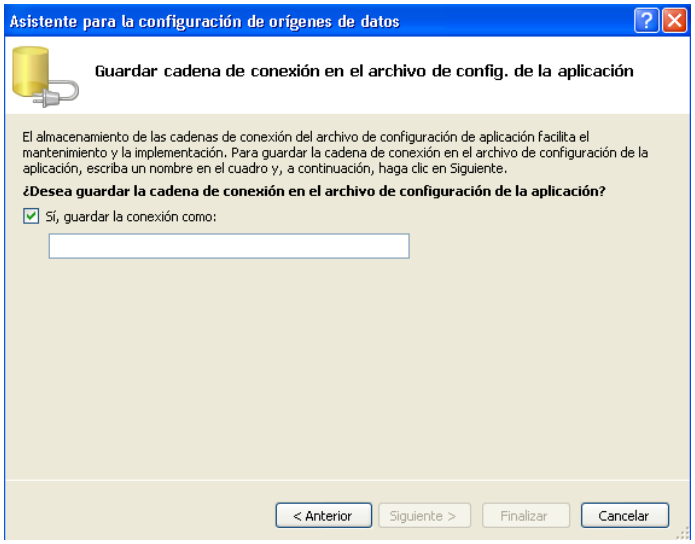


Figura 4.24 Guardar la conexión

Después se debe agregar el elemento Crystal Report donde se mostrará el reporte:

- Dar click derecho en Control Base de Datos del explorador de soluciones, seleccionar Agregar y elegir Nuevo elemento (Ver Figura 4.25).

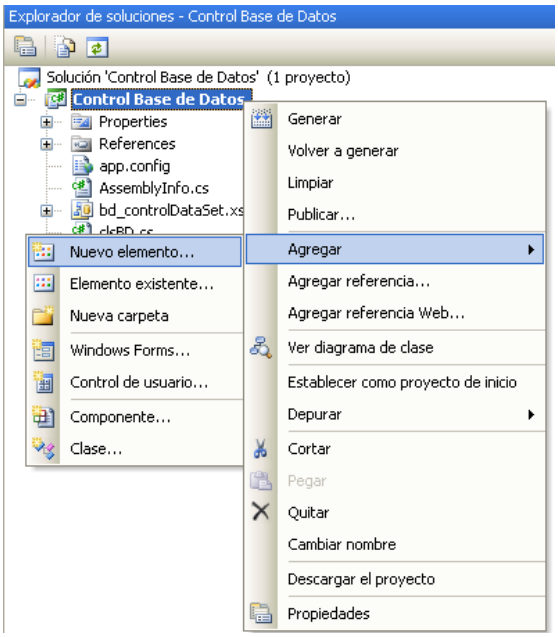


Figura 4.25 Creación del elemento Crystal Report

- En la ventana de la Figura 4.26, seleccionar Crystal Reports, para generar el archivo de reporte.

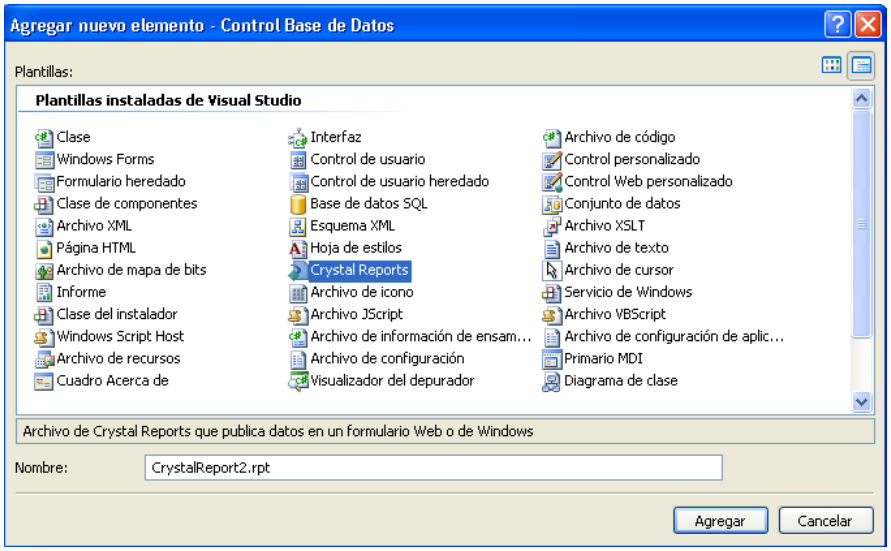


Figura 4.26 Selección del elemento Crystal Report

- A continuación aparece el asistente de la Figura 4.27 donde se genera el archivo de reporte, en donde se escoge el asistente estándar:

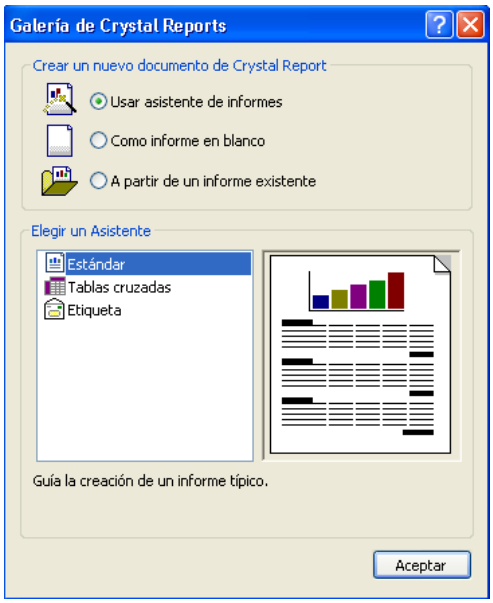


Figura 4.27 Asistente para creación del elemento Crystal report

- En la Figura 4.28, se selecciona la tabla del origen de datos donde se encuentran los datos a ser mostrados en el reporte.

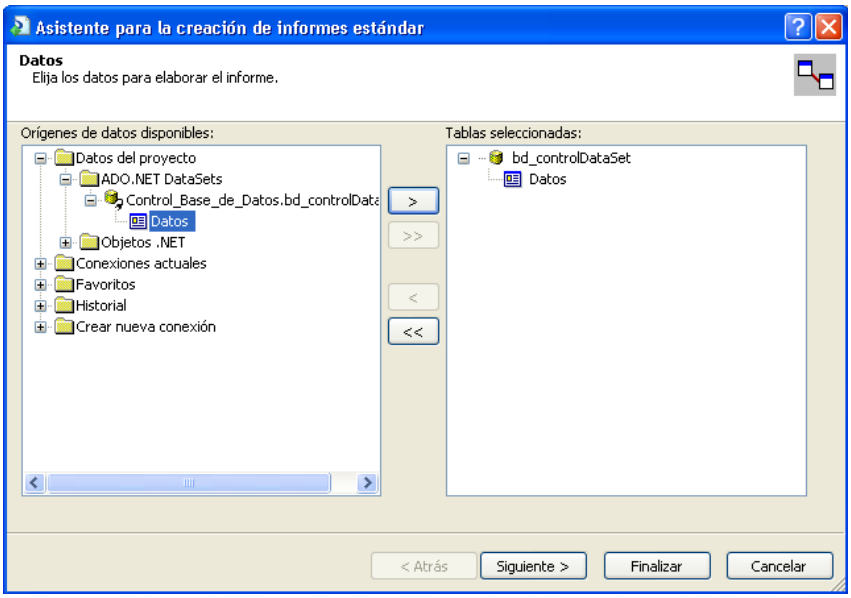


Figura 4.28 Selección de los datos de la Tabla en la Base de Datos

- En la Figura 4.29, se selecciona los campos a ser mostrados en el reporte.

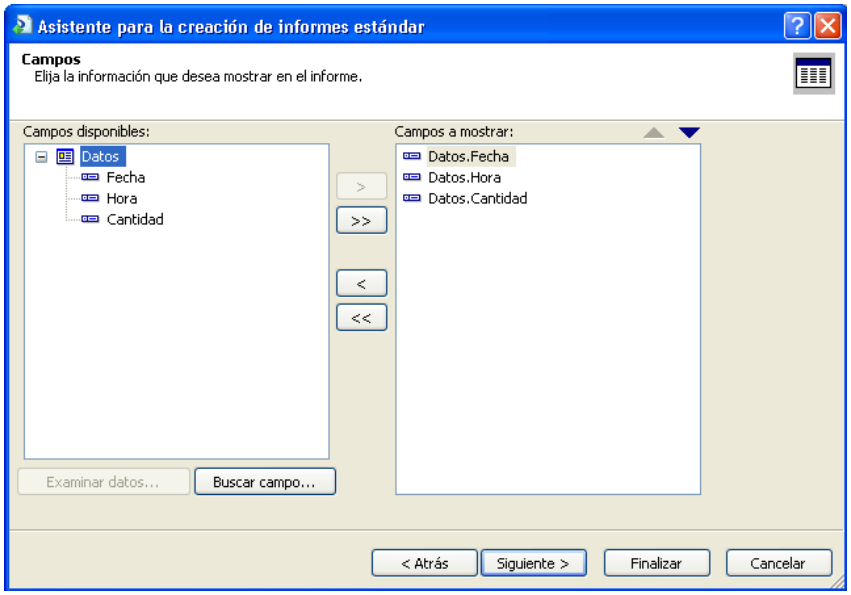


Figura 4.29 Selección de los datos a mostrar en el Reporte

- En la Figura 4.30, se seleccionar como se agrupará la información en el reporte.

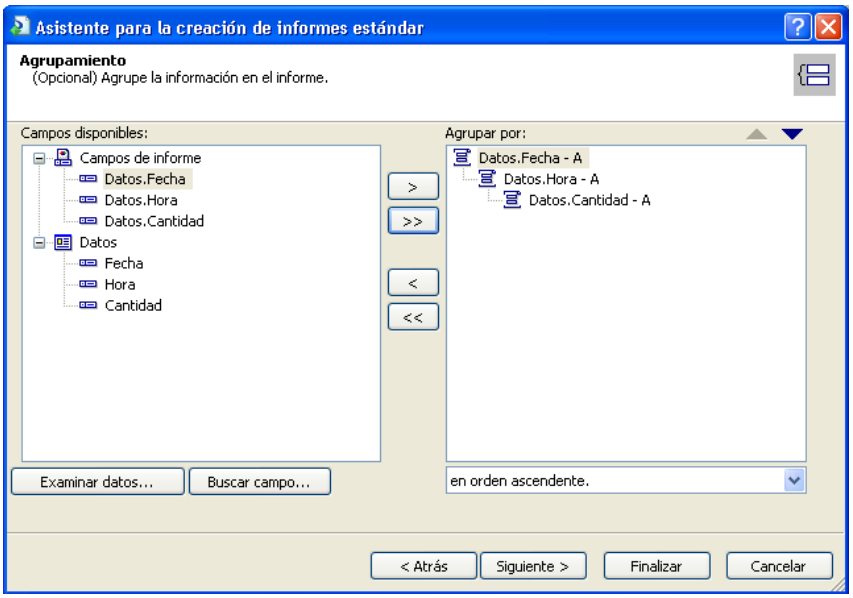


Figura 4.30 Selección de agrupación de la información en el Reporte

- En la Figura 4.31 se muestra como seleccionar que se muestre la sumatoria total del campo cantidad y finalizar el asistente, de esta forma se tendrá creado el informe

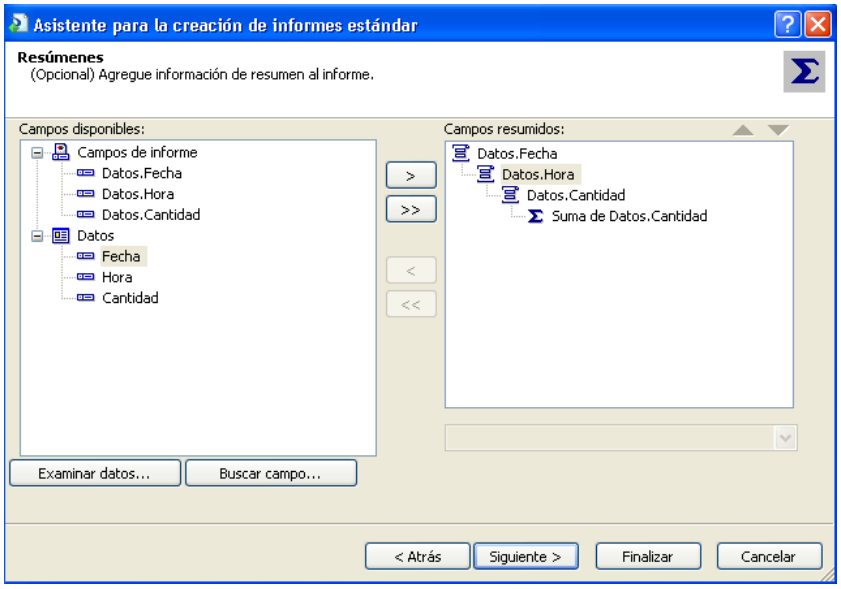


Figura 4.31 Selección para mostrar la sumatoria de los datos del campo cantidad

- Una vez creado el informe, dar click derecho en el reporte y seleccionar insertar gráfico.

- En el asistente de diagramas de la Figura 4.32, se selecciona la pestaña Datos, se establece la siguiente configuración y se finaliza el asistente.

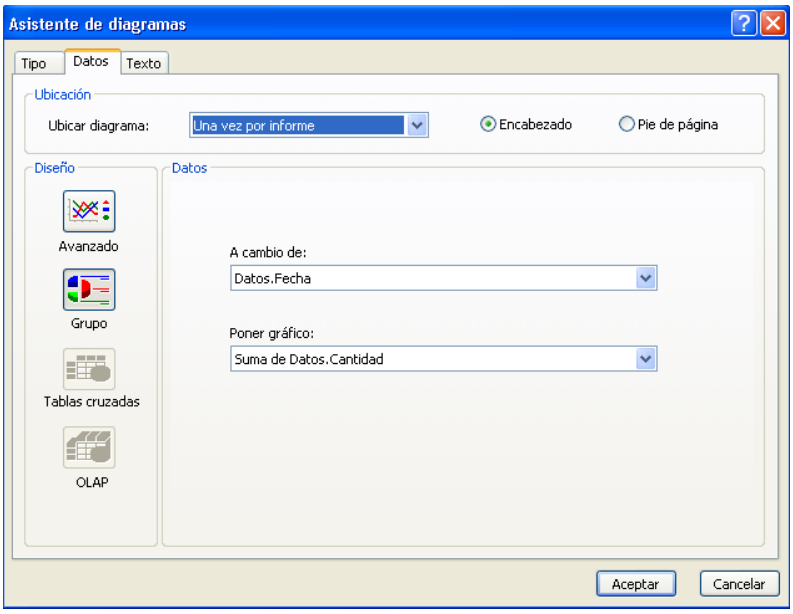


Figura 4.32 Creación del diagrama de barras

- El formato del reporte final se muestra en la Figura 4.33:

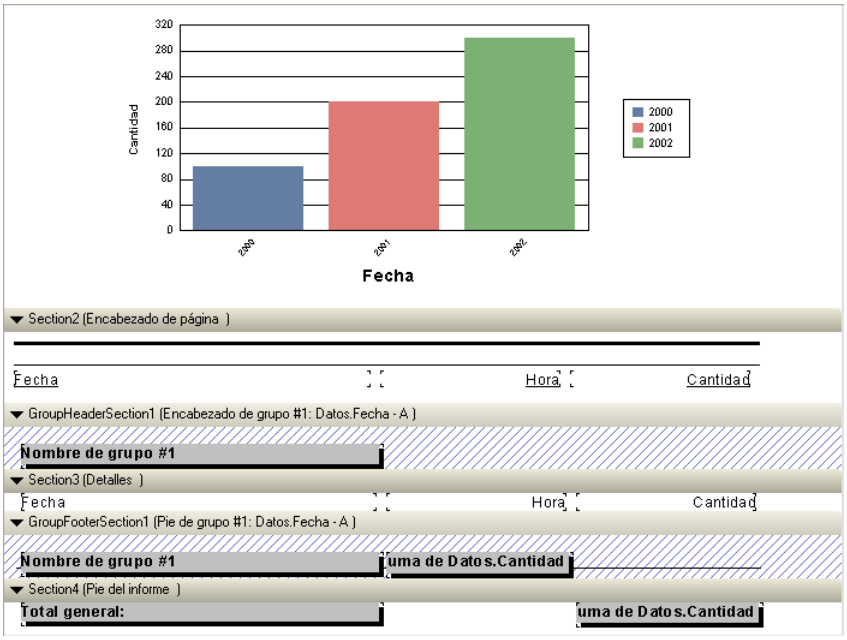


Figura 4.33 Formato del diseño del Reporte

Como siguiente paso se agrega al formulario Reporte la herramienta CrystalReportViewer:

- En el cuadro de herramientas se selecciona la herramienta CrystalReportViewer y se lo inserta en el Formulario, como se muestra en la Figura 4.34.

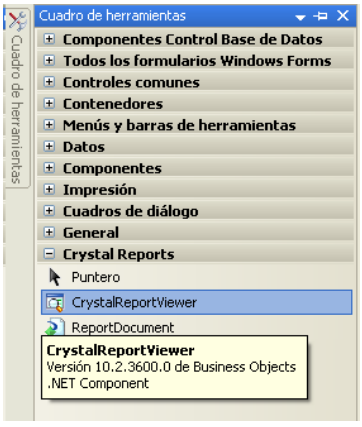


Figura 4.34 Herramienta Crystal Report Viewer

- Se realiza la conexión del CrystalReportViewer con el reporte creado en Crystal Report, mediante la opción elegir un informe de Crystal Report, como se muestra en la Figura 4.35.

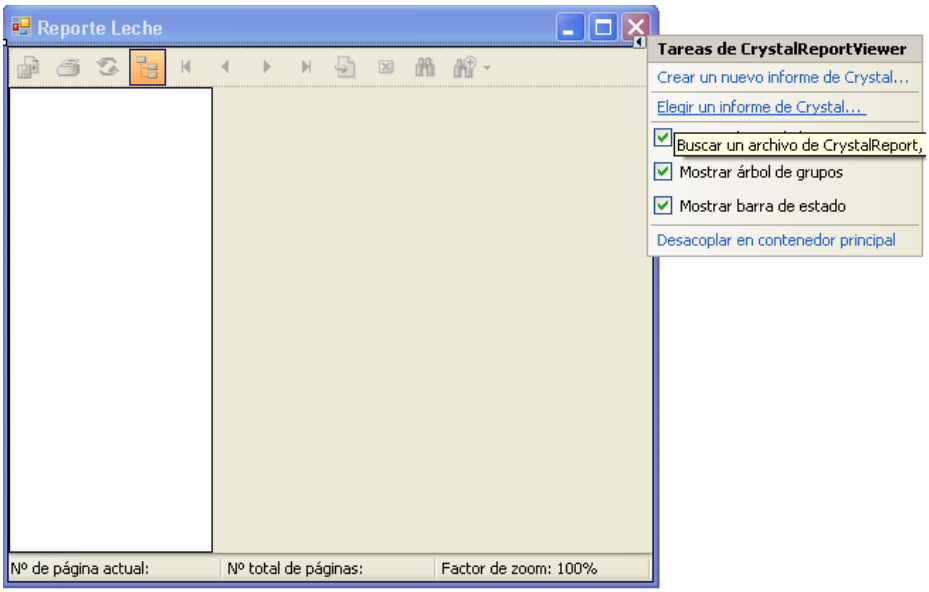


Figura 4.35 Conexión de Crystal Report Viewer con el Reporte creado en Crystal Report

El diseño final del Formulario Reporte se muestra en la Figura 4.36:

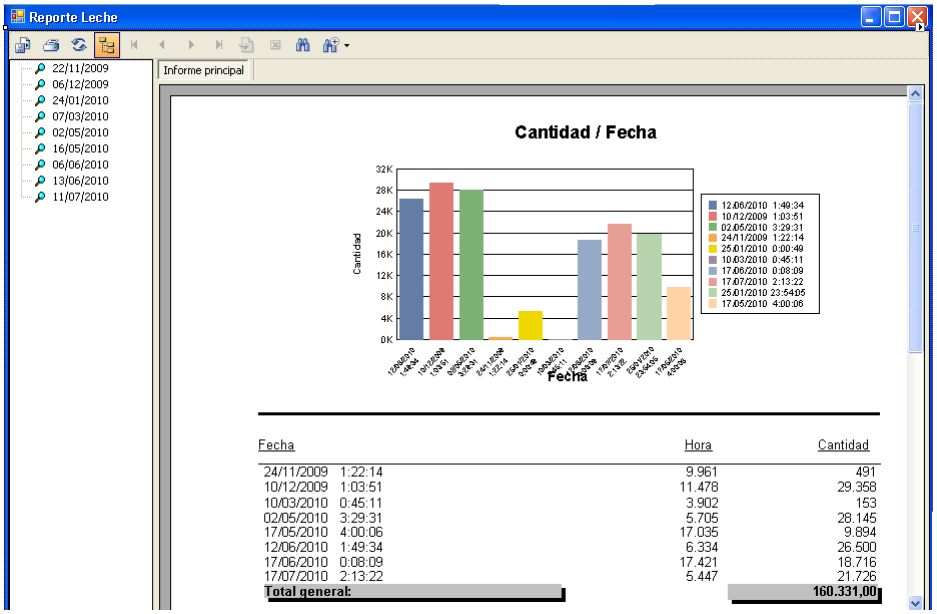


Figura 4.36 Diseño Final del Reporte

Diseño del Código del Formulario

La explicación del diseño del Formulario Reporte se lo realizará a través del diagrama de flujo de la Figura A.21 del Anexo A3.

4.4. Descripción del Sistema Receptor de Datos

A continuación se describirá la aplicación para la recepción de datos. Esta aplicación maneja la decodificación, almacenamiento, consulta y reporte de los datos de la cantidad de leche; la aplicación también maneja validación y creación de usuarios, consulta de estado del equipo transmisor e ingreso manual de datos.

4.4.1. Formulario Validación de Usuarios

Figura 4.37 Formulario Ingreso

El Formulario de la Figura 4.37 permite validar el usuario que va a ingresar; si es un usuario valido permite ingresar a la aplicación de recepción de datos, si no es un usuario valido devuelve el mensaje de la Figura 4.38.

Figura 4.38 Usuario no Registrado

Si es un usuario valido, la aplicación revisa la memoria del Modem para ver si existe SMS, si existe SMS, los guarda si son validos o los elimina si son no validos. Después de la revisión de la memoria del Modem, devuelve un mensaje indicando que no hay mensajes en la memoria del Modem (Ver Figura 4.39), finalmente ingresa a la aplicación de recepción de datos.

Figura 4.39 Memoria del Modem Vacía

4.4.2. Formulario Principal del Sistema de Recepción de Datos

Figura 4.40 Formulario Principal del Sistema Receptor

El formulario de la Figura 4.40 permite controlar todo el sistema de Recepción de datos, a continuación una breve descripción de cada elemento en el formulario:

- **Botón Estado Sensor**: Este botón permite revisar el estado del sistema transmisor, envía un mensaje con la petición de Estado Sensor, el sistema transmisor devuelve un mensaje con Estado Activo (Sistema transmisor con sensor activo) o Estado Pasivo (Sistema Transmisor con sensor pasivo).

- **Botón Ingreso Manual:** Este botón permite ingresar los datos de la recolección de leche manualmente.
- **Botón Ver Reporte:** Este botón permite establecer los intervalos de fecha y hora para ser consultados y mostrados en un reporte.
- **Botón Revisar SMS:** Este botón permite revisar si existen SMS en la memoria del Modem y guardarlos en la aplicación del sistema receptor, si no existen SMS en la memora devuelve un mensaje indicando que no existen SMS en la memoria del Modem.
- **Menú Estado Sensor:** Realiza la misma función que el botón Estado Sensor.
- **Menú Ingreso Manual:** Realiza la misma función que el botón Ingreso Manual.
- **Menú Reporte:** Realiza la misma función que el botón Ver Reporte.
- **Menú Usuarios:** Permite ingresar y eliminar usuarios para el ingreso a la aplicación de recepción de datos.
- **Menú Fecha y Hora:** Muestra la fecha y hora actualizadas.

4.4.3. Formulario Ingreso Manual

Figura 4.41 Formulario Ingreso Manual

El Formulario de la Figura 4.41 permite ingresar manualmente los datos de fecha, hora y cantidad, en casos cuando la aplicación del sistema receptor no ha almacenado los datos.

Si los datos ingresados manualmente ya están registrados en la Base de datos, se devuelve el mensaje “El Dato ya Existe”.

4.4.4. Formulario Ver Reporte

Al ejecutar el botón Ver Reporte se muestra el Formulario Consulta Leche (Figura 4.42), el cual permite seleccionar las fechas y horas de los datos almacenados en la base de datos.

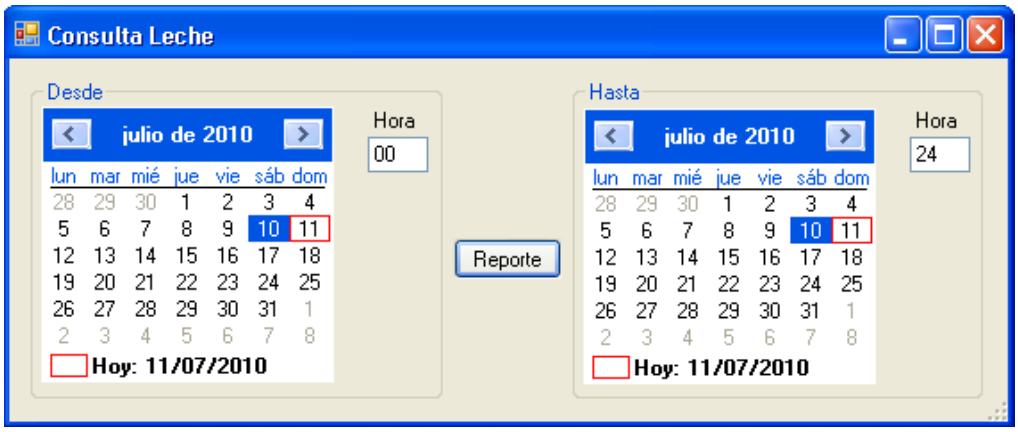
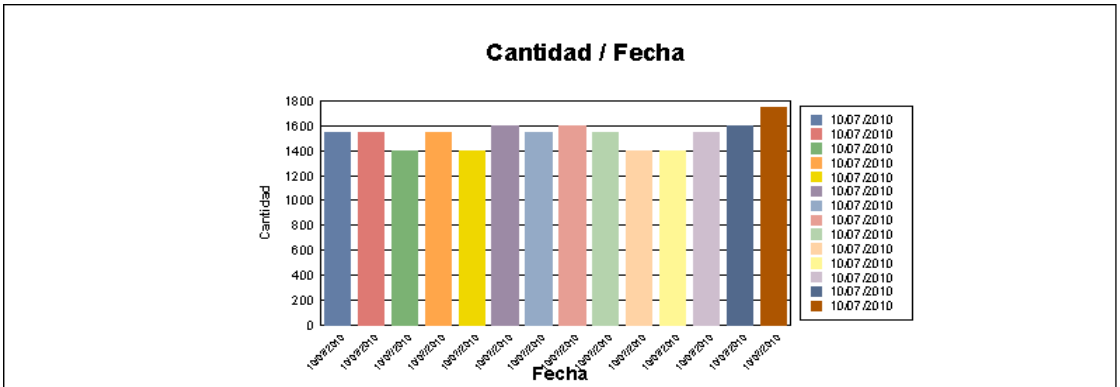


Figura 4.42 Formulario Consulta Leche

Una vez seleccionadas las fechas y horas, se ejecuta el botón Reporte, el cual mostrará en un informe los datos consultados.

El informe mostrará la fecha, hora y cantidad de leche consultadas (Ver Figura 4.43), también mostrará la suma de todas las cantidades de leche consultadas y un gráfico de barras, para obtener una estadística de las cantidades recolectadas de leche.



Fecha	Hora	Cantidad
10/07/2010 0:00:00	10	1.550
10/07/2010 0:00:00	10	1.550
10/07/2010 0:00:00	10	1.400
10/07/2010 0:00:00	10	1.550
10/07/2010 0:00:00	10	1.400
10/07/2010 0:00:00	10	1.600
10/07/2010 0:00:00	10	1.550
10/07/2010 0:00:00	10	1.600
10/07/2010 0:00:00	10	1.550
10/07/2010 0:00:00	10	1.400
10/07/2010 0:00:00	10	1.400
10/07/2010 0:00:00	10	1.550
10/07/2010 0:00:00	10	1.600
10/07/2010 0:00:00	8	1.750
Total general:		**21.450,00**

Figura 4.43 Informe de datos consultados

4.4.5. Formulario Control Usuarios

Este formulario permite realizar las dos siguientes acciones:

- **Buscar y eliminar Usuarios**

El Formulario de la Figura 4.44, permite consultar los usuarios disponibles, también permite eliminar los usuarios que se desee.

Figura 4.44 Buscar y Eliminar Usuarios

- **Agregar Usuarios**

El Formulario de la Figura 4.45, permite ingresar nuevos usuarios para ingresar a la aplicación del sistema receptor de datos.

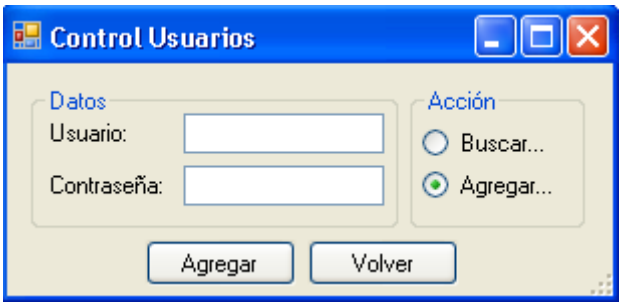


Figura 4.45 Agregar Usuarios

4.5. Costo de los Elementos Electrónicos utilizados en el Sistema Receptor

La Tabla 4.1, muestra el costo de los Elementos Electrónicos utilizados en el Sistema Receptor:

Tabla 4.1 Costo de la implementación del sistema de transmisión

Cantidad	Elementos Electrónicos	Precio unitario	Precio Total
1	MODEM	130	130
1	CABLE ADAPTADOR PUERTO SERIAL - USB	15	15
		TOTAL	145

4.6. Costo Total del Sistema Transmisor y Receptor

La Tabla 4.2, muestra el costo total del Sistema Transmisor y Receptor:

Tabla 4.2 Costo Total del Sistema Transmisor y Receptor

Cantidad	Sistema	Precio unitario	Precio Total
1	TRANSMISOR	220,75	220,75
1	RECEPTOR	145	145
		TOTAL	365,75

CAPITULO 5

5. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- Los objetivos y alcances planteados para el desarrollo de este proyecto, han sido cumplidos en su totalidad satisfactoriamente; creando un sistema de monitoreo y registro de la cantidad de leche en un ordeño mecánico a través del servicio SMS de una red GSM.

- En la actualidad el servicio de SMS es muy utilizado en aplicaciones de monitoreo, debido a que prestan una solución de bajo costo, con lo que se puede obtener el registro de datos monitoreados en diferentes áreas geográficas.

- La transmisión de SMS no es en tiempo real debido a que la transmisión depende del tráfico de la red GSM de la operadora que esté prestando los servicios.

- Para el diseño de la etapa de sensamiento del Sistema Transmisor fue necesario analizar el funcionamiento del sistema de ordeño mecánico, sus elementos e infraestructura, de esta forma se seleccionó el sensor más apto a ser utilizado en el sistema.

- El diseño del sistema transmisor fue realizado en el compilador BASCOM-AVR, debido a que este software proporciona una gran cantidad de instrucciones para el manejo del puerto serial, con el que se maneja el Modem del Sistema Transmisor.

- Para obtener lecturas correctas por parte del sensor es necesario calcular el número de pulsos generados por el sensor a diferentes distancias, con los datos obtenidos se realiza una gráfica en donde se puede obtener la ecuación de la respuesta linealizada del sensor, la cual se aplicará en los cálculos al momento de obtener las distancias generadas por el sensor.

- SharpDevelop 3.2, es una herramienta muy útil para el manejo de lenguajes de programación e interfaces gráficas, con la gran ventaja de ser un Software Libre. El entorno de desarrollo de este Software es totalmente compatible con el Software Microsoft Visual Studio 2005. En la etapa de generación del reporte de la aplicación desarrollada para el Sistema Receptor no fue posible utilizar SharpDevelop debido a que la herramienta Crystal Report no es compatible, motivo por el cual se utilizó el Software Microsoft Visual Studio 2005.

- Para el diseño del Sistema Transmisor se selecciono el microcontrolador de la familia ATMEGA, debido a que este microcontrolador posee una gran variedad de instrucciones para el manejo del puerto serial, que facilitaron el manejo del Modem. Con respecto al tamaño de la memoria Flash utilizada por la aplicación del Sistema Transmisor, se escogió el microcontrolador ATMEGA8 (8 Kbytes memoria Flash) que posee la suficiente memoria para el manejo de la aplicación.

- Para el correcto manejo de las Bases de Datos, fue necesario analizar el lenguaje SQL, sus comandos, clausulas, operadores lógicos y operadores de comparación; estos elementos se combinan en las instrucciones para crear, actualizar y manipular las Bases de Datos.

- El Software Crystal Report permitió generar reportes de una forma flexible y dinámica, debido a que posee un asistente que guía en la creación del reporte, tanto en la conexión con la Base de Datos como en la estructura del reporte.

- Para lograr la conexión entre la Base de Datos y Crystal Report a través de Microsoft Visual Studio, fue necesario crear un acceso ODBC, el cual hace posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué Sistema Gestor de Bases de Datos. Al crear el acceso ODBC es necesario establecer el controlador de origen de datos, el servidor de conexión y la Base de Datos a acceder.

- Al momento del diseño de los Sistemas Transmisor y Receptor fue muy importante conocer el formato con el que se recibiría los SMS, el Sistema Transmisor tiene el formato de SMS AT+CNMI=3,2,0,0,0 (Receptar y eliminar el SMS) y el Sistema Receptor tiene el formato de SMS AT+CNMI=3,1,0,0,0 (Almacenar en memoria el SMS).

- La comunicación I2C, es un bus de comunicación que permite comunicar dispositivos electrónicos, a través de 2 líneas de señal (datos y reloj), lo que es una gran ventaja. Los dispositivos electrónicos se comunican a través de direcciones que indican si el dispositivo va a escribir o leer un dato. Este tipo de comunicación facilitó en el sistema transmisor la comunicación entre el microcontrolador y el reloj calendario (DS1307).

- Al momento de utilizar el Software Progisp 1.6.7, es muy importante configurar de forma correcta los fuses del microcontrolador, ya que si se lo hace de forma incorrecta el microcontrolador puede bloquearse.

- El diseño del Sistema Transmisor permite adaptar un mayor número de sensores para un mejor desempeño de la etapa de sensamiento, por ejemplo sensores de flujo, caudal, entre otros.

- El diseño del Sistema Transmisor fue realizado para obtener el volumen de un tanque de almacenamiento cilíndrico, en caso de cambiar la forma del tanque de almacenamiento, se tendría que cambiar el algoritmo para obtener el volumen del tanque.

5.2. Recomendaciones

- Para el correcto funcionamiento del sensor ultrasónico de nivel PING es necesario tener en cuenta la correcta colocación del sensor sobre el tanque.

- Para obtener una correcta comunicación entre el sistema transmisor y el sistema receptor, se recomienda siempre verificar la correcta conexión del conector serial de los módems GSM.

- Siempre que se realice la configuración inicial de un modem GSM para realizar el envió de SMS, se recomienda enviar el comando AT&W al final de la configuración, de esta forma se guarda todos los parámetros de la configuración inicial.

- Para garantizar el correcto funcionamiento del sistema transmisor, es recomendable conectarlo a un UPS para garantizar su alimentación.

- Para obtener el mejor desempeño del Modem es necesario que trabaje a una velocidad de transmisión de 115200 baudios, motivo por el cual se utiliza en el diseño del equipo transmisor un cristal de 11,059200MHz.

- Es importante actualizar los datos de hora del reloj calendario DS1307, debido a que el reloj calendario se retrasa un intervalo de 1 a 2 minutos por día.

- Para garantizar que los datos de la base de datos no sean modificados, es recomendable crear usuarios de acceso a las bases de datos, de esta forma al momento de ingresar a la base de datos se solicitará ingresar el usuario y contraseña.

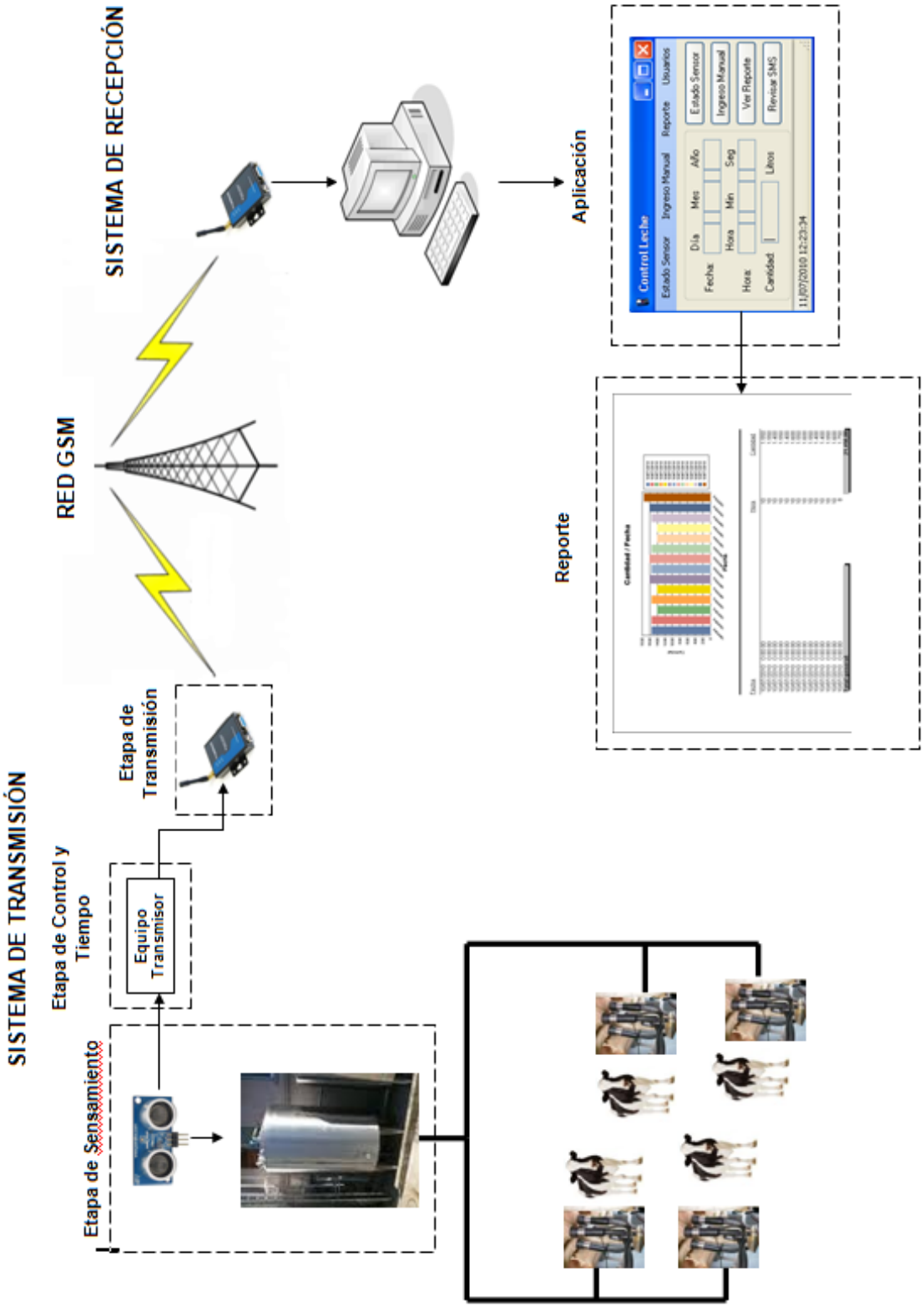
- El necesario tener instalado el Software Service Pack 3.5 en la PC, que va a manejar el Sistema Receptor.

- Para el correcto de funcionamiento del Sistema Receptor, se debe instalar la aplicación en el Sistema Operativo Windows XP.

- Es importante tener mucho cuidado al momento de instalar el sensor en el tanque de almacenamiento, ya que si entra en contacto con algún líquido puede averiarse el sensor.

ANEXOS

ANEXO A1: DIAGRAMA DEL SISTEMA DE MONITOREO


SISTEMA DE TRANSMISIÓN
Etapa de Sensamiento
Etapa de Control y Tiempo
Equipo Transmisor
Etapa de Transmisión
RED GSM
SISTEMA DE RECEPCIÓN
Aplicación
Reporte

ANEXO A2: DIAGRAMAS DE FLUJO DEL SISTEMA TRANSMISOR

Etapa de Sensamiento

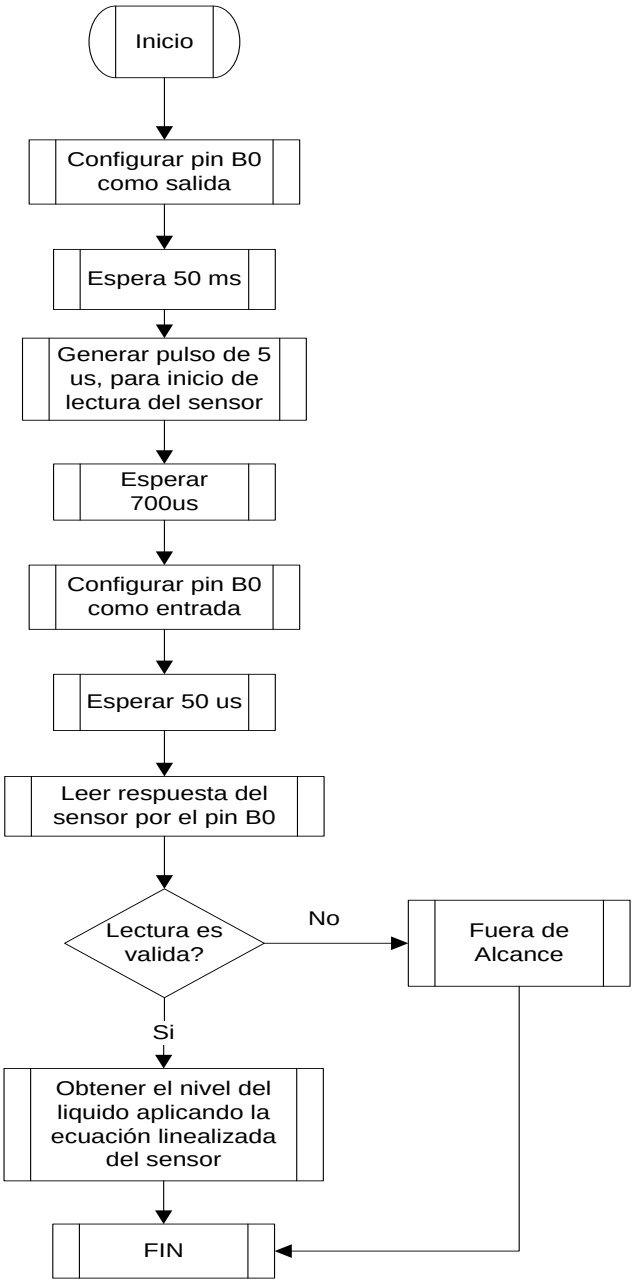


Figura A.1Etapa de Sensamiento

Etapa de Tiempo Principal

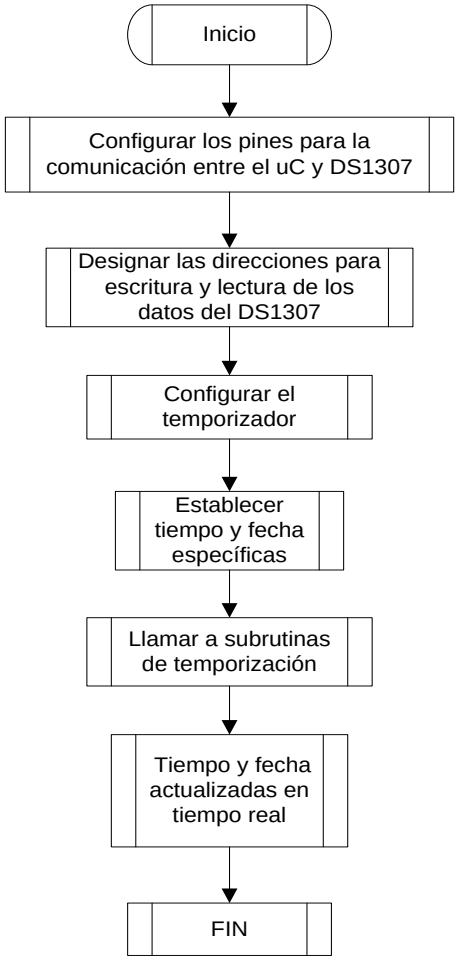


Figura A.2 Etapa de Tiempo Principal

Subrutina Settime

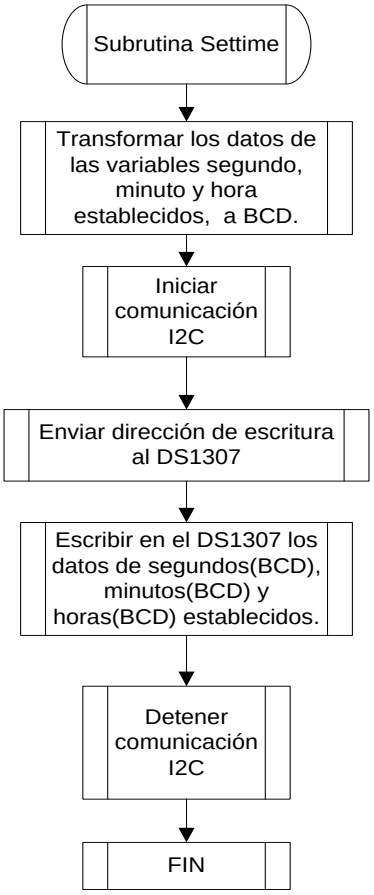


Figura A.3 Subrutina Settime

Subrutina Setdate

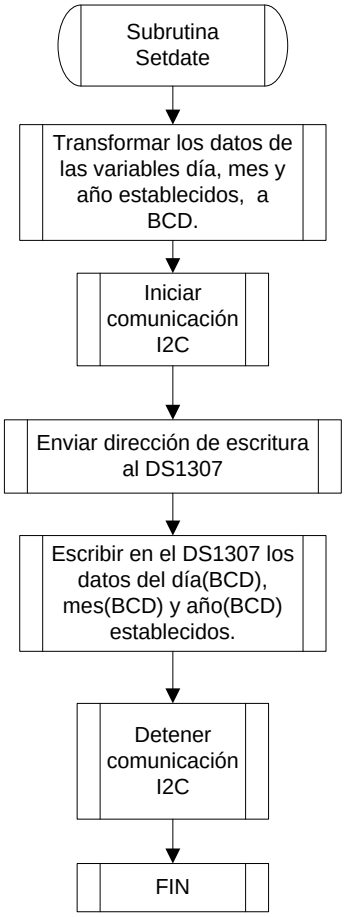


Figura A.4 Subrutina Setdate

Subrutina Getdatetime

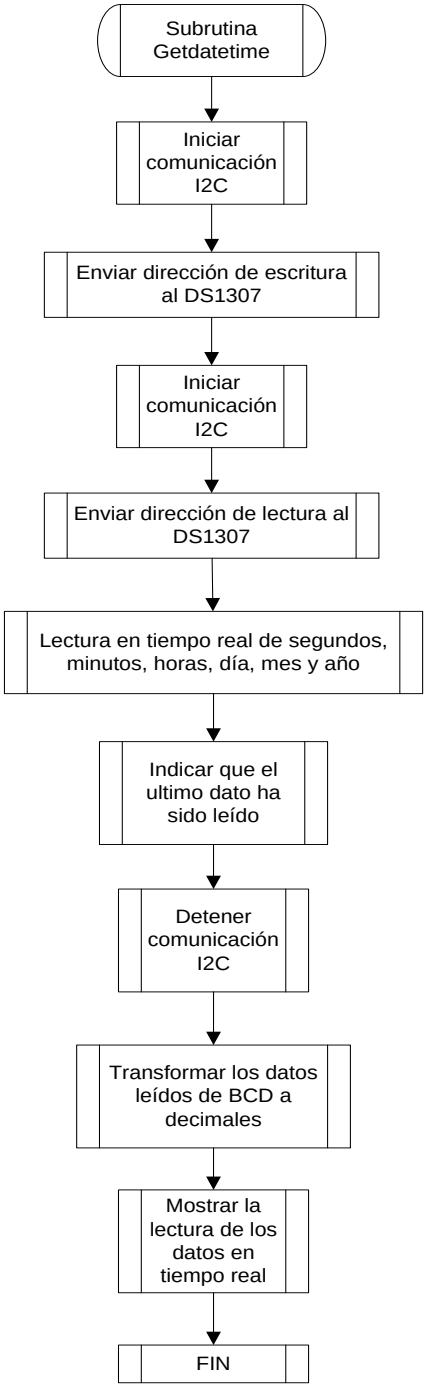


Figura A.5 Subrutina Getdatetime

Función Configuración Inicial del Modem GSM

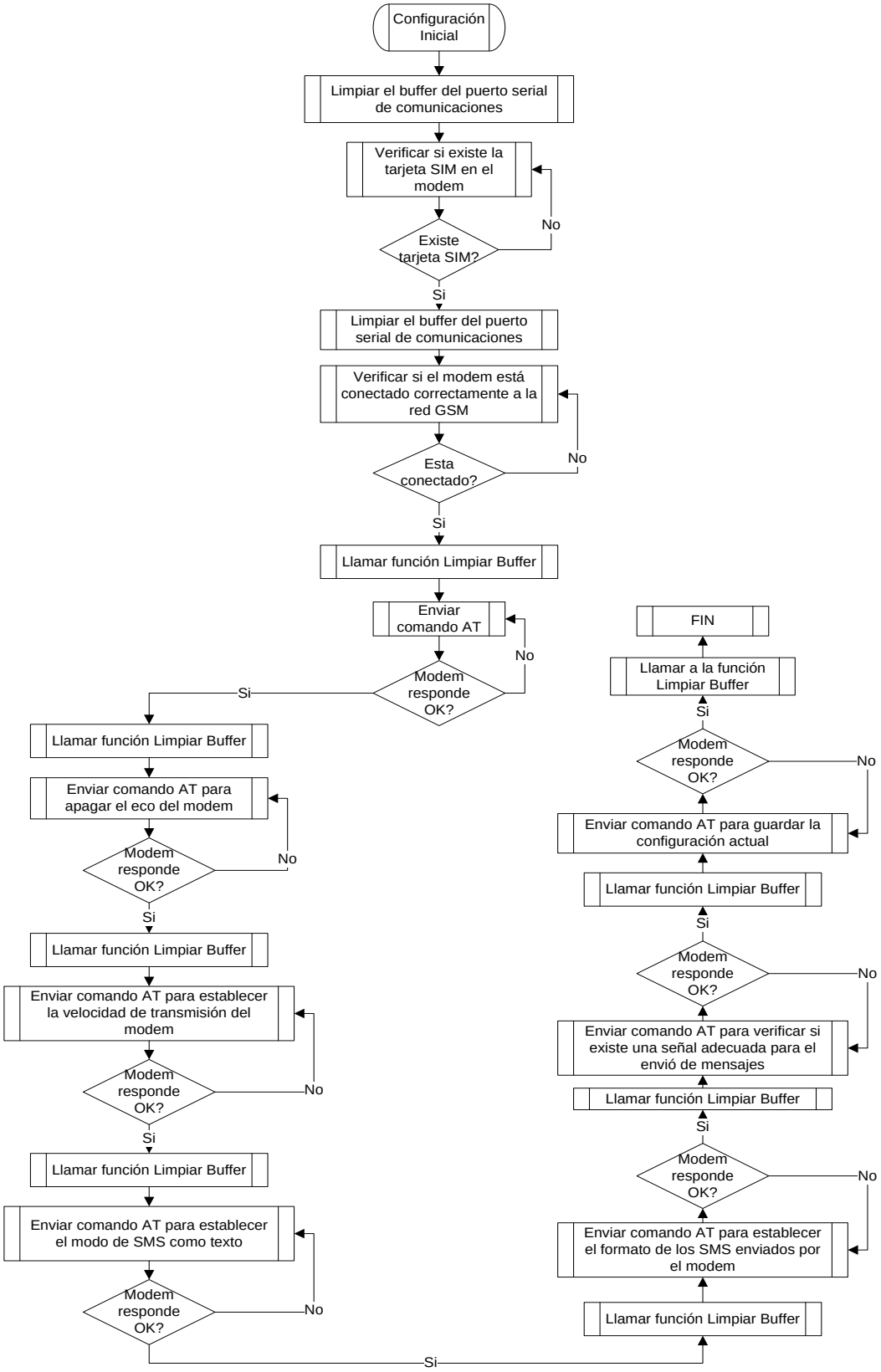


Figura A.6 Función Configuración Inicial del Modem GSM

Función de Envió de Mensajes del Modem GSM

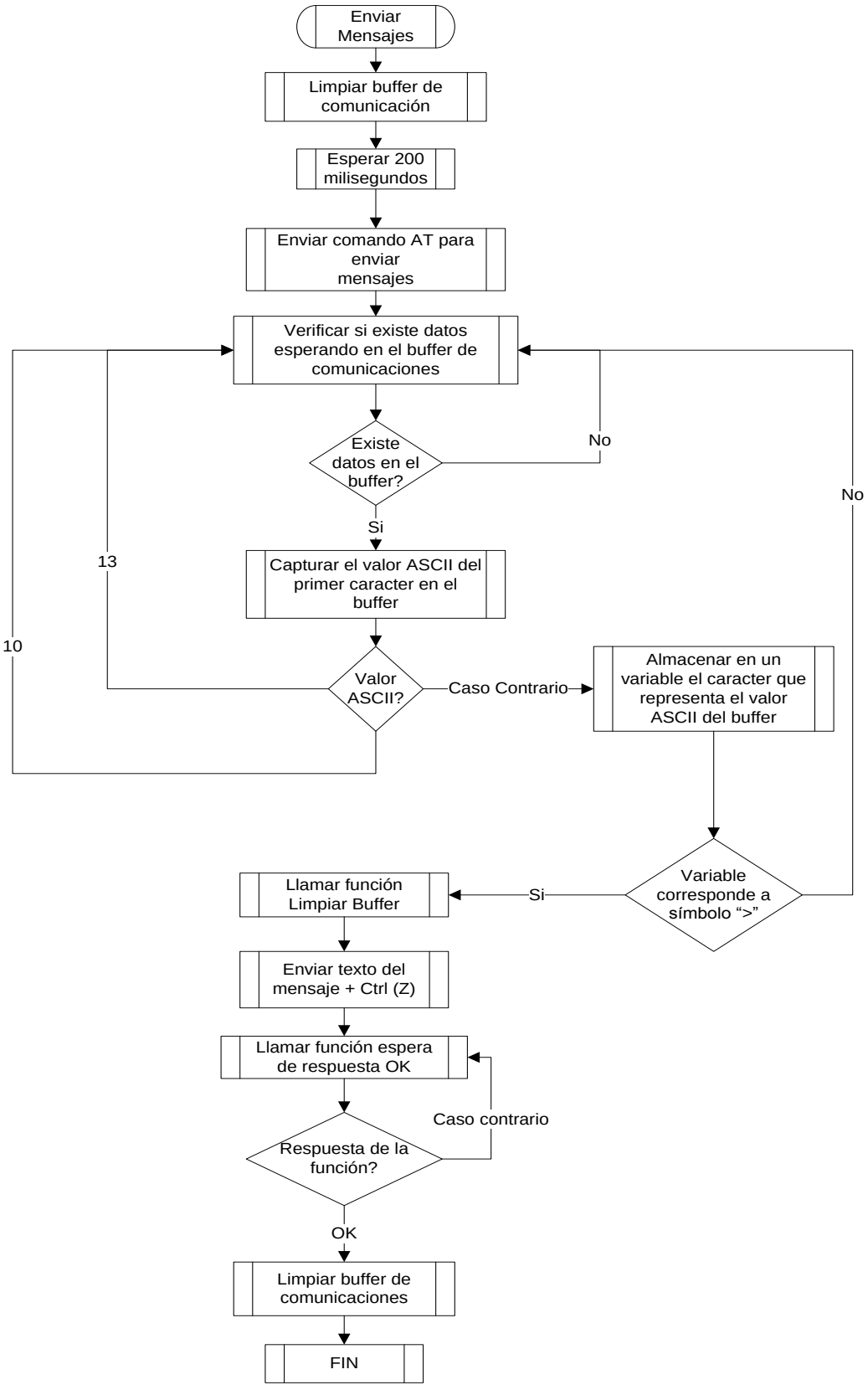


Figura A.7 Función de Envió de Mensajes del Modem GSM

Función de espera de respuesta OK del Modem GSM

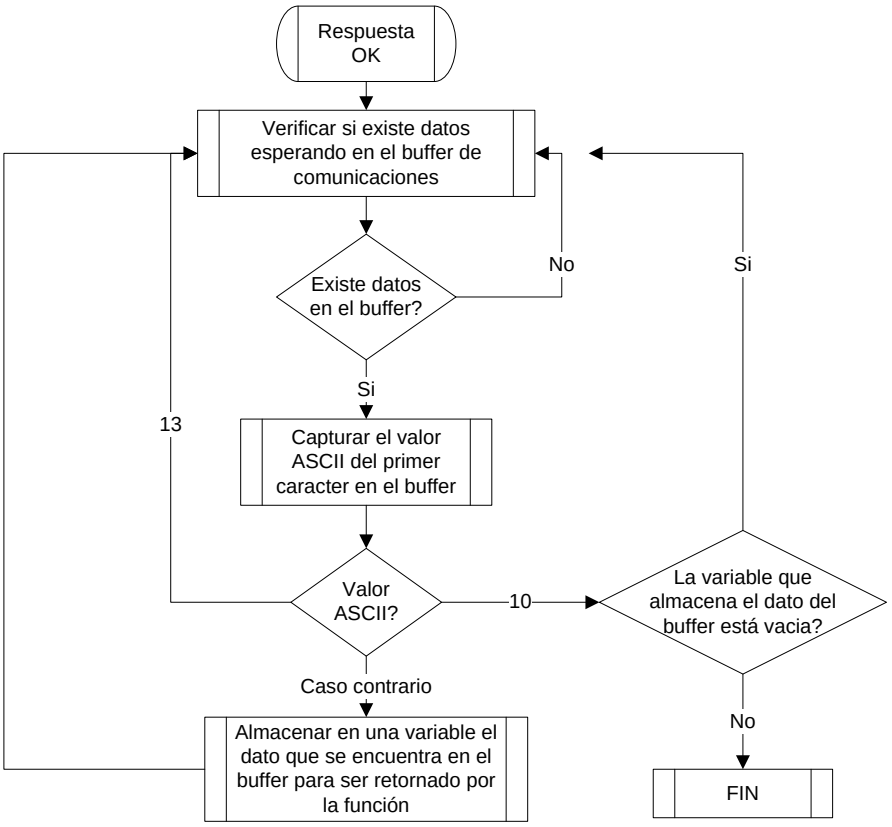


Figura A.8 Función de espera de respuesta OK del Modem GSM

Función Limpiar Buffer del puerto de comunicaciones del Modem GSM

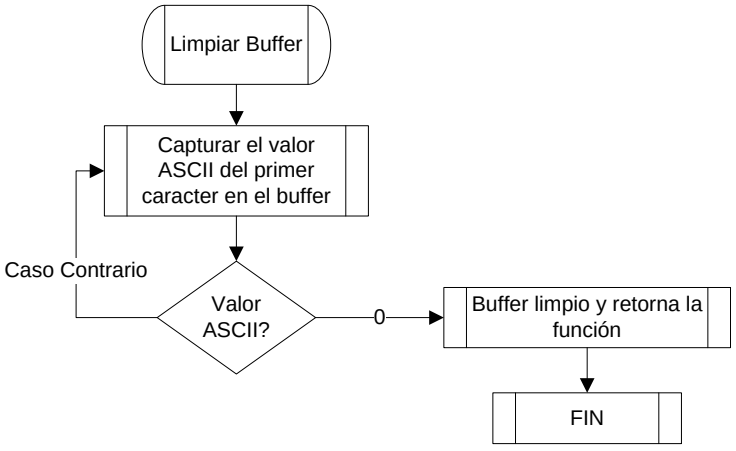


Figura A.9 Función Limpiar Buffer del puerto de comunicaciones del Modem GSM

Función Recibir Mensaje del Modem GSM

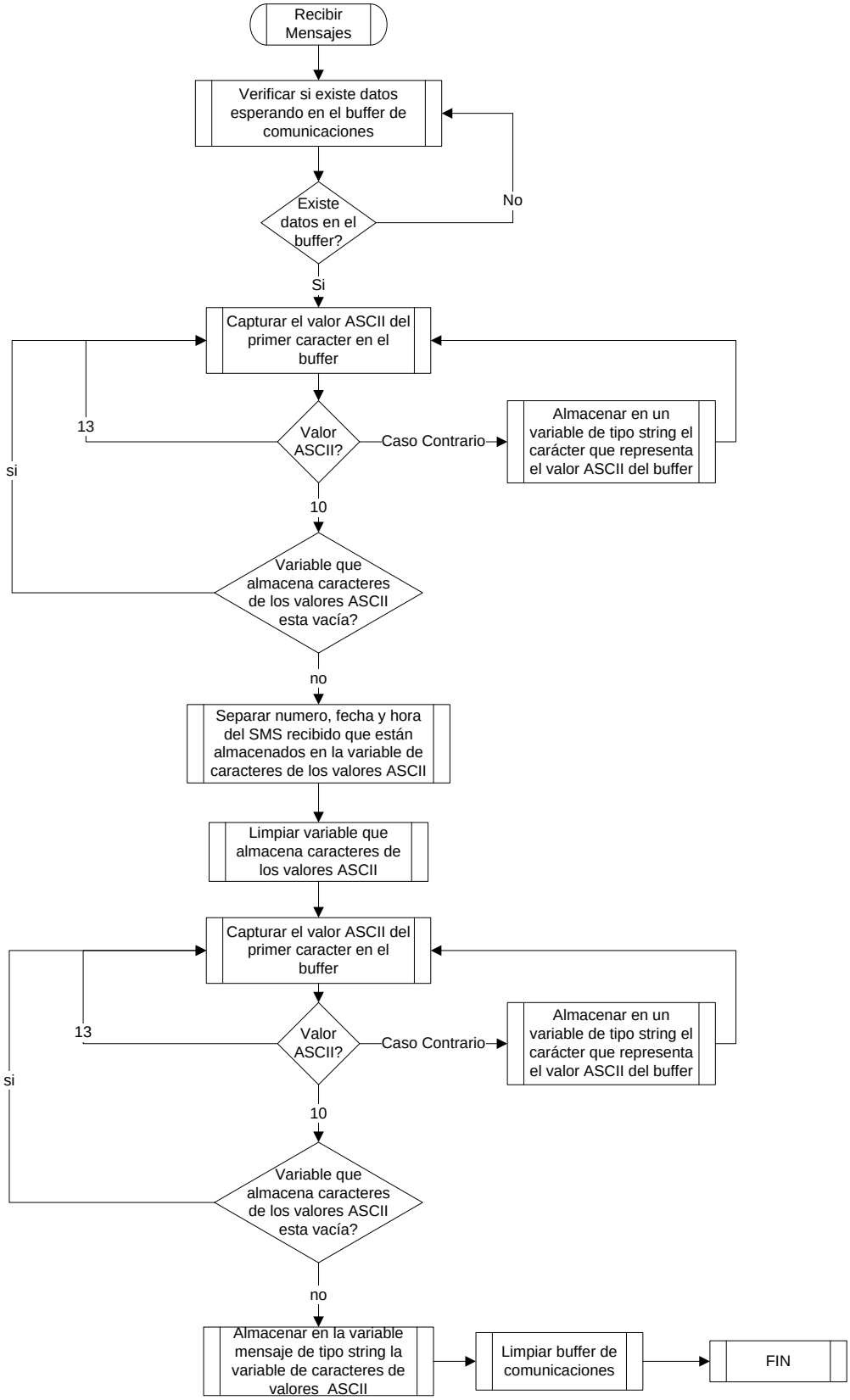


Figura A.10 Función Recibir Mensaje del Modem GSM

Función Validar Mensaje del Modem GSM

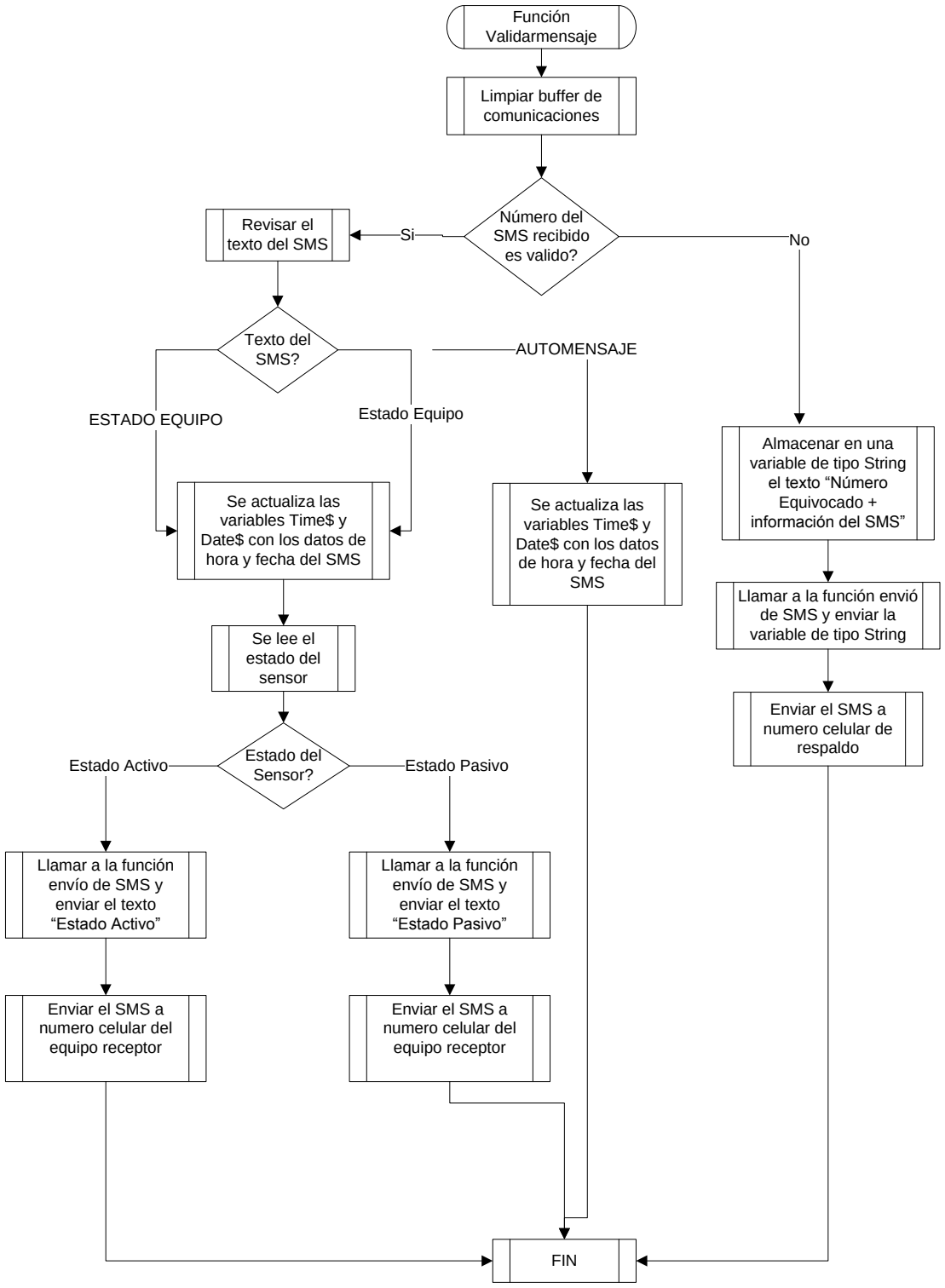


Figura A.11 Función Validar Mensaje del Modem GSM

Función Automensaje

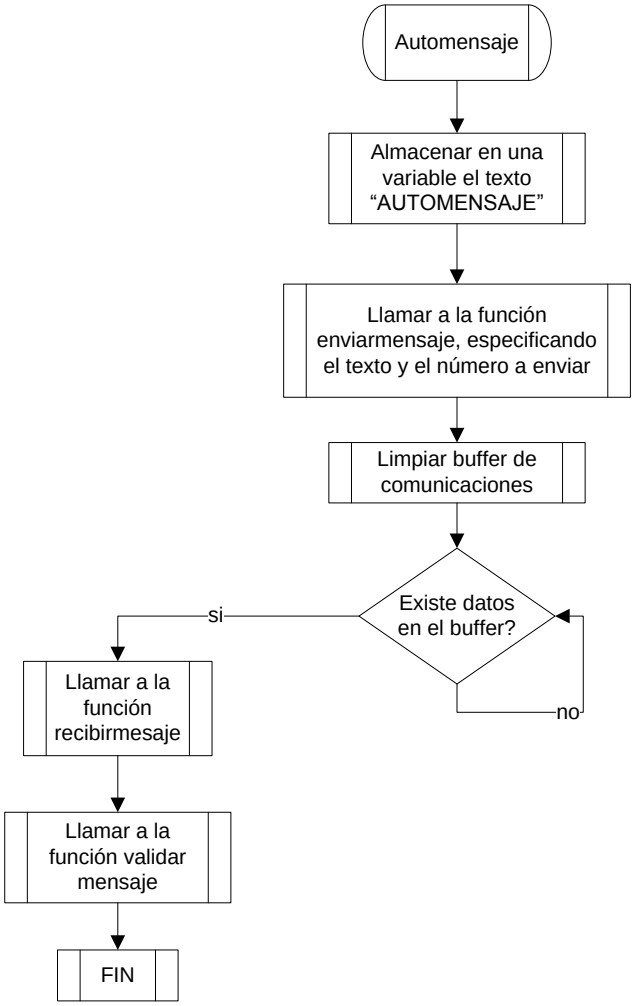


Figura A.12 Función Automensaje

Etapa de Control

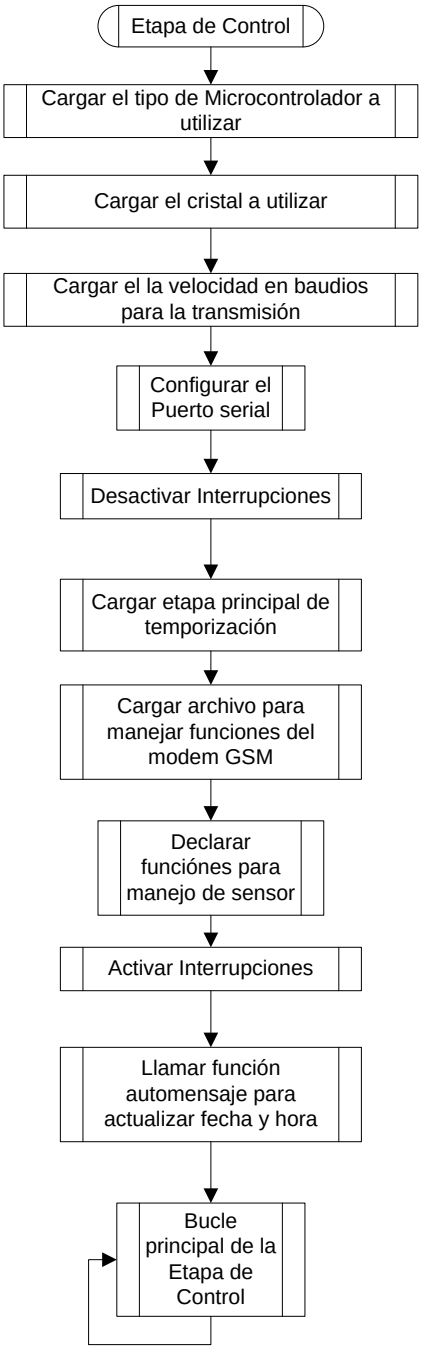


Figura A.13 Etapa de Control

Bucle Principal de la Etapa de Control

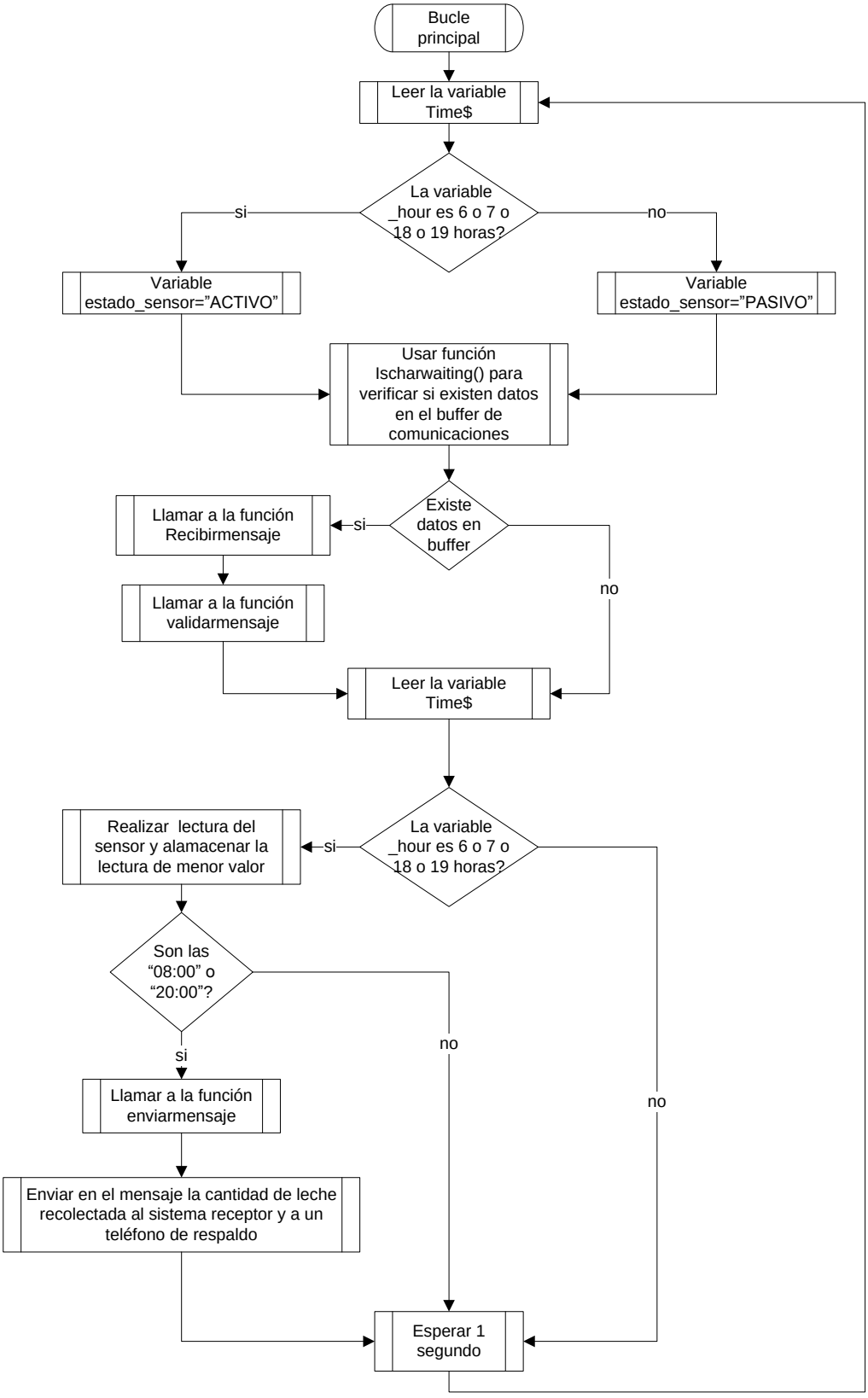


Figura A.14 Bucle Principal de la Etapa de Control

ANEXO A3: DIAGRAMAS DE FLUJO DEL SISTEMA RECEPTOR

Formulario Principal

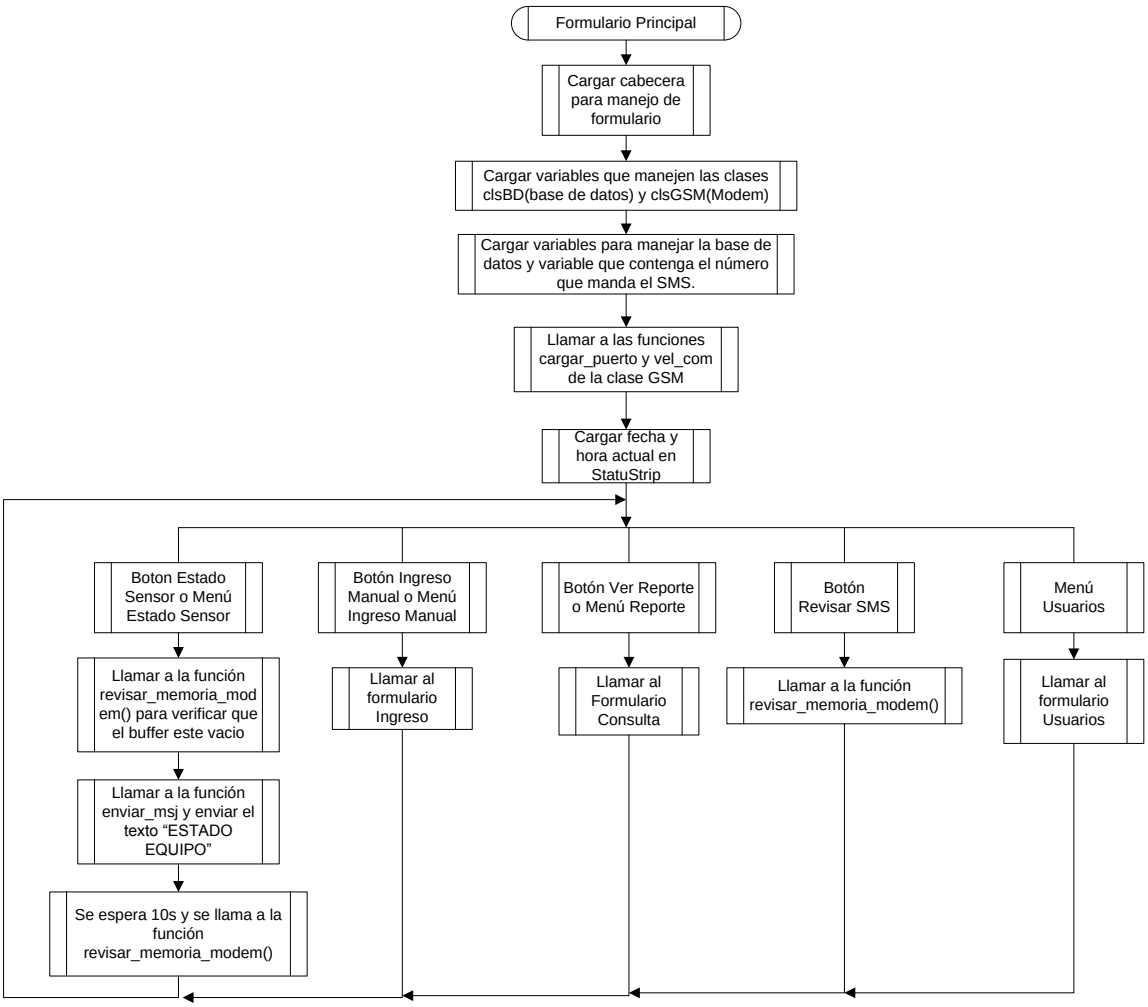


Figura A.15 Formulario Principal

Método Revisar Memoria del Modem

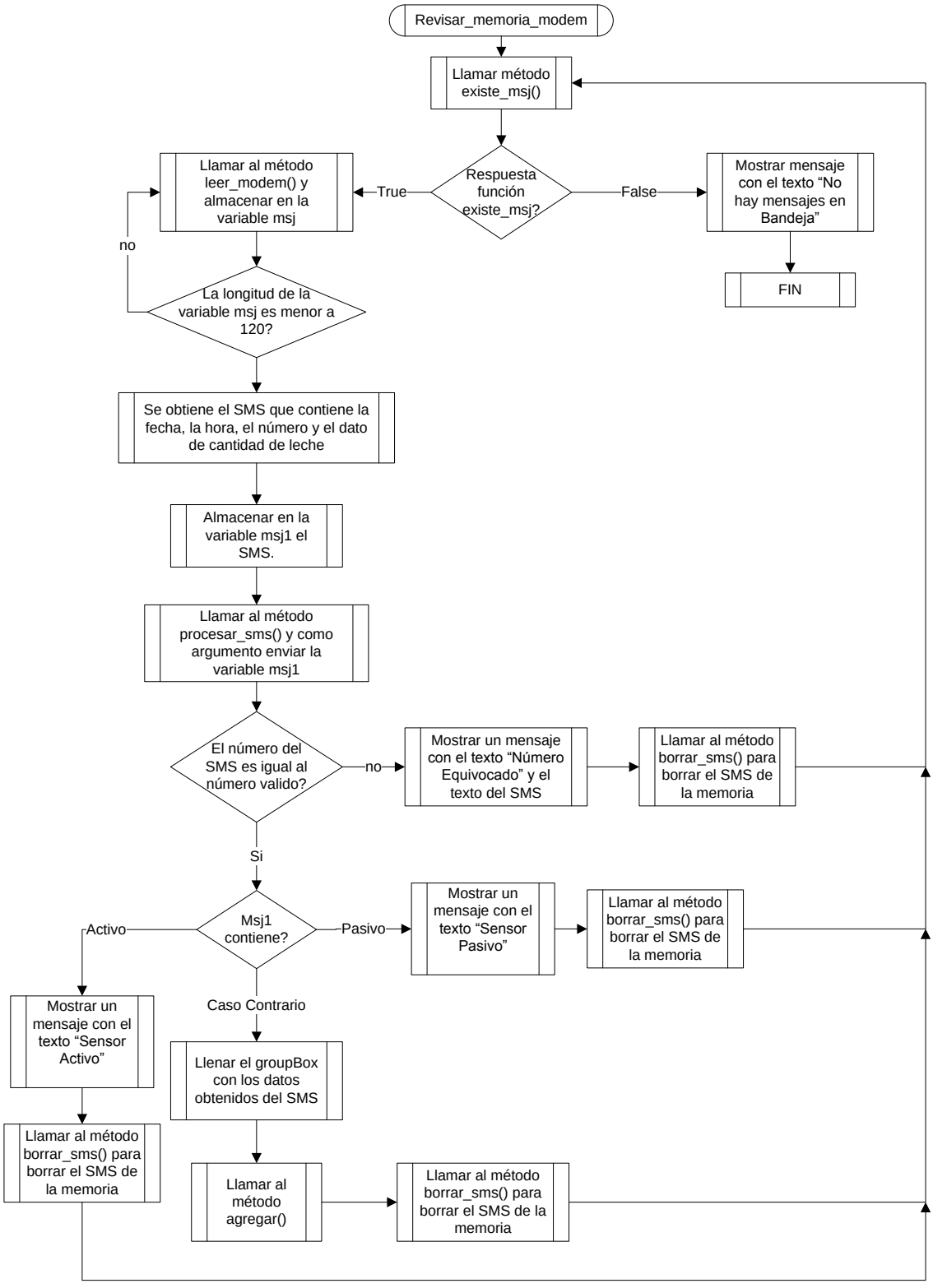


Figura A.16 Método Revisar Memoria del Modem

Formulario Acceso

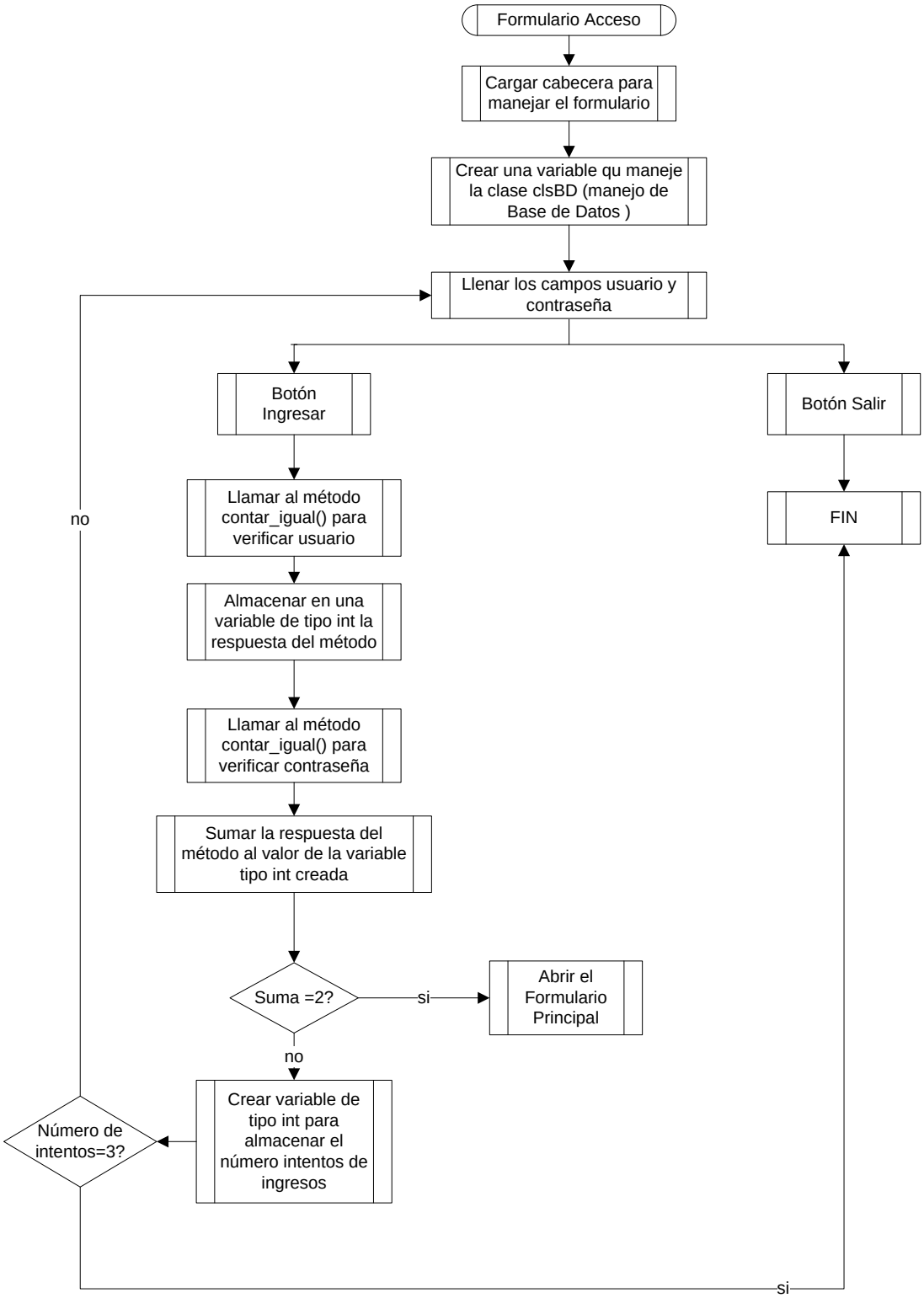


Figura A.17 Formulario Acceso

Formulario Usuarios

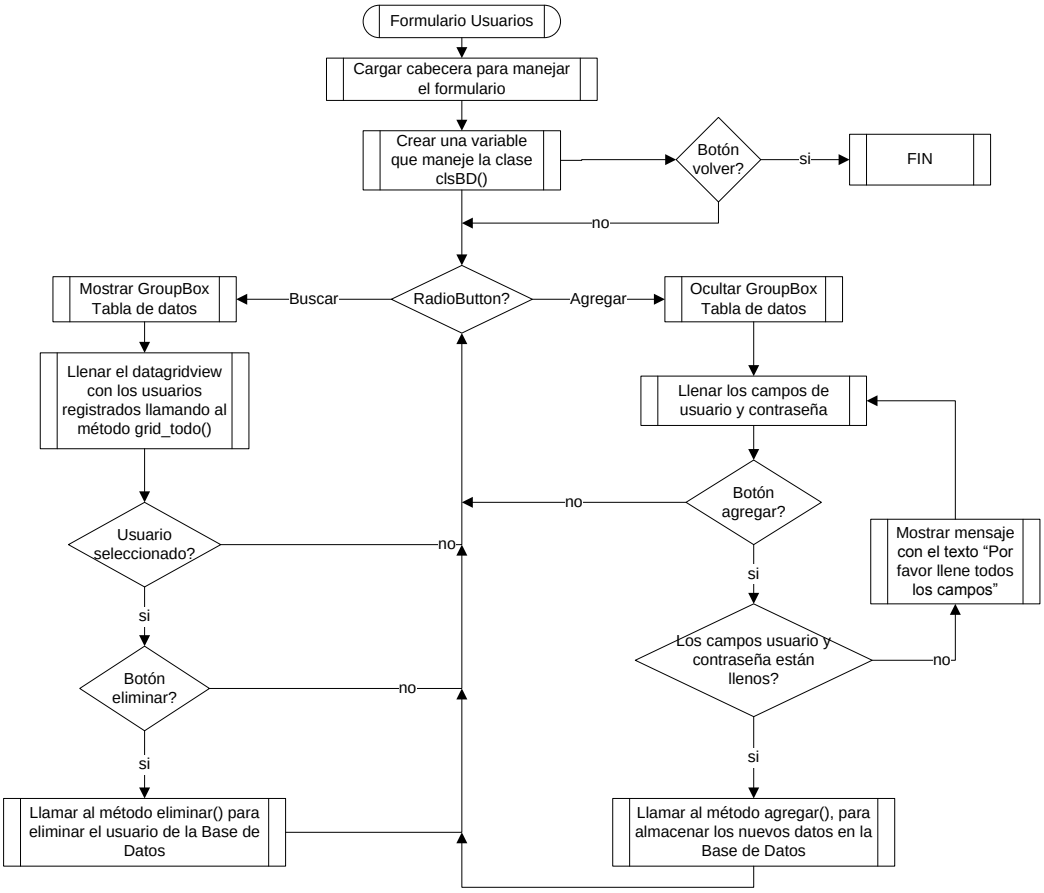


Figura A.18 Diagrama de Flujo del Formulario Usuarios

Formulario Ingreso Manual

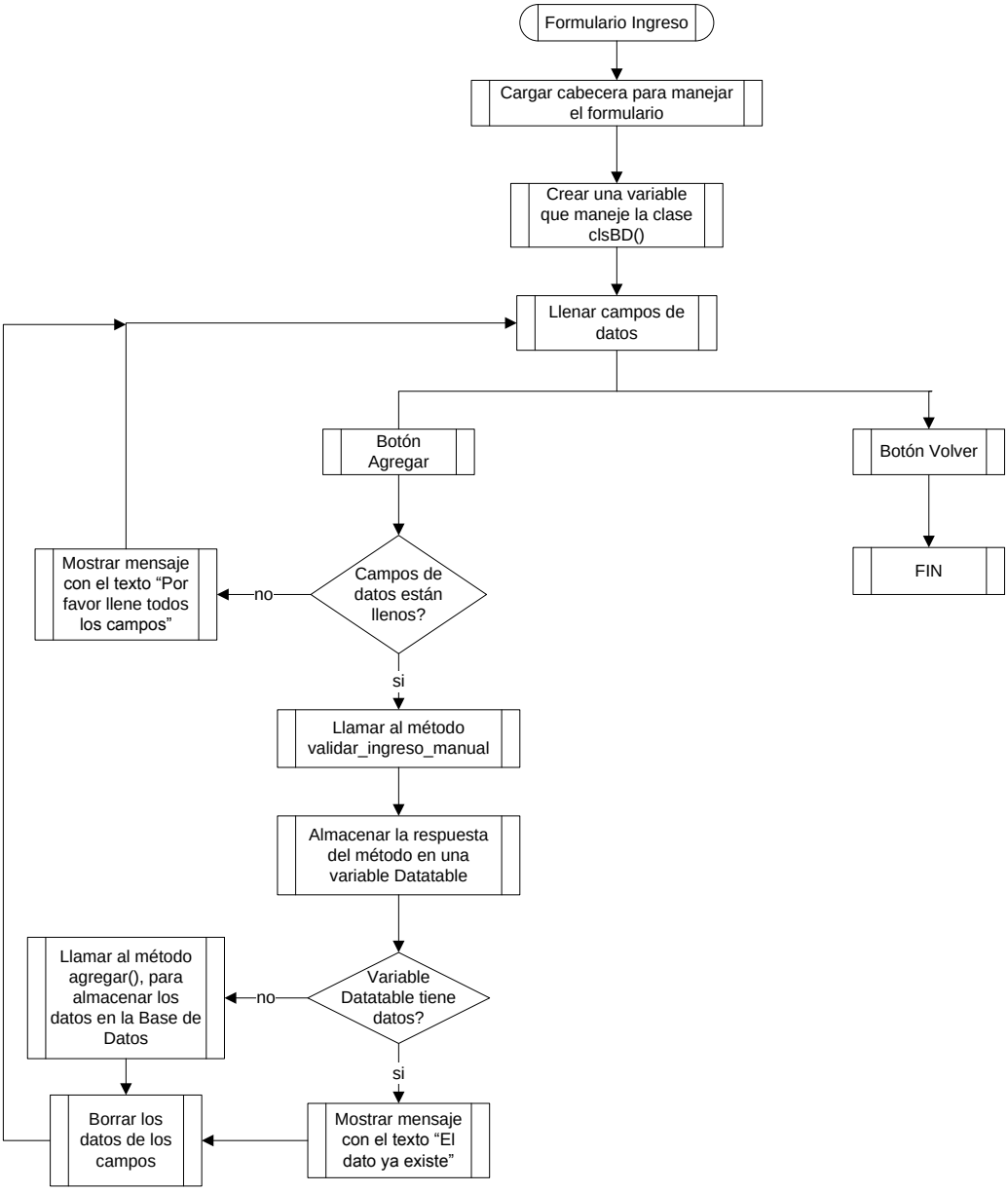


Figura A.19 Formulario Ingreso Manual

Formulario Consulta

Figura A.20 Formulario Consulta

Formulario Reporte

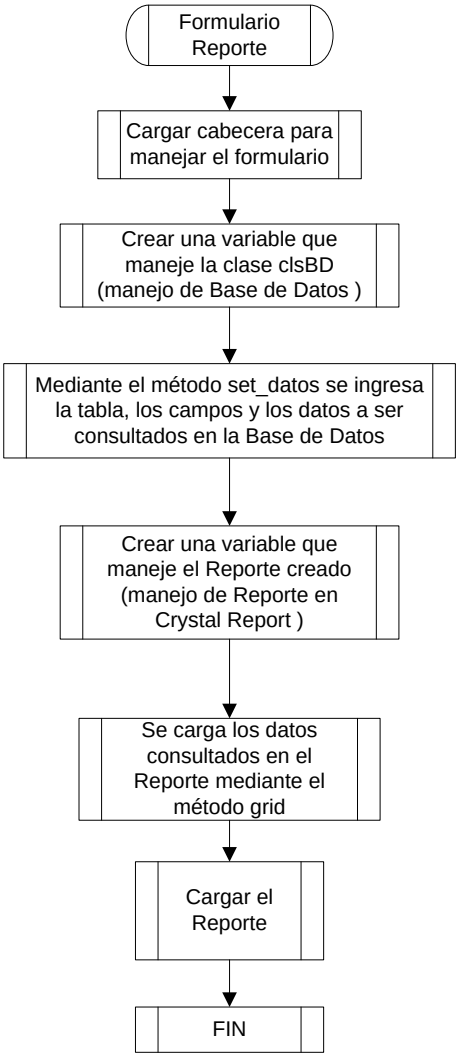


Figura A.21 Formulario Reporte

ANEXO A4: CODIFICACION SISTEMA TRANSMISOR

PROGRAMA PRINCIPAL

```
$regfile = "m8def.dat"
$crystal = 11059200
$baud = 115200

$hwstack = 64
$swstack = 64
$framesize = 64

Open "comb.1:115200,8,n,1,inverted" For Output As #1

Config Serialin = Buffered , Size = 100
Disable Interrupts

$include "SMS_INIT.BAS"

Ddrd.6 = 1 : Portd.6 = 0                    ' congifurar d6 y d7 como salida
Ddrd.7 = 1 : Portd.7 = 0

Led_gsm Alias Portd.6
Led_modem Alias Portd.7

Led_gsm = 0 : Led_modem = 0

                    '
Config Clock = User
Config Date = Ymd , Separator = /
Config Sda = Portc.4
Config Scl = Portc.5

Const Ds1307w = &HD0
Const Ds1307r = &HD1

Dim Weekday As Byte
Dim I As Byte

Print #1 , "Sistema TX"
Print #1 , " GSM "

Waitms 100

Print #1 , "TIME: " ; Time$
Print #1 , "DATE: " ; Date$

Waitms 1500

Declare Sub Iniciar_ultrasonido
Declare Function Lectura_ultrasonido() As Integer
```

```
Config Single = Scientific , Digits = 1

Dim Range As Integer
Dim Dist As Single
Dim Distancia As Integer
Dim Dist_str As String * 5
Dim Time_str As String * 10
Dim Date_str As String * 10
Dim Dist_min As Single
Dim Dist_ini As Single
'Dim Inicio As Bit
Dim Estado_sensor As Bit
Dim X As Bit

Clear Serialin
Enable Interrupts
Reset Led_modem : Reset Led_gsm
Config_inicial
Waitms 500
Print #1 , "TIME: " ; Time$
Print #1 , "DATE: " ; Date$
Automensaje
X = 1

Do

  Print #1 , "TIME: " ; Time$
  Print #1 , "DATE: " ; Date$

  If _hour > 5 And _hour < 8 Or _hour > 17 And _hour < 20 Then

    Estado_sensor = 1
  Else

    Estado_sensor = 0
  End If


  If Ischarwaiting() = 1 Then

    Recibirmensaje
    Validarmensaje , Mensaje

  End If

  If _hour = 6 Or _hour = 7 Or _hour = 22 Or _hour = 23 Then

        If X = 1 Then
```

```
  Iniciar_ultrasonido
  Range = Lectura_ultrasonido()
  Dist_ini = Dist
  X = 0

Else

  Range = Lectura_ultrasonido()
  Dist_min = Dist

  If Dist_ini <> Dist_min Then
     If Dist_min < Dist_ini Then
       Dist_ini = Dist_min              ' dist_ini va a a tener siempre la
                                          distancia minima
     End If
     Print #1 , "Dmin: " ; Dist_ini ; " cm."
  End If
End If


If Time$ = "23:28:00" Or Time$ = "23:30:00" Then

    'calcular el volumen del tanque con el dato del nivel en variable
      volumen

    'v=3.1416*r^2*hmedida
    'hmedida=(htanque+hcolocación)-hsensor


    Print #1 , "fecha:" ; Date$ ; " hora: " ; Time$ ; " nivel: " ; Dist_ini
    Waitms 200
    Mensa = ""
    Dist_str = ""
    Distancia = Dist_ini
    Dist_str = Str(distancia)
    Date_str = ""
    Date_str = Date_str + Str(_day)
    Date_str = Date_str + "/" + Str(_month)
    Date_str = Date_str + "/" + Str(_year)

    Mensa = Dist_str
    Enviarmensaje Mensa , Numero
    Waitms 500
    Mensa = "FECHA: " + Date_str + " HORA: " + Time$ + " NIVEL: " +
      Dist_str
    Enviarmensaje Mensa , Numero_dos
    Waitms 500
    X = 1

End If
```

```
   End If
   Waitms 1000                              'espera 1 s para cada lectura
Loop

End

$include "SMS_END.BAS"


Getdatetime:
  I2cstart
  I2cwbyte Ds1307w
  I2cwbyte 0
  I2cstart
  I2cwbyte Ds1307r
  I2crbyte _sec , Ack
  I2crbyte _min , Ack
  I2crbyte _hour , Ack
  I2crbyte Weekday , Ack
  I2crbyte _day , Ack
  I2crbyte _month , Ack
  I2crbyte _year , Nack
  I2cstop
  _sec = Makedec(_sec) : _min = Makedec(_min) : _hour = Makedec(_hour)
  _day = Makedec(_day) : _month = Makedec(_month) : _year = Makedec(_year)
Return

Setdate:
  _day = Makebcd(_day) : _month = Makebcd(_month) : _year = Makebcd(_year)
  I2cstart
  I2cwbyte Ds1307w
  I2cwbyte 4
  I2cwbyte _day
  I2cwbyte _month
  I2cwbyte _year
  I2cstop
Return

Settime:
  _sec = Makebcd(_sec) : _min = Makebcd(_min) : _hour = Makebcd(_hour)
  I2cstart
  I2cwbyte Ds1307w
  I2cwbyte 0
  I2cwbyte _sec
  I2cwbyte _min
  I2cwbyte _hour
  I2cstop
Return
```

```
Sub Iniciar_ultrasonido
  Config Pinb.0 = Output
  Ddrb.0 = 1 : Portb.0 = 0
End Sub

Function Lectura_ultrasonido() As Integer
  Ddrb.0 = 1 : Portb.0 = 0
  Waitms 50
  Pulseout Portb , 0 , 55                    ' GENERAMOS el pulso DE 5US a
11,059200 MHz PULSO=(Tdeseado*fclock)
  Waitus 700
  Ddrb.0 = 0 : Portb.0 = 0
  Waitus 50
  Pulsein Lectura_ultrasonido , Pinb , 0 , 1
  If Err = 1 Then
    Print #1 , "FUERA DE ALCANCE"
  Else
    'Print #1 , "R: " ; Range ; "   "
    'Dist = Range * 0.06269 -> valor antes de linealizacion
    Dist = Range * 0.1669
    Dist = Dist - 0.5022
  End If
End Function
```

PROGRAMA PARA DECLARACION DE VARIABLES Y FUNCIONES PARA MANEJO DEL MODEM GSM

```
$nocompile

'DECLARACION DE VARIABLES'

Dim B As Byte
Dim K As Byte
Dim Sret As String * 6
Dim Mensa As String * 50

Dim Modem_answer As String * 100

Dim Mensaje As String * 50
Dim Datos_sms_recibido As String * 100
Dim Numero As String * 13
Dim Numero_auto As String * 13
Dim Numero_dos As String * 13
Dim Modem_csq_rssi As String * 4
Dim Modem_csq_ber As String * 4
Dim Numero_sms_recibido As String * 15
```

```
Dim Fecha_sms_recibido As String * 10
Dim Hora_sms_recibido As String * 10

Dim Cont_comillas As Byte

Numero = "+59395530889"
Numero_auto = "+59392754016"
Numero_dos = "+59395530677"

'DECLARACION FUNCIONES'


Declare Sub Config_inicial()                    'FUNCION PARA
CONFIGURACION INICIAL DEL CELULAR
Declare Sub Getok(s As String)                   'FUNCION PARA OBTENER
RESPUESTA OK
Declare Sub Limpiarbuffer()                     'FUNCION PARA LIMPIAR
BUFFER
Declare Sub Enviarmensaje(s As String , N As String)       'FUNCION PARA
ENVIAR MENSAJE
Declare Sub Recibirmensaje()                     'FUNCION PARA RECIBIR
MENSAJE
Declare Sub Validarmensaje(s As String)                'FUNCION PARA VALIDAR
MENSAJE
Declare Sub Automensaje()                       'FUNCION AUTOMENSAJE
```

PROGRAMA PARA DESARROLLO DE FUNCIONES PARA MANEJO DEL MODEM GSM

```
$nocompile

''DESARROLLO DE FUNCIONES'

'FUNCION CONFIGURACION INICIAL'

Sub Config_inicial()
  Print #1 , "MODEM ZTE MG3006"
  Print #1 , "ESPERANDO..."

  Clear Serialin
  Modem_answer = ""
  Do
    If Iscarwaiting() = 1 Then
      Set Led_gsm
      Do
        B = Inkey()
        Select Case B
```

```
            Case 10 : If Modem_answer <> "" Then Exit Do
            Case 13 :
            Case Else:
               Modem_answer = Modem_answer + Chr(b)
          End Select
        Loop
        Exit Do
      End If
    Loop                                    'Until Modem_answer <> ""
    Print #1 , Modem_answer
    Clear Serialin
    Modem_answer = ""
    Do
      If Ischarwaiting() = 1 Then
        Do
          B = Inkey()
          Select Case B
            Case 10 : If Modem_answer <> "" Then Exit Do
            Case 13 :
            Case Else:
               Modem_answer = Modem_answer + Chr(b)
          End Select
        Loop
        Exit Do
      End If
    Loop                                    'Until Modem_answer <> ""
    Print #1 , Modem_answer

    Clear Serialin
    Print #1 , "MODEM LISTO"
    Clear Serialin
    Reset Led_gsm

'   Print #1 , "AT -> OK"
'   Print Chr(13);
    Limpiarbuffer
    For I = 0 To 25
      Toggle Led_modem
      Waitms 25
    Next


    Do
'     Print #1 , "AT -> OK"
'     Toggle Led_gsm
      Print "AT" ; Chr(13);
      Getok Modem_answer
'     Print #1 , Modem_answer
    Loop Until Modem_answer = "OK"
```

```
  Limpiarbuffer
  For I = 0 To 5
    Toggle Led_modem
    Waitms 25
  Next

  Print #1 , "AT -> OK"

  Do
    Print "ATE0" ; Chr(13);                    ' APAGAMOS EL ECO DEL
MODEM
    Getok Modem_answer
  Loop Until Modem_answer = "OK"

  Limpiarbuffer
  For I = 0 To 5
    Toggle Led_modem
    Waitms 25
  Next
  Print #1 , "ATE0 -> OK"


  Do
    Print "AT+IPR=115200" ; Chr(13);           ' COMUNICACION A 115200
BPS
    Getok Modem_answer
  Loop Until Modem_answer = "OK"
  Limpiarbuffer
  For I = 0 To 5
    Toggle Led_modem
    Waitms 25
  Next
  Print #1 , "AT+IPR=115200 -> OK"

  Do
    Print "AT+CMGF=1" ; Chr(13);              ' CONFIGURAMOS PARA
ENVIAR Y RECIBIR TEXTO
    Getok Modem_answer
  Loop Until Modem_answer = "OK"
  Limpiarbuffer
  For I = 0 To 5
    Toggle Led_modem
    Waitms 25
  Next
  Print #1 , "AT+CMGF=1 -> OK"

  Do
    Print "AT+CNMI=3,2,0,0,0" ; Chr(13);      ' CONFIGURAMOS PARA Q
AL RECIBIR MENSAJE Y BORRAR
    Getok Modem_answer
```

```
  Loop Until Modem_answer = "OK"
  Limpiarbuffer
  For I = 0 To 5
    Toggle Led_modem
    Waitms 25
  Next
  Print #1 , "AT+CNMI=3,2,0,0,0 -> OK"


  Do
    Modem_csq_rssi = ""
    Modem_csq_ber = ""
    Do
      Print "AT+CSQ" ; Chr(13);                    ' SEÑAL DEL CELULAR
      Getok Modem_answer
    Loop Until Modem_answer <> ""
    For I = 0 To 5
      Toggle Led_modem
      Waitms 20
    Next
    Modem_csq_rssi = Mid(modem_answer , 7 , 2)
    Modem_csq_ber = Mid(modem_answer , 10 , 2)
    Waitms 100
  Loop Until Modem_csq_rssi <> "" And Modem_csq_ber <> "" And
Modem_csq_rssi <> "99" And Modem_csq_ber <> "99"      'Until Modem_csq_rssi
<> "99" And Modem_csq_ber <> "99"

  Print #1 , "+CSQ: " ; Modem_csq_rssi ; "," ; Modem_csq_ber

  Clear Serialin

  Do
    Print "AT&W" ; Chr(13);                        ' GUARDAR LA CONFIGURACION
LA CONFIGURACION ACTUAL
    Getok Modem_answer
  Loop Until Modem_answer = "OK"
  Limpiarbuffer
  For I = 0 To 5
    Toggle Led_modem
    Waitms 25
  Next
  Print #1 , "AT&W -> OK"

  Limpiarbuffer
  Set Led_modem
  Reset Led_gsm
  Clear Serialin
End Sub
```

```
'FUNCION ESPERA RESPUESTA OK (ESPERAR LA RESPUESTA DEL
CELULAR ->  "OK")'

Sub Getok(s As String)
  S = ""
  Do
    If Iscarwaiting() = 1 Then
      B = Inkey()
      Select Case B
        Case 10 : If S <> "" Then Exit Do
        Case 13 :
        Case Else:
           S = S + Chr(b)
      End Select
    End If
  Loop
End Sub


'FUNCION LIMPIAR BUFFER (LIMPIAR BUFFER DE COMUNICACION ENTRE
MICRO Y CELULAR)'

Sub Limpiarbuffer()
  Do
    B = Inkey()
  Loop Until B = 0
End Sub


'FUNCION ENVIAR MENSAJE'

Sub Enviarmensaje(s As String , N As String)
   Clear Serialin
   Print #1 , "AT+CMGS=" ; Chr(34) ; N ; Chr(34) ; Chr(13);
   Waitms 200
   Do
     Print "AT+CMGS=" ; Chr(34) ; N ; Chr(34) ; Chr(13);
     Sret = ""
     Do
       If Iscarwaiting() = 1 Then
         B = Inkey()
         Select Case B
           Case 13 :
           Case 10 :
           Case Else
             Sret = Sret + Chr(b)
         End Select
       End If
     Loop Until Sret = "> "
```

```
   Loop Until Sret = "> "
                        'Until B = 62                              'Until Sret = "> "
   'Clear Serialin
   Limpiarbuffer
   Print S ; Chr(26);
   Do
     Getok Sret
   Loop Until Sret = "OK"
'   Print #1 , "OK"
'   Clear Serialin
   Print #1 , "MENSAJE ENVIADO"
   Clear Serialin
   Mensa = ""
End Sub

'FUNCION RECIBIR MENSAJE'


Sub Recibirmensaje()
     Modem_answer = ""
     Do
       B = Inkey()
       Select Case B
         Case 10 : If Modem_answer <> "" Then Exit Do
         Case 13:
         Case Else
           Modem_answer = Modem_answer + Chr(b)
       End Select
     Loop
     Datos_sms_recibido = Modem_answer
     Numero_sms_recibido = Mid(datos_sms_recibido , 8 , 12)
     Fecha_sms_recibido = Mid(datos_sms_recibido , 24 , 8)
     Hora_sms_recibido = Mid(datos_sms_recibido , 33 , 8)
     Modem_answer = ""
     Do
       B = Inkey()
       Select Case B
         Case 10 : If Modem_answer <> "" Then Exit Do
         Case 13:
         Case Else
           Modem_answer = Modem_answer + Chr(b)
       End Select
     Loop
     Mensaje = Modem_answer
     Clear Serialin

'      Print #1 , Datos_sms_recibido
'      Print #1 , Mensaje
'      Print #1 , Numero_sms_recibido
'      Print #1 , Fecha_sms_recibido
```

```
'      Print #1 , Hora_sms_recibido
End Sub


'FUNCION VALIDAR MENSAJE'


Sub Validarmensaje(s As String)
     Clear Serialin
     Mensa = ""
     If Numero_sms_recibido = Numero Or Numero_sms_recibido = Numero_auto
Then
        Print #1 , "NUMERO CORRECTO"
        Select Case S
          Case "ESTADO EQUIPO":
            Print #1 , Fecha_sms_recibido
            Print #1 , Hora_sms_recibido
            Time$ = Hora_sms_recibido
            Date$ = Fecha_sms_recibido

            If Estado_sensor = 1 Then
              Mensa = "ACTIVO"
            Else
              Mensa = "PASIVO"
            End If
            Print #1 , "MENSA: " ; Mensa
            Enviarmensaje Mensa , Numero
          Case "Estado equipo":
            Print #1 , Fecha_sms_recibido
            Print #1 , Hora_sms_recibido
            Time$ = Hora_sms_recibido
            Date$ = Fecha_sms_recibido

            If Estado_sensor = 1 Then
              Mensa = "ACTIVO"
            Else
              Mensa = "PASIVO"
            End If
            Print #1 , "MENSA: " ; Mensa
            Enviarmensaje Mensa , Numero
          Case "AUTOMENSAJE"
             Print #1 , "AUTOMENSAJE RECIBIDO"
             Time$ = Hora_sms_recibido
             Date$ = Fecha_sms_recibido


        End Select
      Else
        Print #1 , "NUMERO DESCONOCIDO"
```

```
        Mensa = "NUMERO DESCONOCIDO - MENSAJE: " + S + "  NUMERO: " +
Numero_sms_recibido
        Enviarmensaje Mensa , Numero_dos
      End If

End Sub


'FUNCION AUTOMENSAJE'


Sub Automensaje()

   Mensa = "AUTOMENSAJE"
   Enviarmensaje Mensa , Numero_auto
   Clear Serialin

   Do
     Print #1 , "TIME: " ; Time$
     Print #1 , "DATE: " ; Date$
     Waitms 500
   Loop Until Iscarwaiting() = 1


   Recibirmensaje
   Print #1 , "mensaje recibido"

   Validarmensaje , Mensaje

End Sub
```

ANEXO A5: CODIFICACION SISTEMA RECEPTOR

Formulario Principal

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;

namespace Control_Base_de_Datos
{
	/// <summary>
	/// Description of MainForm.
	/// </summary>
	public partial class MainForm : Form
	{
		clsBD BD = new clsBD();
		clsGSM GSM = new clsGSM();

		string tabla = "Datos";
		string []campos = new string[]{"Fecha", "Hora", "Cantidad"};
		string []datos = new string[3];

	//string num_sensor = "95530677";
	string num_sensor = "92754016";
	int cont_timer2;


		public MainForm()
		{
			//
			// The InitializeComponent() call is required for Windows
			Forms designer support.
			//
			InitializeComponent();

			//
			// TODO: Add constructor code after the InitializeComponent()
			call.
			//
		}

	void limpiar_textos()
	{
	   foreach (Control c in groupBox1.Controls)
	   {
	      if (c is TextBox)
	      {
	         c.Text = "";
	      }
	   }
```

```
 }

 void MainFormLoad(object sender, EventArgs e)
{
        //serial.cargar_puerto(serial.get_puertos()[0]);
        GSM.cargar_puerto("COM7");
        GSM.vel_com(115200);
        timer1.Start();
        mnuFechaHora.Text = "" + DateTime.Now.ToString();
        revisar_memoria_modem();
 }



 void Timer1Tick(object sender, EventArgs e)
{
            mnuFechaHora.Text = "" + DateTime.Now;

 }

 void MnuReporteClick(object sender, EventArgs e)
 {
        ver_Reporte();
 }

 void ver_Reporte()
 {
         frmConsultar Consultar = new frmConsultar();
        Consultar.ShowDialog();
 }

  private void btnEstado_Click(object sender, EventArgs e)
  {
        estado_sensor();
  }

  private void btnIngresoManual_Click(object sender, EventArgs e)
  {
     ingreso_manual();
  }

  private void btnReporte_Click(object sender, EventArgs e)
  {
     ver_Reporte();
  }

  private void mnuUsuarios_Click(object sender, EventArgs e)
  {
     frmUsuarios usuarios = new frmUsuarios();
     usuarios.ShowDialog();
```

```
}

private void MainForm_FormClosing(object sender, FormClosingEventArgs
e)
{
   frmAcceso Acceso = new frmAcceso();
   this.Hide();
   Acceso.Show();
   //Application.Exit();
}

private void btnRevisarMemoria_Click(object sender, EventArgs e)
{
   revisar_memoria_modem();
}

private void button2_Click(object sender, EventArgs e)
{
   //GSM.enviar_msj("095530889", "300");
   //GSM.enviar_msj("095530889", "400");
   //GSM.enviar_msj("095530889", "500");
}

void ingreso_manual()
{
   frmIngreso Ingreso = new frmIngreso();
   Ingreso.ShowDialog();
}

void procesar_datos(string sms)
{
   datos = GSM.procesar_sms(sms);
   if (datos[3]=="0" + num_sensor)
   {
      if (!sms.Contains("ACTIVO") && !sms.Contains("PASIVO"))
      {
         try
         {
            txtAño.Text = "20" + datos[0].Substring(0, 2).Trim();
            txtMes.Text = datos[0].Substring(3, 2).Trim();
            txtDia.Text = datos[0].Substring(6, 2).Trim();

            txtHora.Text = datos[1].Substring(0, 2).Trim();
            txtMin.Text = datos[1].Substring(3, 2).Trim();
            txtSeg.Text = datos[1].Substring(6, 2).Trim();
            datos[1] = datos[1].Substring(0, 2).Trim();
            txtDato.Text = datos[2].Trim();
         }
         catch (Exception ex)
         {
```

```
                //MessageBox.Show(ex.Message);
            }
        }
    }
}

void guardar()
{
    datos[0] = txtDia.Text + "/" + txtMes.Text + "/" + txtAño.Text;
    datos[1] = txtHora.Text;
    datos[2] = txtDato.Text;
    BD.agregar(tabla, campos, datos);
}

void revisar_memoria_modem()
{
    limpiar_textos();

    int cont = 3;
        while (GSM.existe_msj())
        {
            string msj = "";
            string msj1 = "";
           do
           {

               msj += GSM.leer_modem().Trim();
               //MessageBox.Show(msj);
               //MessageBox.Show(Convert.ToString(msj.Length));
           }
           while (msj.Length < 120);
           //MessageBox.Show(msj);
            //MessageBox.Show(msj);
            int cont1 = msj.IndexOf("+CMGL");
            int cont2 = msj.LastIndexOf("+CMGL");
            //MessageBox.Show(Convert.ToString(cont1));
            //MessageBox.Show(Convert.ToString(cont2));

            msj1 = msj.Substring(cont1, cont2);
            //MessageBox.Show(msj1);
            procesar_datos(msj1);
            //MessageBox.Show(datos[3]);
            if (datos[3]=="0" + num_sensor)//(datos[3].Contains(num_sensor))
            {
               //MessageBox.Show("Número correcto: " + datos[3]);

               if (msj1.Contains("ACTIVO")) MessageBox.Show("Sensor Activo");
               if (msj1.Contains("PASIVO")) MessageBox.Show("Sensor Pasivo");
```

```
                if (!msj1.Contains("ACTIVO") && !msj1.Contains("PASIVO"))
                guardar();

                GSM.borrar_sms(cont);
                cont++;

            }
            else
            {
                MessageBox.Show("Número equivocado: " + datos[3] + "Mensaje:
                " + datos [2] );

                //if (msj.Contains("ACTIVO")) MessageBox.Show("Sensor Activo");
                //if (msj.Contains("PASIVO")) MessageBox.Show("Sensor Pasivo");
                GSM.borrar_sms(cont);
                cont++;
            }
        }
        if (!GSM.existe_msj())
        {
            MessageBox.Show("No hay mensajes en bandeja");
            GSM.borrar_sms(3);
            GSM.borrar_sms(4);
        }
    }

    void estado_sensor()
    {
        revisar_memoria_modem();
        if (MessageBox.Show("¿Desea enviar 1 mensaje de estado al Sensor?",
"Estado Sensor", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            //revisar_memoria_modem();

            cont_timer2 = 0;

            MessageBox.Show(GSM.enviar_msj("0" + num_sensor, "ESTADO
            EQUIPO"));
            //string msj = GSM.leer();
            //MessageBox.Show(msj);
            //revisar_memoria_modem();
            blockear_pantalla();
            timer2.Start();
        }
    }

    private void mnuIngresoManual_Click(object sender, EventArgs e)
    {
        ingreso_manual();
    }
```

```
        private void mnuEstadoSensor_Click(object sender, EventArgs e)
        {
            estado_sensor();
        }

        private void timer2_Tick(object sender, EventArgs e)
        {
            cont_timer2++;
            if (cont_timer2 > 10)
            {
                timer2.Stop();
                revisar_memoria_modem();
                this.Enabled = true;

            }
        }
```

Formulario Acceso

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace Control_Base_de_Datos
{
    public partial class frmAcceso : Form
    {
        clsBD BD = new clsBD();
        MainForm  Principal = new MainForm();
        int cont = 0;

        public frmAcceso()
        {
            InitializeComponent();
        }

        private void btnSalir_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void btnIngresar_Click(object sender, EventArgs e)
        {
```

```
            validar();
        }

        void validar()
        {
            int i = 0;
            i = BD.contar_igual("Usuarios", "Usuario", txtUsuario.Text);
            i += BD.contar_igual("Usuarios", "Contraseña", txtContraseña.Text);
            if (i == 2)
            {
                Principal.Show();
                this.Hide();
            }
            else
            {
                cont++;
                MessageBox.Show("Usuario no registrado");
            }
            if (cont == 3) Application.Exit();
        }

        private void frmAcceso_FormClosing(object sender, FormClosingEventArgs
        e)
        {
            Application.Exit();
        }

        private void frmAcceso_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Formulario Ingreso

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace Control_Base_de_Datos
{
    public partial class frmIngreso : Form
    {
        string tabla = "Datos";
```

```
string[] campos = new string[] { "Fecha", "Hora", "Cantidad" };
string[] datos = new string[3];
clsBD BD = new clsBD();


public frmIngreso()
{
    InitializeComponent();
}

private void button1_Click(object sender, EventArgs e)
{
    string msj="";
    DataTable dt = new DataTable();
    datos[0] = txtDia.Text + "/" + txtMes.Text + "/" + txtAño.Text;
    datos[1] = txtHora.Text;
    datos[2] = txtDato.Text;
    if (datos[0].Length > 0 && datos[1].Length > 0 && datos[2].Length > 0)
    {
        dt = BD.validar_ingreso_manual(tabla, campos, datos);

        //MessageBox.Show(msj);

        if (dt.Rows.Count > 0)
        {
            MessageBox.Show("El dato ya existe");
            limpiar_textos();
        }
        else
        {
            BD.agregar(tabla, campos, datos);
            limpiar_textos();
        }

    }
    else
    {
        MessageBox.Show("Por favor llene todos los campos");

    }

    //guardar();
}

/*void guardar()
{
    datos[0] = txtDia.Text + "/" + txtMes.Text + "/" + txtAño.Text;
    datos[1] = txtHora.Text;
    datos[2] = txtDato.Text;
    BD.agregar(tabla, campos, datos);
```

```
        }*/

        private void button2_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void frmIngreso_Load(object sender, EventArgs e)
        {

        }

        void limpiar_textos()
        {
            foreach (Control c in groupBox1.Controls)
            {
                if (c is TextBox)
                {
                    c.Text = "";
                }
            }
        }
    }
}
```

Formulario Consulta

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Diagnostics;

namespace Control_Base_de_Datos
{
        /// <summary>
        /// Description of frmConsultar.
        /// </summary>
        public partial class frmConsultar : Form
        {
                clsBD BD = new clsBD();
                frmReporte reporteCrystal = new frmReporte();

                string []campos = new string[]{"Fecha", "Hora", "Cantidad"};
                string []datos =  new string[4];
                string tabla = "Datos";

                public frmConsultar()
                {
```

```
            //
            // The InitializeComponent() call is required for Windows
            Forms designer support.
            //
            InitializeComponent();

            //
            // TODO: Add constructor code after the InitializeComponent()
            call.
            //
        }

        void BtnReporteClick(object sender, EventArgs e)
        {
            datos[2] = txtHora.Text;
            datos[3] = txtHora2.Text;
            reporteCrystal.set_datos(tabla, campos, datos);
            reporteCrystal.WindowState = FormWindowState.Maximized;
            reporteCrystal.ShowDialog();
        }

        void McDesdeDateSelected(object sender, DateRangeEventArgs e)
        {
            datos[0] = mcDesde.SelectionStart.ToString();
        }

        void McHastaDateSelected(object sender, DateRangeEventArgs e)
        {
            datos[1] = mcHasta.SelectionStart.ToString();
        }
    }
}
```

Formulario Usuarios

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace Control_Base_de_Datos
{
    public partial class frmUsuarios : Form
    {
        clsBD BD = new clsBD();
        string tabla = "Usuarios";
        string[] campos = new string[] { "Usuario", "Contraseña" };
```

```
string[] datos = new string[2];

public frmUsuarios()
{
    InitializeComponent();
}

private void rbBuscar_CheckedChanged(object sender, EventArgs e)
{
    txtUsuario.Text = "";
    txtContraseña.Text = "";
    btnAccion.Text = "Eliminar";
    this.Size = new Size(310, 356);
    gbTabla.Visible = true;
    dgDatos.DataSource = BD.grid_todo(tabla);
}

private void rbAgregar_CheckedChanged(object sender, EventArgs e)
{
    txtUsuario.Text = "";
    txtContraseña.Text = "";
    btnAccion.Text = "Agregar";
    this.Size = new Size(310, 150);
    gbTabla.Visible = false;
}

private void btnAccion_Click(object sender, EventArgs e)
{
    if (btnAccion.Text == "Eliminar")
    {
        if (MessageBox.Show("¿Desea borrar permanentemente \"" +
        txtUsuario.Text.Trim() + "\"?", "Borrar Usuario",
        MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            BD.eliminar(tabla, "Usuario", txtUsuario.Text);
        }
    }
    if (btnAccion.Text == "Agregar")
    {
        datos[0] = txtUsuario.Text;
        datos[1] = txtContraseña.Text;
        if (datos[0].Length > 0 && datos[1].Length > 0) BD.agregar(tabla,
        campos, datos);
        else MessageBox.Show("Por favor llene todos los campos");
    }
}

private void btnVolver_Click(object sender, EventArgs e)
{
    this.Close();
```

```
        }


        private void frmUsuarios_Load(object sender, EventArgs e)
        {
            this.Size = new Size(310, 150);
            gbTabla.Visible = false;
        }

        private void dgDatos_RowEnter(object sender, DataGridViewCellEventArgs
e)
        {
            txtUsuario.Text = dgDatos.Rows[e.RowIndex].Cells[0].Value.ToString();
            txtContraseña.Text =
dgDatos.Rows[e.RowIndex].Cells[1].Value.ToString();
        }


    }
}
```

Formulario Reporte

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace Control_Base_de_Datos
{
    public partial class frmReporte : Form
    {
        clsBD BD = new clsBD();
        public string _tabla;
        public string[] _campos;
        public string[] _datos;

        public frmReporte()
        {
            InitializeComponent();
        }

        public void set_datos(string tabla, string[] campos, string [] datos)
        {
            _tabla = tabla;
            _campos = campos;
```

```
            _datos = datos;
        }

        void reporte()
        {
            CrystalReport1 rel = new CrystalReport1();
            rel.SetDataSource(BD.grid(_tabla, _campos, _datos));
            crystalReportViewer1.ReportSource = rel;
        }

        private void frmReporte_Load(object sender, EventArgs e)
        {
            reporte();
        }
    }
}
```

Clase GSM

```
using System;
using System.IO;
using System.IO.Ports;
using System.Windows.Forms;

namespace Control_Base_de_Datos
{
        /// <summary>
        /// Description of clsSerial.
        /// </summary>
        public class clsGSM
        {
        SerialPort puerto = new SerialPort();
        char c_z = '\x001a';
        //int c_z = 26;

        public clsGSM()
        {
        }

        //SETEO Y CONFIGURACION DE PUERTO SERIAL
        public string[] get_puertos()
        {
                string[] puertos = SerialPort.GetPortNames();
                return puertos;
        }

        public void vel_com(int vel)
        {
```

```
    puerto.BaudRate = vel;
}

        public void cargar_puerto(string com)
        {
            puerto.PortName = com;
        }

        public void abrir_puerto()
        {
    try
    {
       puerto.Open();
    }
    catch(Exception ex)
    {
       MessageBox.Show(ex.Message);
    }
        }

        public void cerrar_puerto()
        {
            puerto.Close();
        }

//METODOS DE COMUNICACION
  public string leer()
  {
        string msj = "";
        try
  {
   abrir_puerto();
        //for (int i = 0; i < 1; i++)
        //{
            //msj += puerto.ReadLine();
            msj += puerto.ReadExisting();
            //MessageBox.Show(msj);
        //}
            cerrar_puerto();
            }
            catch(Exception ex)
            {
                msj = ex.Message;
            }
    return msj;
        }

private string leer2()
{
    string msj = "";
```

```
        try
        {
            abrir_puerto();
            //for (int i = 0; i < 1; i++)
            //{
                msj += puerto.ReadExisting();
                //MessageBox.Show(msj);
            //}
            //msj += puerto.ReadLine();
            cerrar_puerto();
        }
        catch (Exception ex)
        {
            msj = ex.Message;
        }
        return msj;
    }

    //REVISIÓN AUTOMÁTICA DE MENSAJES
    public string leer_modem()
    {
        string msj = "";
        try
        {
            escribir("AT+CNMI=3,1,0,0,0");
            escribir("AT+CMGL=\"ALL\"");
            msj = leer();
            //MessageBox.Show(msj);
        }
        catch(Exception ex)
        {
            msj = ex.Message;
        }
        return msj;
    }

    public bool existe_msj()
    {
        string msj = "";
        bool existe = false;
        try
        {
            escribir("AT+CNMI=3,1,0,0,0");
            escribir("AT+CMGL=\"ALL\"");
            msj = leer2();
            //MessageBox.Show(msj);
            if (msj.Length > 0) existe = true;
        }
        catch (Exception ex)
        {
```

```
            msj = ex.Message;
        }
        return existe;
    }

    public string escribir(string texto)
    {
        string msj = "";
        try
        {
            abrir_puerto();
            puerto.Write(texto + "\r");
            cerrar_puerto();
        }
        catch (Exception ex)
        {
            msj = ex.Message;
        }
        return msj;
    }

    //ENVIAR MENSAJE
    public string enviar_msj(string numero, string texto)
    {
        string msj = "";
        try
        {
            escribir("AT+CMGF=1");
            escribir("AT+CMGS=\"" + numero + "\"");
            escribir(texto);
            escribir(c_z + " \r");
            msj = "Petición de estado enviada a " + numero;
        }
        catch (Exception ex)
        {
            msj = ex.Message;
        }
        return msj;
    }

    //ENVIAR PETICIÓN DE CONFIRMACIÓN
            public string leer_sms()
            {
                    string sms = "";
        try
        {
            //mensaje tipo texto
            escribir("AT+CMGF=1");
            //comando leer sms
            escribir("AT+CNMI= 3,2,0,0,0");
```

```
            sms = leer();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
                    return sms;
            }

    //SEPARAR MENSAJE EN FECHA, HORA, DATOS Y NUMERO
TELEFÓNICO
public string[] procesar_sms(string sms)
{
        string [] sms_procesado =  new string[4];
        int cont = 0;
        try
        {
            cont = sms.IndexOf("/");
            sms_procesado[0] = sms.Substring(cont - 2, 8);

            cont = sms.LastIndexOf(":");
            sms_procesado[1] = sms.Substring(cont - 5, 8);

            cont = sms.LastIndexOf("\"");

            if (sms.Contains("OK")) sms_procesado[2] = sms.Substring(cont + 2,
            sms.Length - cont -  5);
            else sms_procesado[2] = sms.Substring(cont + 2, sms.Length - cont –
            3);
            //MessageBox.Show(sms_procesado[2]);
            if (sms.Contains("D\",\""))
            {
                cont = sms.IndexOf("D\",\"");
                sms_procesado[3] = sms.Substring(cont + 4, 9);
            }

            if (sms.Contains("D\",\"+"))
            {
                cont = sms.IndexOf("D\",\"+");
                sms_procesado[3] = "0" + sms.Substring(cont + 8, 8);
            }
        }
        catch(Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
                    return sms_procesado;
            }

    public string borrar_sms(int pos)
```

```
{
    string msj = "";
    try
    {
        escribir("AT+CMGD=" + pos);
        msj = "Mensaje " + pos + " borrado";
    }
    catch(Exception ex)
    {
        msj = ex.Message;
    }
    return msj;
}
```

Clase Base de Datos

```
using System;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Data;

namespace Control_Base_de_Datos
{
    /// <summary>
    /// Description of clsConexion.
    /// </summary>
    public class clsBD
    {
   string cadena  = "server=(local);
   database=bd_control;uid=Carlos;password=ciml";

            string consulta = "";

   SqlCommand command;
   //SqlDataReader dr;
   //DataSet ds = new DataSet();
   public SqlConnection conexion;


   public clsBD()
            {
    conexion = new SqlConnection(cadena);
    //conex = new SqlConnection(cadena);
            }

            public void abrir()
            {
                conexion.Open();
```

```
        }

        public void cerrar()
        {
                conexion.Close();
        }

        public int contar_igual(string tabla, string registro, string valor)
        {
int num = 0;
try
{
   string cadena;
   cadena = "select count(*) from " + tabla + " where " + registro + " = '" +
   valor + "'";
   //MessageBox.Show(cadena);
   //cadena = consulta;
   conexion.Open();

   command = new SqlCommand(cadena, conexion);
   //Ejecutar un conteo, devuelve entero
   num = Convert.ToInt16(command.ExecuteScalar());
   conexion.Close();

}
catch(Exception ex)
{
   MessageBox.Show(ex.Message);
   Application.Exit();
}
                return num;
        }

        public void agregar(string tabla, string [] campos, string [] datos)
        {
                try
                {
                        string dato = "";
                        string campo = "";

                        for (int i=0; i<campos.Length; i++)
                        {
                                campo += campos[i] + ",";
                                dato += "'" + datos[i] + "',";
                        }
                        //MessageBox.Show(campo);
                          //MessageBox.Show(dato);
                        if (dato.Length > 0) dato = dato.Remove(dato.Length –
                        1);
```

```
                    if (campo.Length > 0) campo =
                    campo.Remove(campo.Length - 1);
                    //MessageBox.Show(campo);
                    //MessageBox.Show(dato);
                    string cadena = "insert into " + tabla + " (" + campo + ")
                    values (" + dato + ")";
                    //MessageBox.Show(cadena);

                    conexion.Open();
                    command = new SqlCommand(cadena, conexion);
                    //Ejecuta consulta de acción
                    command.ExecuteNonQuery();
                    conexion.Close();
                    MessageBox.Show("Dato Guardado");
                }
                catch(Exception ex)
                {
                    conexion.Close();
                    MessageBox.Show(ex.Message);
                }
        }

        public void eliminar(string tabla, string registro, string valor)
        {
            string cadena = "delete from " + tabla + " where " + registro + "
            = '" + valor + "'";
            conexion.Open();
             command = new SqlCommand(cadena, conexion);
            //Ejecuta consulta de acción
            command.ExecuteNonQuery();
            conexion.Close();
        }

    public DataTable grid_todo(string tabla)
    {
            DataTable dt = new DataTable();
            try
            {
            string cadena = "select * from " + tabla;
            conexion.Open();
            command = new SqlCommand(cadena, conexion);
            SqlDataAdapter da = new SqlDataAdapter(command);
                da.Fill(dt);
                conexion.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            return dt;
```

```
}

public DataTable grid(string tabla, string []campos, string[] datos)
{
                DataTable dt = new DataTable();
                try
                {
                        consulta = "select * from " + tabla + " where ";
                        consulta += campos[0] + " between '" + datos[0] + "' and
                        '" + datos[1] + "' AND ";
                        consulta += campos[1] + " between '" + datos[2] + "' and
                        '" + datos[3] + "';";

                        conexion.Open();
                        command = new SqlCommand(consulta, conexion);
                        SqlDataAdapter da = new SqlDataAdapter(command);
                        da.Fill(dt);
                        conexion.Close();
                        //MessageBox.Show(consulta);
                }
                catch (Exception ex)
                {
                        //MessageBox.Show(consulta);
                        conexion.Close();
                        MessageBox.Show(ex.Message);
                }
                return dt;
 }


public DataTable validar_ingreso_manual(string tabla, string[] campos,
string[] datos)
 {
 DataTable dt = new DataTable();
 try
 {
    consulta = "select * from " + tabla + " where ";
    consulta += campos[0] + " = '" + datos[0] + "' and ";
    consulta += campos [1] + " = '" + datos[1] + "';";


    conexion.Open();
    command = new SqlCommand(consulta, conexion);
    SqlDataAdapter da = new SqlDataAdapter(command);
    da.Fill(dt);
    conexion.Close();
    //MessageBox.Show(consulta);
 }
 catch (Exception ex)
 {
```

```
            //MessageBox.Show(consulta);
            conexion.Close();
            MessageBox.Show(ex.Message);
        }
        return dt;
    }
      }
}
```

ANEXO A6: DATASHEET MICROCONTROLADOR ATMEGA8

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 130 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
- High Endurance Non-volatile Memory segments
  - 8K Bytes of In-System Self-programmable Flash program memory
  - 512 Bytes EEPROM
  - 1K Byte Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler, one Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Three PWM Channels
  - 8-channel ADC in TQFP and QFN/MLF package
    Eight Channels 10-bit Accuracy
  - 6-channel ADC in PDIP package
    Six Channels 10-bit Accuracy
  - 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
  - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-lead PDIP, 32-lead TQFP, and 32-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.5V (ATmega8L)
  - 4.5 - 5.5V (ATmega8)
- Speed Grades
  - 0 - 8 MHz (ATmega8L)
  - 0 - 16 MHz (ATmega8)
- Power Consumption at 4 Mhz, 3V, 25°C
  - Active: 3.6 mA
  - Idle Mode: 1.0 mA
  - Power-down Mode: 0.5 µA

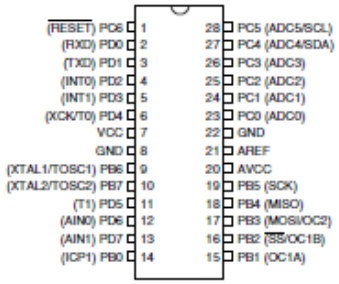
8-bit AVR® with 8K Bytes In-System Programmable Flash

ATmega8
ATmega8L

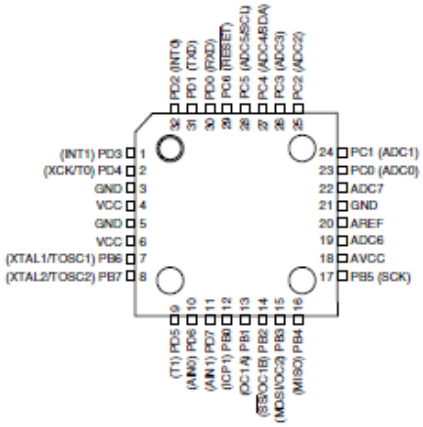
Summary

Pin Configurations

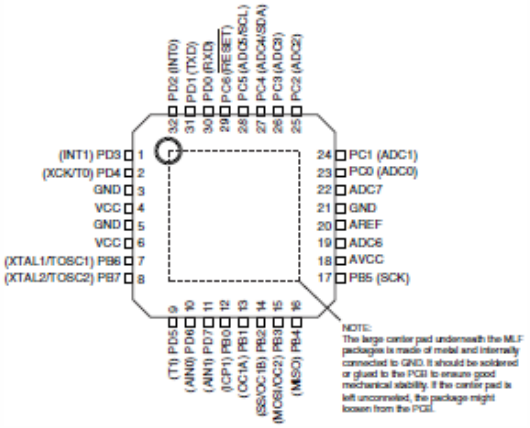
PDIP



TQFP Top View



MLF Top View



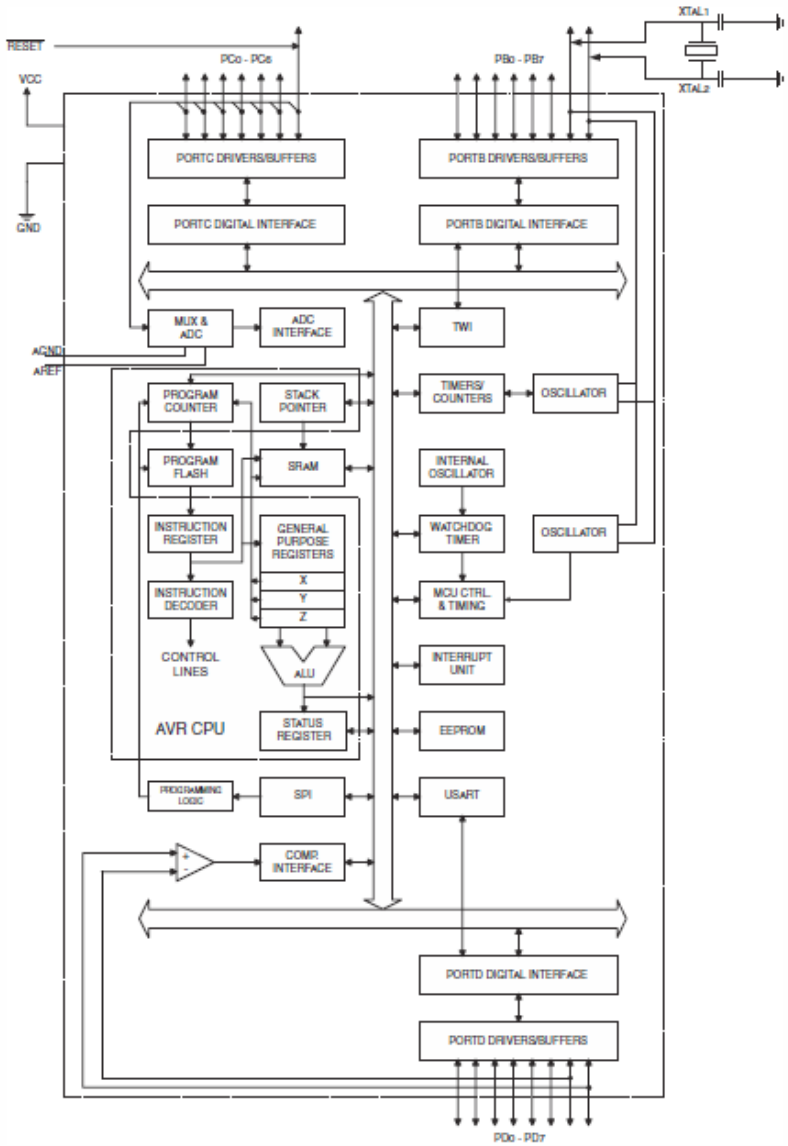
ATmega8(L)

Overview

The ATmega8 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega8 achieves throughputs approaching 1 MIPS per MHz, allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 1. 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 ATmega8 provides the following features: 8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes of EEPROM, 1K byte of SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, a 6-channel ADC (eight channels in TQFP and QFN/MLF packages) with 10-bit accuracy, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the 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 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.

The device is manufactured using Atmel's high density non-volatile memory technology. The Flash Program memory can be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile 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 ATmega8 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega8 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.

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.

Pin Descriptions

VCC	Digital supply voltage.
GND	Ground.
Port B (PB7..PB0) XTAL1/XTAL2/TOSC1/ TOSC2	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. Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit. Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier. If the Internal Calibrated RC Oscillator is used as chip clock source, PB7..6 is used as TOSC2..1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set. The various special features of Port B are elaborated in "Alternate Functions of Port B" on page 58 and "System Clock and Clock Options" on page 25.
Port C (PC5..PC0)	Port C is an 7-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.
PC6/$\overline{\text{RESET}}$	If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C. If the RSTDISBL Fuse is unprogrammed, PC6 is used as a 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 38. Shorter pulses are not guaranteed to generate a Reset. The various special features of Port C are elaborated on page 61.
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 ATmega8 as listed on page 63.
$\overline{\text{RESET}}$	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 38. Shorter pulses are not guaranteed to generate a reset.

AV_{CC}	AV_{CC} is the supply voltage pin for the A/D Converter, Port C (3..0), and ADC (7..6). 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. Note that Port C (5..4) use digital supply voltage, V_{CC}.
AREF	AREF is the analog reference pin for the A/D Converter.
ADC7..6 (TQFP and QFN/MLF Package Only)	In the TQFP and QFN/MLF package, ADC7..6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

Resources

A comprehensive set of development tools, application notes and datasheets are available for download on http://www.atmel.com/avr.

Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	11
0x3E (0x5E)	SPH	–	–	–	–	–	SP10	SP9	SP8	13
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	13
0x3C (0x5C)	Reserved									
0x3B (0x5B)	GICR	INT1	INT0	–	–	–	–	IVSEL	IVCE	49, 67
0x3A (0x5A)	GIFR	INTF1	INTF0	–	–	–	–	–	–	68
0x39 (0x59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	72, 102, 122
0x38 (0x58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	73, 102, 122
0x37 (0x57)	SPMCR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	213
0x36 (0x56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	171
0x35 (0x55)	MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	33, 66
0x34 (0x54)	MCUCSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	41
0x33 (0x53)	TCCR0	–	–	–	–	–	CS02	CS01	CS00	72
0x32 (0x52)	TCNT0	Timer/Counter0 (8 Bits)								72
0x31 (0x51)	OSCCAL	Oscillator Calibration Register								31
0x30 (0x50)	SFIOR	–	–	–	–	ACME	PUD	PSR2	PSR10	58, 75, 123, 193
0x2F (0x4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	96
0x2E (0x4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	100
0x2D (0x4D)	TCNT1H	Timer/Counter1 – Counter Register High byte								101
0x2C (0x4C)	TCNT1L	Timer/Counter1 – Counter Register Low byte								101
0x2B (0x4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High byte								101
0x2A (0x4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low byte								101
0x29 (0x49)	OCR1BH	Timer/Counter1 – Output Compare Register B High byte								101
0x28 (0x48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low byte								101
0x27 (0x47)	ICR1H	Timer/Counter1 – Input Capture Register High byte								102
0x26 (0x46)	ICR1L	Timer/Counter1 – Input Capture Register Low byte								102
0x25 (0x45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	117
0x24 (0x44)	TCNT2	Timer/Counter2 (8 Bits)								119
0x23 (0x43)	OCR2	Timer/Counter2 Output Compare Register								119
0x22 (0x42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	119
0x21 (0x41)	WDTCR	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0	43
0x20[1] (0x40)[1]	UBRRH	URSEL	–	–	–	UBRR[11:8]				158
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	156
0x1F (0x3F)	EEARH	–	–	–	–	–	–	–	EEAR8	20
0x1E (0x3E)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	20
0x1D (0x3D)	EEDR	EEPROM Data Register								20
0x1C (0x3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	20
0x1B (0x3B)	Reserved									
0x1A (0x3A)	Reserved									
0x19 (0x39)	Reserved									
0x18 (0x38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	65
0x17 (0x37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	65
0x16 (0x36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	65
0x15 (0x35)	PORTC	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	65
0x14 (0x34)	DDRC	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	65
0x13 (0x33)	PINC	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	65
0x12 (0x32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	65
0x11 (0x31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	65
0x10 (0x30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	65
0x0F (0x2F)	SPDR	SPI Data Register								131
0x0E (0x2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	131
0x0D (0x2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	129
0x0C (0x2C)	UDR	USART I/O Data Register								153
0x0B (0x2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	154
0x0A (0x2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	155
0x09 (0x29)	UBRRL	USART Baud Rate Register Low byte								158
0x08 (0x28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	194
0x07 (0x27)	ADMUX	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	205
0x06 (0x26)	ADCSRA	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	207
0x05 (0x25)	ADCH	ADC Data Register High byte								208
0x04 (0x24)	ADCL	ADC Data Register Low byte								208
0x03 (0x23)	TWDR	Two-wire Serial Interface Data Register								173
0x02 (0x22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	174

Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x01 (0x21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	173
0x00 (0x20)	TWBR	Two-wire Serial Interface Bit Rate Register								171

Notes:
1. Refer to the USART description for details on how to access UBRRH and UCSRC.
2. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
3. Some of the Status Flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

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 ← 0xFF − Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 − 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 • (0xFF - 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 ← 0xFF	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
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
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**

Instruction Set Summary (Continued)

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
Mnemonics	**Operands**	**Description**	**Operation**	**Flags**	**#Clocks**

Instruction Set Summary (Continued)

CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1

Ordering Information

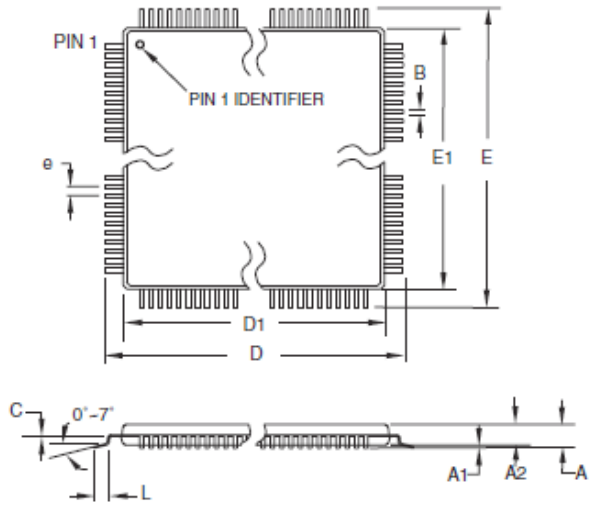
Speed (MHz)	Power Supply	Ordering Code	Package[1]	Operation Range
8	2.7 - 5.5	ATmega8L-8AC ATmega8L-8PC ATmega8L-8MC	32A 28P3 32M1-A	Commercial (0°C to 70°C)
		ATmega8L-8AI ATmega8L-8AU[2] ATmega8L-8PI ATmega8L-8PU[2] ATmega8L-8MI ATmega8L-8MU[2]	32A 32A 28P3 28P3 32M1-A 32M1-A	Industrial (-40°C to 85°C)
16	4.5 - 5.5	ATmega8-16AC ATmega8-16PC ATmega8-16MC	32A 28P3 32M1-A	Commercial (0°C to 70°C)
		ATmega8-16AI ATmega8-16AU[2] ATmega8-16PI ATmega8-16PU[2] ATmega8-16MI ATmega8-16MU[2]	32A 32A 28P3 28P3 32M1-A 32M1-A	Industrial (-40°C to 85°C)

Notes: 1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging alternative, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

Package Type	
32A	32-lead, Thin (1.0 mm) Plastic Quad Flat Package (TQFP)
28P3	28-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP)
32M1-A	32-pad, 5 x 5 x 1.0 body, Lead Pitch 0.50 mm Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)

Packaging Information

32A



COMMON DIMENSIONS
(Unit of Measure = mm)

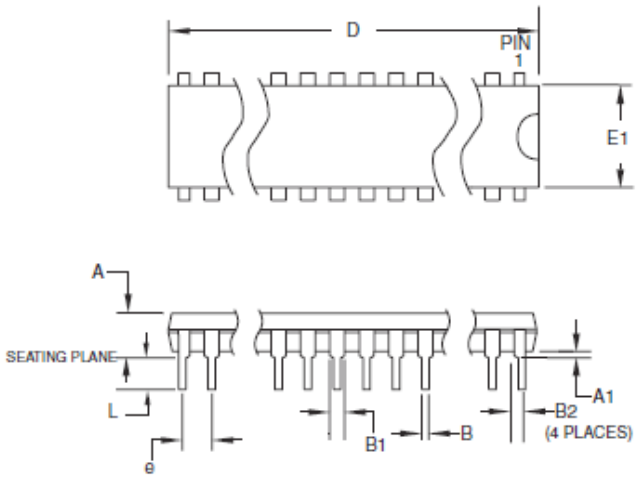
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	8.75	9.00	9.25	
D1	6.90	7.00	7.10	Note 2
E	8.75	9.00	9.25	
E1	6.90	7.00	7.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.80 TYP			

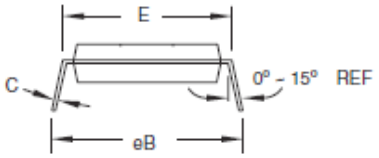
Notes:
1. This package conforms to JEDEC reference MS-026, Variation ABA.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10 mm maximum.

10/5/2001

2325 Orchard Parkway San Jose, CA 95131	TITLE **32A**, 32-lead, 7 x 7 mm Body Size, 1.0 mm Body Thickness, 0.8 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP)	DRAWING NO. 32A	REV. B

28P3





COMMON DIMENSIONS
(Unit of Measure = mm)

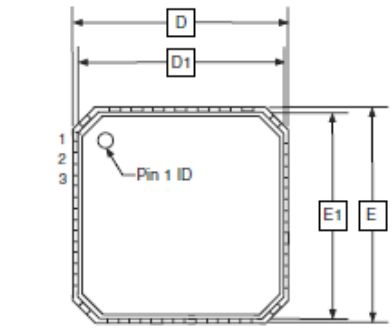
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.5724	
A1	0.508	–	–	
D	34.544	–	34.798	Note 1
E	7.620	–	8.255	
E1	7.112	–	7.493	Note 1
B	0.381	–	0.533	
B1	1.143	–	1.397	
B2	0.762	–	1.143	
L	3.175	–	3.429	
C	0.203	–	0.356	
eB	–	–	10.160	
e	2.540 TYP			

Note: 1. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

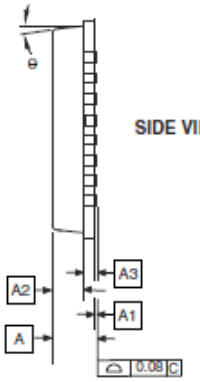
09/28/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**28P3**, 28-lead (0.300"/7.62 mm Wide) Plastic Dual Inline Package (PDIP)	28P3	B

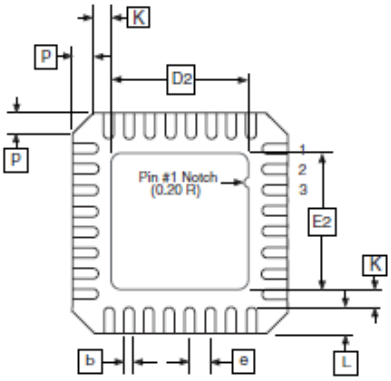
32M1-A



TOP VIEW



SIDE VIEW



BOTTOM VIEW

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.90	1.00	
A1	–	0.02	0.05	
A2	–	0.65	1.00	
A3		0.20 REF		
b	0.18	0.23	0.30	
D	4.90	5.00	5.10	
D1	4.70	4.75	4.80	
D2	2.95	3.10	3.25	
E	4.90	5.00	5.10	
E1	4.70	4.75	4.80	
E2	2.95	3.10	3.25	
e		0.50 BSC		
L	0.30	0.40	0.50	
P	–	–	0.60	
θ	–	–	12^0	
K	0.20	–	–	

Note: JEDEC Standard MO-220, Fig. 2 (Anvil Singulation), VHHD-2.

5/25/06

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**32M1-A**, 32-pad, 5 x 5 x 1.0 mm Body, Lead Pitch 0.50 mm, 3.10 mm Exposed Pad, Micro Lead Frame Package (MLF)	32M1-A	E

ATmega8(L)

Errata

The revision letter in this section refers to the revision of the ATmega8 device.

ATmega8 Rev. D to I

- **First Analog Comparator conversion may be delayed**
- **Interrupts may be lost when writing the timer registers in the asynchronous timer**
- **Signature may be Erased in Serial Programming Mode**
- **CKOPT Does not Enable Internal Capacitors on XTALn/TOSCn Pins when 32 KHz Oscillator is Used to Clock the Asynchronous Timer/Counter2**
- **Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request**

1. **First Analog Comparator conversion may be delayed**

   If the device is powered by a slow rising V_{CC}, the first Analog Comparator conversion will take longer than expected on some devices.

   Problem Fix / Workaround

   When the device has been powered or reset, disable then enable theAnalog Comparator before the first conversion.

2. **Interrupts may be lost when writing the timer registers in the asynchronous timer**

   If one of the timer registers which is synchronized to the asynchronous timer2 clock is written in the cycle before a overflow interrupt occurs, the interrupt may be lost.

   Problem Fix / Workaround

   Always check that the Timer2 Timer/Counter register, TCNT2, does not have the value 0xFF before writing the Timer2 Control Register, TCCR2, or Output Compare Register, OCR2

3. **Signature may be Erased in Serial Programming Mode**

   If the signature bytes are read before a chiperase command is completed, the signature may be erased causing the device ID and calibration bytes to disappear. This is critical, especially, if the part is running on internal RC oscillator.

   Problem Fix / Workaround:

   Ensure that the chiperase command has exceeded before applying the next command.

4. **CKOPT Does not Enable Internal Capacitors on XTALn/TOSCn Pins when 32 KHz Oscillator is Used to Clock the Asynchronous Timer/Counter2**

   When the internal RC Oscillator is used as the main clock source, it is possible to run the Timer/Counter2 asynchronously by connecting a 32 KHz Oscillator between XTAL1/TOSC1 and XTAL2/TOSC2. But when the internal RC Oscillator is selected as the main clock source, the CKOPT Fuse does not control the internal capacitors on XTAL1/TOSC1 and XTAL2/TOSC2. As long as there are no capacitors connected to XTAL1/TOSC1 and XTAL2/TOSC2, safe operation of the Oscillator is not guaranteed.

   Problem Fix / Workaround

   Use external capacitors in the range of 20 - 36 pF on XTAL1/TOSC1 and XTAL2/TOSC2. This will be fixed in ATmega8 Rev. G where the CKOPT Fuse will control internal capacitors also when internal RC Oscillator is selected as main clock source. For ATmega8 Rev. G, CKOPT = 0 (programmed) will enable the internal capacitors on XTAL1 and XTAL2. Customers who want compatibility between Rev. G and older revisions, must ensure that CKOPT is unprogrammed (CKOPT = 1).

5. **Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.**

   Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

ANEXO A7: DATASHEET RELOJ CALENDARIO DS1307

DALLAS SEMICONDUCTOR

DS1307
64 x 8 Serial Real-Time Clock

www.maxim-ic.com

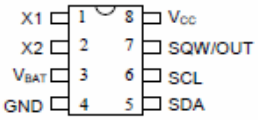
FEATURES

- Real-time clock (RTC) counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap-year compensation valid up to 2100
- 56-byte, battery-backed, nonvolatile (NV) RAM for data storage
- Two-wire serial interface
- Programmable squarewave output signal
- Automatic power-fail detect and switch circuitry
- Consumes less than 500nA in battery backup mode with oscillator running
- Optional industrial temperature range: -40°C to +85°C
- Available in 8-pin DIP or SOIC
- Underwriters Laboratory (UL) recognized

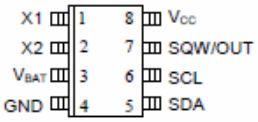
ORDERING INFORMATION

DS1307	8-Pin DIP (300-mil)
DS1307Z	8-Pin SOIC (150-mil)
DS1307N	8-Pin DIP (Industrial)
DS1307ZN	8-Pin SOIC (Industrial)

PIN ASSIGNMENT



DS1307 8-Pin DIP (300-mil)



DS1307 8-Pin SOIC (150-mil)

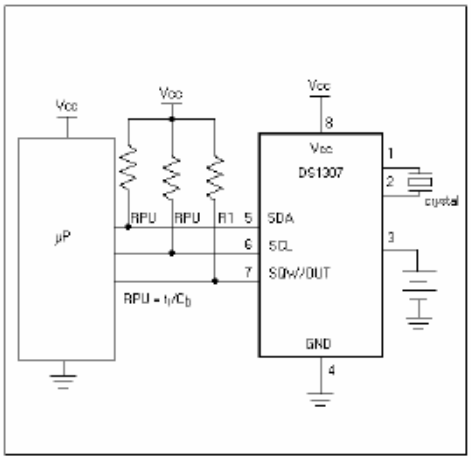
PIN DESCRIPTION

V_{CC}	- Primary Power Supply
X1, X2	- 32.768kHz Crystal Connection
V_{BAT}	- +3V Battery Input
GND	- Ground
SDA	- Serial Data
SCL	- Serial Clock
SQW/OUT	- Square Wave/Output Driver

DESCRIPTION

The DS1307 Serial Real-Time Clock is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially via a 2-wire, bi-directional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power sense circuit that detects power failures and automatically switches to the battery supply.

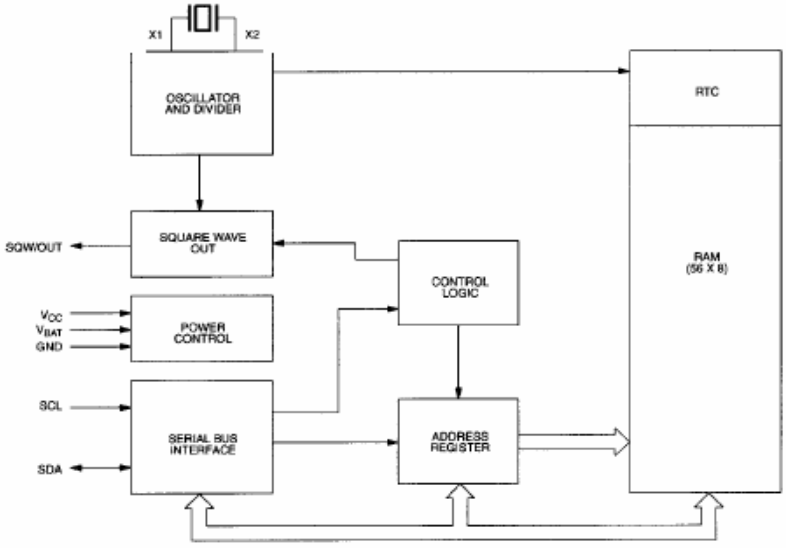
TYPICAL OPERATING CIRCUIT



OPERATION

The DS1307 operates as a slave device on the serial bus. Access is obtained by implementing a START condition and providing a device identification code followed by a register address. Subsequent registers can be accessed sequentially until a STOP condition is executed. When V_{CC} falls below 1.25 x V_{BAT} the device terminates an access in progress and resets the device address counter. Inputs to the device will not be recognized at this time to prevent erroneous data from being written to the device from an out of tolerance system. When V_{CC} falls below V_{BAT} the device switches into a low-current battery backup mode. Upon power-up, the device switches from battery to V_{CC} when V_{CC} is greater than V_{BAT} + 0.2V and recognizes inputs when V_{CC} is greater than 1.25 x V_{BAT}. The block diagram in Figure 1 shows the main elements of the serial RTC.

DS1307 BLOCK DIAGRAM Figure 1



SIGNAL DESCRIPTIONS

V_{CC}, GND – DC power is provided to the device on these pins. V_{CC} is the +5V input. When 5V is applied within normal limits, the device is fully accessible and data can be written and read. When a 3V battery is connected to the device and V_{CC} is below 1.25 x V_{BAT}, reads and writes are inhibited. However, the timekeeping function continues unaffected by the lower input voltage. As V_{CC} falls below V_{BAT} the RAM and timekeeper are switched over to the external power supply (nominal 3.0V DC) at V_{BAT}.

V_{BAT} – Battery input for any standard 3V lithium cell or other energy source. Battery voltage must be held between 2.0V and 3.5V for proper operation. The nominal write protect trip point voltage at which access to the RTC and user RAM is denied is set by the internal circuitry as 1.25 x V_{BAT} nominal. A lithium battery with 48mAhr or greater will back up the DS1307 for more than 10 years in the absence of power at 25°C. UL recognized to ensure against reverse charging current when used in conjunction with a lithium battery.

See "Conditions of Acceptability" at http://www.maxim-ic.com/TechSupport/QA/ntrl.htm.

SCL (Serial Clock Input) – SCL is used to synchronize data movement on the serial interface.

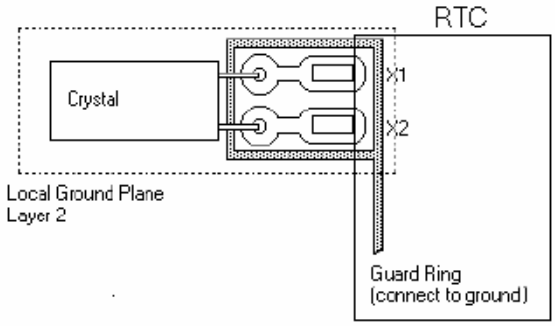
SDA (Serial Data Input/Output) – SDA is the input/output pin for the 2-wire serial interface. The SDA pin is open drain which requires an external pullup resistor.

SQW/OUT (Square Wave/Output Driver) – When enabled, the SQWE bit set to 1, the SQW/OUT pin outputs one of four square wave frequencies (1Hz, 4kHz, 8kHz, 32kHz). The SQW/OUT pin is open drain and requires an external pull-up resistor. SQW/OUT will operate with either Vcc or Vbat applied.

X1, X2 – Connections for a standard 32.768kHz quartz crystal. The internal oscillator circuitry is designed for operation with a crystal having a specified load capacitance (CL) of 12.5pF.

For more information on crystal selection and crystal layout considerations, please consult Application Note 58, "Crystal Considerations with Dallas Real-Time Clocks." The DS1307 can also be driven by an external 32.768kHz oscillator. In this configuration, the X1 pin is connected to the external oscillator signal and the X2 pin is floated.

RECOMMENDED LAYOUT FOR CRYSTAL



CLOCK ACCURACY

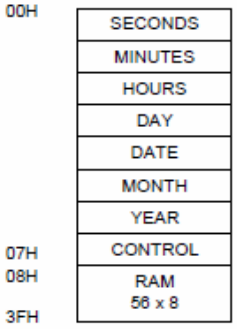
The accuracy of the clock is dependent upon the accuracy of the crystal and the accuracy of the match between the capacitive load of the oscillator circuit and the capacitive load for which the crystal was trimmed. Additional error will be added by crystal frequency drift caused by temperature shifts. External circuit noise coupled into the oscillator circuit may result in the clock running fast. See Application Note 58, "Crystal Considerations with Dallas Real-Time Clocks" for detailed information.

Please review Application Note 95, "Interfacing the DS1307 with a 8051-Compatible Microcontroller" for additional information.

RTC AND RAM ADDRESS MAP

The address map for the RTC and RAM registers of the DS1307 is shown in Figure 2. The RTC registers are located in address locations 00h to 07h. The RAM registers are located in address locations 08h to 3Fh. During a multi-byte access, when the address pointer reaches 3Fh, the end of RAM space, it wraps around to location 00h, the beginning of the clock space.

DS1307 ADDRESS MAP Figure 2



CLOCK AND CALENDAR

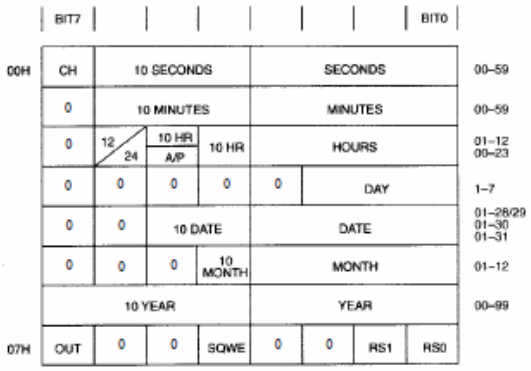
The time and calendar information is obtained by reading the appropriate register bytes. The RTC registers are illustrated in Figure 3. The time and calendar are set or initialized by writing the appropriate register bytes. The contents of the time and calendar registers are in the BCD format. Bit 7 of register 0 is the clock halt (CH) bit. When this bit is set to a 1, the oscillator is disabled. When cleared to a 0, the oscillator is enabled.

Please note that the initial power-on state of all registers is not defined. Therefore, it is important to enable the oscillator (CH bit = 0) during initial configuration.

The DS1307 can be run in either 12-hour or 24-hour mode. Bit 6 of the hours register is defined as the 12- or 24-hour mode select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10 hour bit (20-23 hours).

On a 2-wire START, the current time is transferred to a second set of registers. The time information is read from these secondary registers, while the clock may continue to run. This eliminates the need to re-read the registers in case of an update of the main registers during a read.

DS1307 TIMEKEEPER REGISTERS Figure 3



CONTROL REGISTER

The DS1307 control register is used to control the operation of the SQW/OUT pin.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	0	0	SQWE	0	0	RS1	RS0

OUT (Output control): This bit controls the output level of the SQW/OUT pin when the square wave output is disabled. If SQWE = 0, the logic level on the SQW/OUT pin is 1 if OUT = 1 and is 0 if OUT = 0.

SQWE (Square Wave Enable): This bit, when set to a logic 1, will enable the oscillator output. The frequency of the square wave output depends upon the value of the RS0 and RS1 bits. With the square wave output set to 1Hz, the clock registers update on the falling edge of the square wave.

RS (Rate Select): These bits control the frequency of the square wave output when the square wave output has been enabled. Table 1 lists the square wave frequencies that can be selected with the RS bits.

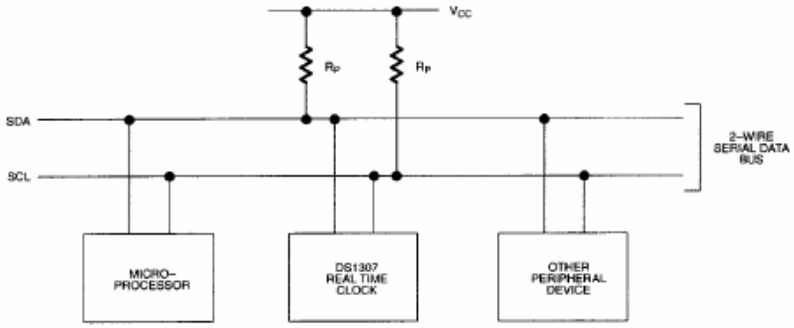
SQUAREWAVE OUTPUT FREQUENCY Table 1

RS1	RS0	SQW OUTPUT FREQUENCY
0	0	1Hz
0	1	4.096kHz
1	0	8.192kHz
1	1	32.768kHz

2-WIRE SERIAL DATA BUS

The DS1307 supports a bi-directional, 2-wire bus and data transmission protocol. A device that sends data onto the bus is defined as a transmitter and a device receiving data as a receiver. The device that controls the message is called a master. The devices that are controlled by the master are referred to as slaves. The bus must be controlled by a master device that generates the serial clock (SCL), controls the bus access, and generates the START and STOP conditions. The DS1307 operates as a slave on the 2-wire bus. A typical bus configuration using this 2-wire protocol is show in Figure 4.

TYPICAL 2-WIRE BUS CONFIGURATION Figure 4



Figures 5, 6, and 7 detail how data is transferred on the 2-wire bus.

- Data transfer may be initiated only when the bus is not busy.
- During data transfer, the data line must remain stable whenever the clock line is HIGH. Changes in the data line while the clock line is high will be interpreted as control signals.

Accordingly, the following bus conditions have been defined:

Bus not busy: Both data and clock lines remain HIGH.

Start data transfer: A change in the state of the data line, from HIGH to LOW, while the clock is HIGH, defines a START condition.

Stop data transfer: A change in the state of the data line, from LOW to HIGH, while the clock line is HIGH, defines the STOP condition.

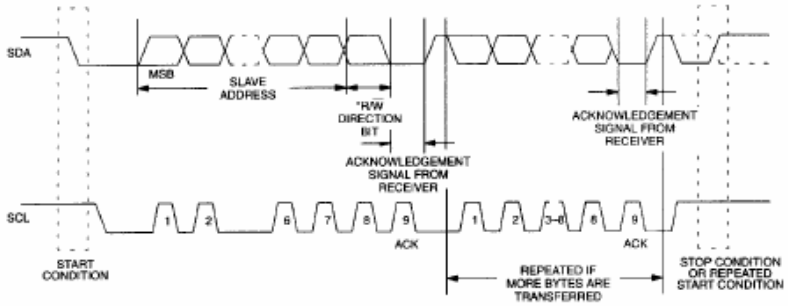
Data valid: The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the HIGH period of the clock signal. The data on the line must be changed during the LOW period of the clock signal. There is one clock pulse per bit of data.

Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of data bytes transferred between START and STOP conditions is not limited, and is determined by the master device. The information is transferred byte-wise and each receiver acknowledges with a ninth bit. Within the 2-wire bus specifications a regular mode (100kHz clock rate) and a fast mode (400kHz clock rate) are defined. The DS1307 operates in the regular mode (100kHz) only.

Acknowledge: Each receiving device, when addressed, is obliged to generate an acknowledge after the reception of each byte. The master device must generate an extra clock pulse which is associated with this acknowledge bit.

A device that acknowledges must pull down the SDA line during the acknowledge clock pulse in such a way that the SDA line is stable LOW during the HIGH period of the acknowledge related clock pulse. Of course, setup and hold times must be taken into account. A master must signal an end of data to the slave by not generating an acknowledge bit on the last byte that has been clocked out of the slave. In this case, the slave must leave the data line HIGH to enable the master to generate the STOP condition.

DATA TRANSFER ON 2-WIRE SERIAL BUS Figure 5



Depending upon the state of the R/$\overline{W}$ bit, two types of data transfer are possible:

1. **Data transfer from a master transmitter to a slave receiver.** The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte. Data is transferred with the most significant bit (MSB) first.

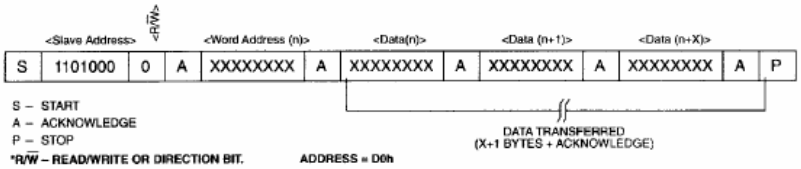
2. **Data transfer from a slave transmitter to a master receiver.** The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. This is followed by the slave transmitting a number of data bytes. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a "not acknowledge" is returned.

The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the bus will not be released. Data is transferred with the most significant bit (MSB) first.

The DS1307 may operate in the following two modes:

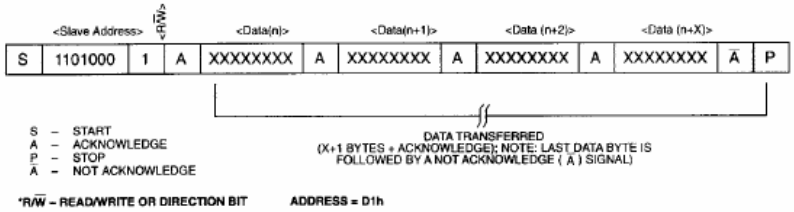
1. **Slave receiver mode (DS1307 write mode):** Serial data and clock are received through SDA and SCL. After each byte is received an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and *direction bit (See Figure 6). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7 bit DS1307 address, which is 1101000, followed by the *direction bit (R/$\overline{W}$) which, for a write, is a 0. After receiving and decoding the address byte the device outputs an acknowledge on the SDA line. After the DS1307 acknowledges the slave address + write bit, the master transmits a register address to the DS1307 This will set the register pointer on the DS1307. The master will then begin transmitting each byte of data with the DS1307 acknowledging each byte received. The master will generate a stop condition to terminate the data write.

DATA WRITE – SLAVE RECEIVER MODE Figure 6



2. **Slave transmitter mode (DS1307 read mode):** The first byte is received and handled as in the slave receiver mode. However, in this mode, the *direction bit will indicate that the transfer direction is reversed. Serial data is transmitted on SDA by the DS1307 while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer (See Figure 7). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7-bit DS1307 address, which is 1101000, followed by the *direction bit (R/$\overline{W}$) which, for a read, is a 1. After receiving and decoding the address byte the device inputs an acknowledge on the SDA line. The DS1307 then begins to transmit data starting with the register address pointed to by the register pointer. If the register pointer is not written to before the initiation of a read mode the first address that is read is the last one stored in the register pointer. The DS1307 must receive a "not acknowledge" to end a read.

DATA READ – SLAVE TRANSMITTER MODE Figure 7



ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	-0.5V to +7.0V
Storage Temperature	-55°C to +125°C
Soldering Temperature	260°C for 10 seconds DIP See JPC/JEDEC Standard J-STD-020A for Surface Mount Devices

* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

Range	Temperature	V_{CC}
Commercial	0°C to +70°C	4.5V to 5.5V V_{CC1}
Industrial	-40°C to +85°C	4.5V to 5.5V V_{CC1}

RECOMMENDED DC OPERATING CONDITIONS

(Over the operating range*)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V_{CC}	4.5	5.0	5.5	V	
Logic 1	V_{IH}	2.2		V_{CC} + 0.3	V	
Logic 0	V_{IL}	-0.5		+0.8	V	
V_{BAT} Battery Voltage	V_{BAT}	2.0		3.5	V	

*Unless otherwise specified.

DC ELECTRICAL CHARACTERISTICS

(Over the operating range*)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Input Leakage (SCL)	I_{LI}			1	µA	
I/O Leakage (SDA & SQW/OUT)	I_{LO}			1	µA	
Logic 0 Output (I_{OL} = 5mA)	V_{OL}			0.4	V	
Active Supply Current	I_{CCA}			1.5	mA	7
Standby Current	I_{CCS}			200	µA	1
Battery Current (OSC ON); SQW/OUT OFF	I_{BAT1}		300	500	nA	2
Battery Current (OSC ON); SQW/OUT ON (32kHz)	I_{BAT2}		480	800	nA	
Power-Fail Voltage	V_{PF}	1.216 x V_{BAT}	1.25 x V_{BAT}	1.284 x V_{BAT}	V	8

*Unless otherwise specified.

AC ELECTRICAL CHARACTERISTICS

(Over the operating range*)

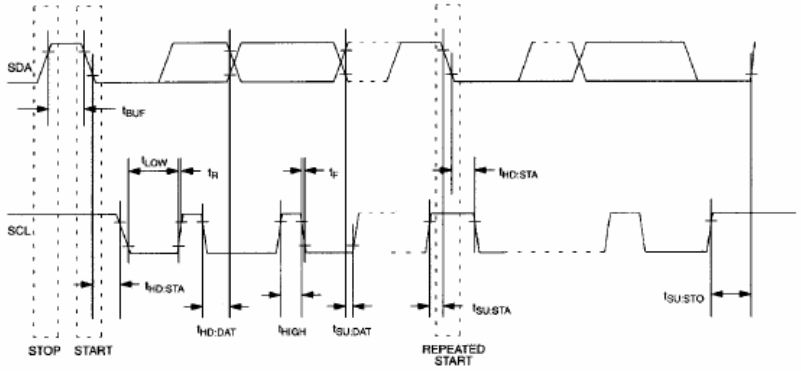
PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
SCL Clock Frequency	f_{SCL}	0		100	kHz	
Bus Free Time Between a STOP and START Condition	t_{BUF}	4.7			µs	
Hold Time (Repeated) START Condition	$t_{HD:STA}$	4.0			µs	3
LOW Period of SCL Clock	t_{LOW}	4.7			µs	
HIGH Period of SCL Clock	t_{HIGH}	4.0			µs	
Set-up Time for a Repeated START Condition	$t_{SU:STA}$	4.7			µs	
Data Hold Time	$t_{HD:DAT}$	0			µs	4,5
Data Set-up Time	$t_{SU:DAT}$	250			ns	
Rise Time of Both SDA and SCL Signals	t_R			1000	ns	
Fall Time of Both SDA and SCL Signals	t_F			300	ns	
Set-up Time for STOP Condition	$t_{SU:STO}$	4.7			µs	
Capacitive Load for each Bus Line	C_B			400	pF	6
I/O Capacitance (T_A = 25°C)	$C_{I/O}$		10		pF	
Crystal Specified Load Capacitance (T_A = 25°C)			12.5		pF	

*Unless otherwise specified.

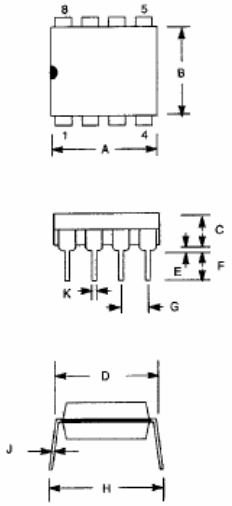
NOTES:

1. I_{CCS} specified with V_{CC} = 5.0V and SDA, SCL = 5.0V.
2. V_{CC} = 0V, V_{BAT} = 3V.
3. After this period, the first clock pulse is generated.
4. A device must internally provide a hold time of at least 300ns for the SDA signal (referred to the V_{IHMIN} of the SCL signal) in order to bridge the undefined region of the falling edge of SCL.
5. The maximum $t_{HD:DAT}$ has only to be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal.
6. C_B – Total capacitance of one bus line in pF.
7. I_{CCA} – SCL clocking at max frequency = 100kHz.
8. V_{PF} measured at V_{BAT} = 3.0V.

TIMING DIAGRAM Figure 8

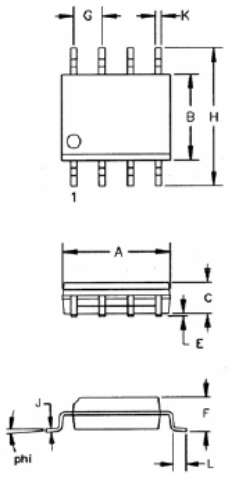


DS1307 64 X 8 SERIAL REAL-TIME CLOCK
8-PIN DIP MECHANICAL DIMENSIONS



PKG	8-PIN	
DIM	MIN	MAX
A IN.	0.360	0.400
MM	9.14	10.16
B IN.	0.240	0.260
MM	6.10	6.60
C IN.	0.120	0.140
MM	3.05	3.56
D IN.	0.300	0.325
MM	7.62	8.26
E IN.	0.015	0.040
MM	0.38	1.02
F IN.	0.120	0.140
MM	3.04	3.56
G IN.	0.090	0.110
MM	2.29	2.79
H IN.	0.320	0.370
MM	8.13	9.40
J IN.	0.008	0.012
MM	0.20	0.30
K IN.	0.015	0.021
MM	0.38	0.53

DS1307Z 64 X 8 SERIAL REAL-TIME CLOCK
8-PIN SOIC (150-MIL) MECHANICAL DIMENSIONS



PKG	8-PIN (150 MIL)	
DIM	MIN	MAX
A IN.	0.188	0.196
MM	4.78	4.98
B IN.	0.150	0.158
MM	3.81	4.01
C IN.	0.048	0.062
MM	1.22	1.57
E IN.	0.004	0.010
MM	0.10	0.25
F IN.	0.053	0.069
MM	1.35	1.75
G IN.	0.050 BSC	
MM	1.27 BSC	
H IN.	0.230	0.244
MM	5.84	6.20
J IN.	0.007	0.011
MM	0.18	0.28
K IN.	0.012	0.020
MM	0.30	0.51
L IN.	0.016	0.050
MM	0.41	1.27
phi	0°	8°

56-G2008-001

ANEXO A8: MODEM

Chapter1

1 Prologue

This document is just suit for the following mode type; it helps you quickly to used M1 Modem function and resolves some common questions.

1.3 Notice

Chapter2

2 Introduction

2.1 Brief

With the development of wireless communication technologies, wireless products are being adopted in numerous industrial and civilian fields. A leader of wireless communication equipment manufacturer releases the M1 Series Wireless Modems, which support various frequency bands of GSM/GPRS/CDMA 1X, and provide industrial terminal solutions for 2.5G/2.75G communication.

M1 series wireless modem adopts industrial level modules, specially designed for the complicated industrial environment which compatible with EMC, and will be your best choose of wireless communication.

2.2 Features

- Industrial design with intelligent software capabilities, making it a reliable cellular solution for data collection and transmission
- Plug-and-play design with easy-to-use software interface for easy integration
- Easily manage and control distributed remote devices over the air
- Built-in Watch Dog
- Real-time Clock (RTC)
- Remote Data Monitor and Control
- Reliable GSM/GPRS/CDMA 1X network connectivity, providing fast and cost-effective long-range wireless communication
- Easy-to-use
- Industrial design with surge protection
- Local and remote configuration over the air
- No need to build expensive fixed line network, saving cost substantially

2.3 Application

- Remote Data Monitor and Control
- Water, gas and oil flow metering
- AMR (automatic meter reading)
- Power station monitoring and control

- Remote POS (point of sale) terminals
- Traffic signals monitor and control
- Fleet management
- Power distribution network supervision
- Central heating system supervision
- Weather station data transmission
- Hydrologic data acquisition
- Vending machine
- Traffic info guidance
- Parking meter
- Telecom equipment supervision (Mobile base station, microwave or optical relay station)
- Oil field data acquisition
- Warehouse supervision

Chapter3

3 Getting Started

3.1 Panel introduction

External (DB9 Interface)

3.2 The LED state

In order to check the module working state. Our product have three Led, pwr LED is power state, Ring LED is Ring state, Data LED is Data state

	PWR	Ring	Data
Start-up	Lights up 3s flashing 0.5s,wink 0.5s ,lights up0.5s	Wink	Lights up 0.5s
Logon network	flashing	Wink	flashing
Sleep state	Lights up 0.5s, wink 0.5s	Wink	wink
date Transfer	Lights up 0.5s, wink 0.5s	Wink	flashing
No date transfer	Lights up 0.5s, wink 0.5s, Lights up 1s	Wink	wink
Voice call	Lights up 0.5s, wink 0.5s	Lights up 1s,	wink

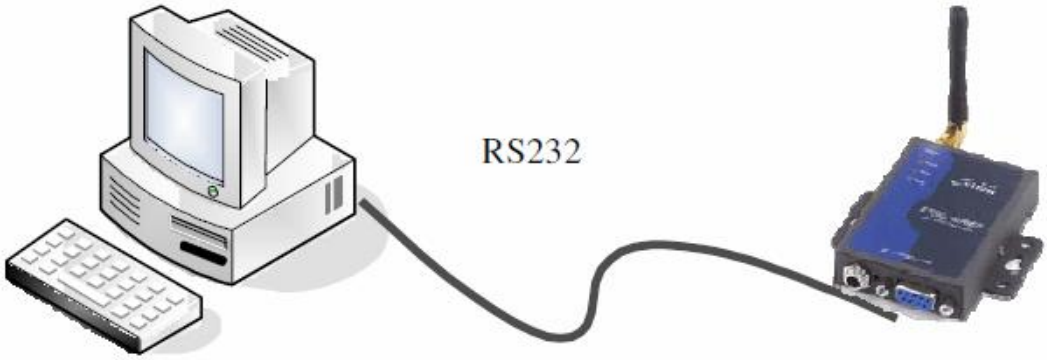
		wink 4s	
reboot	After 5s. wink	wink	wink

3.3 Hardware configuration

- Elevations below
- Back panel as below
- Side panel as below

3.4 Connect to products

1. Please connect antenna and cable with our products, make sure The port is COM1 or COM2?

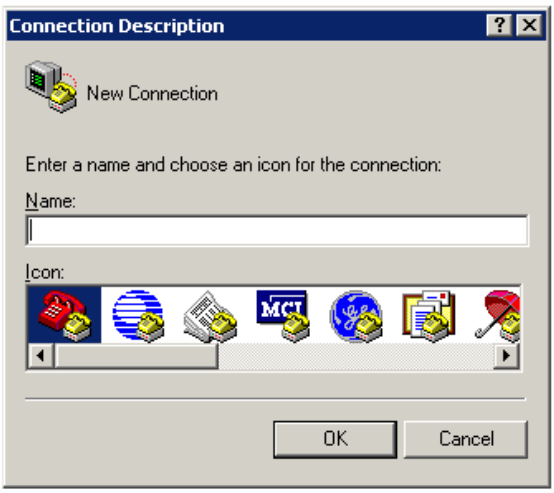


3.5 Insert SIM Card

2. Open the back cover. insert into SIM card as follow

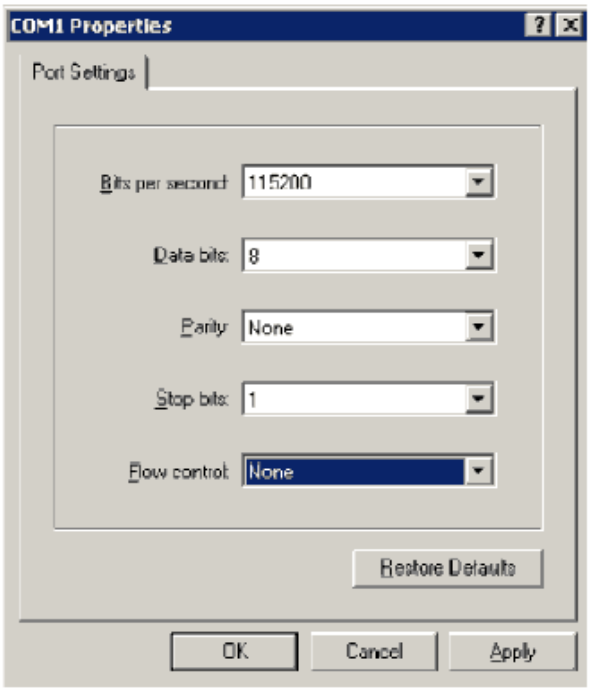
3.6 Note: Hyper Terminal

3. Open the HyperTerminal and input ***(any) as follows



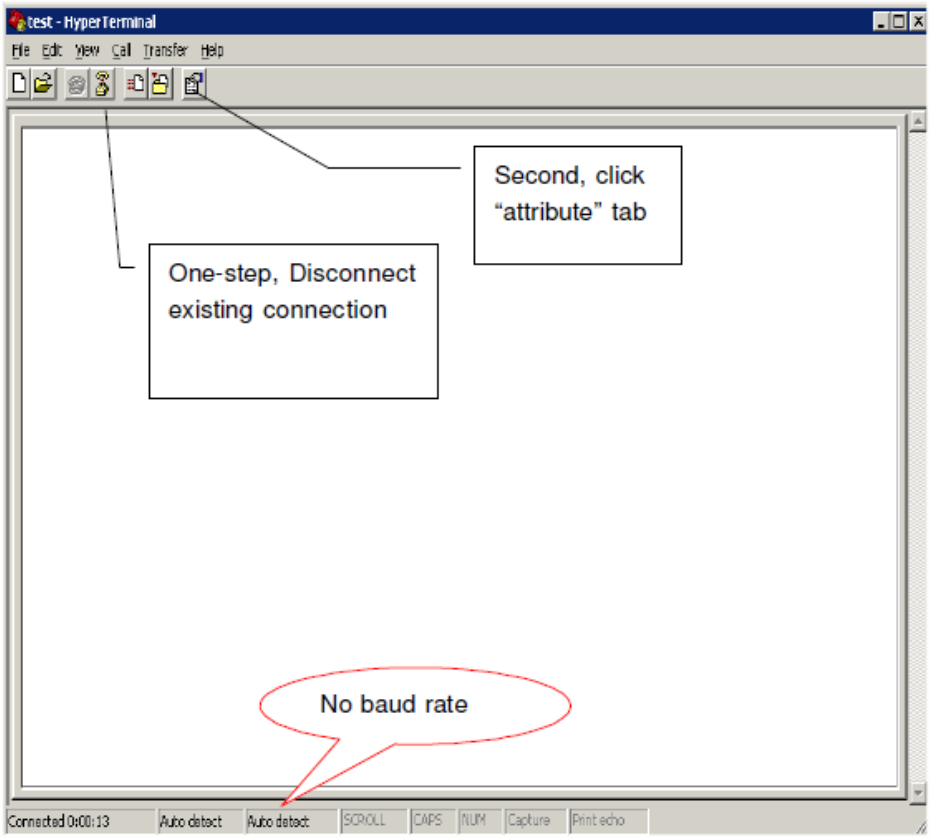
4. Choose a right port

5. The right configuration as following

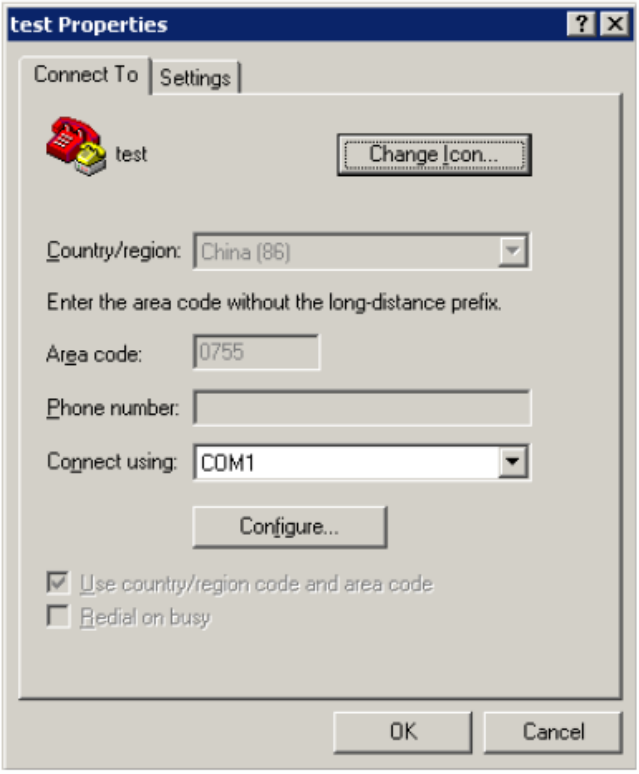


3-3

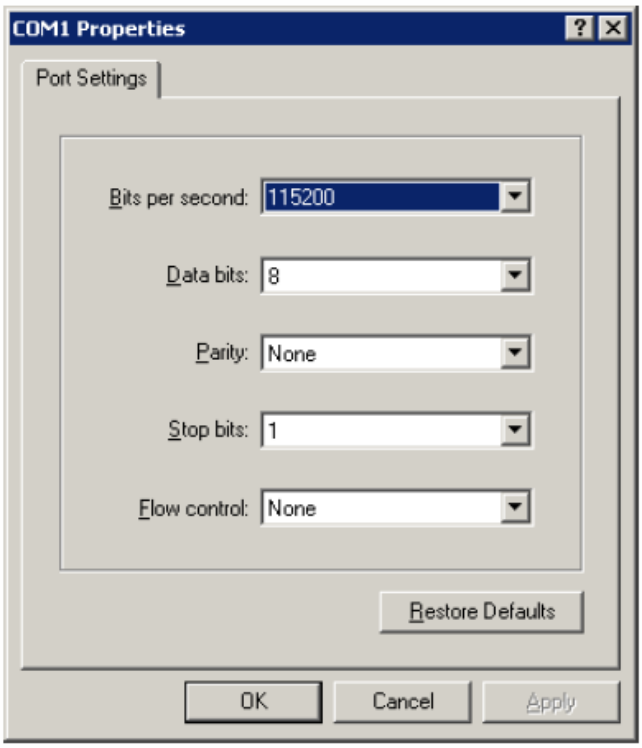
6. When your start-up Hyper Terminal, it is not connected really, you can see the red mark of follow picture without any number .And then, first Disconnect existing connection, second ,Click the red arrowhead



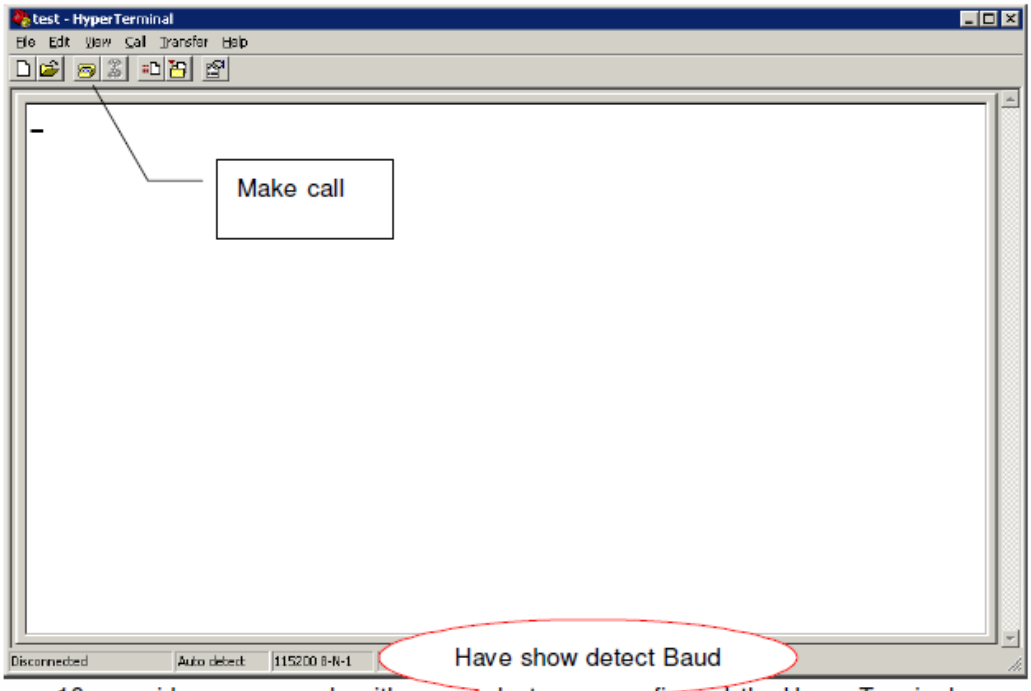
7. click the "configure", and make sure again of you modify configure



8. make sure your modify configure again, click “OK”



9. Then you can see it appeared baud rate on white label, then click the black label to make call



10. provide power supply with our products, you configured the Hyper Terminal successfully

3.7 Test command

Test AT command
AT<CF> //Test “at“command
I/OK //Response ok parameter if successfully connected, you can make sure the module have no malfunction

AT+CSQ<CF> // to check the Signal quality
+CSQ: **, ## // ** Should be the number between 10 and 31, the signal quality becomes better as the number grows. ## should be is 99, Or you should checking the equipment of antenna or SIM card.

If you succeed in testing command, at now, you can begin to SMS

Chapter 5

5 Production list

name	unit	number	description	Sketch-map
Host	Entries	1	Standard supply	
power	Entries	1	Supply 12V	
antenna	Entries	1	Standard supply	
Production-CD	piece	1	Standard supply	

ANEXO A9: DATASHEET SENSOR ULTRASONICO PING

PARALLAX

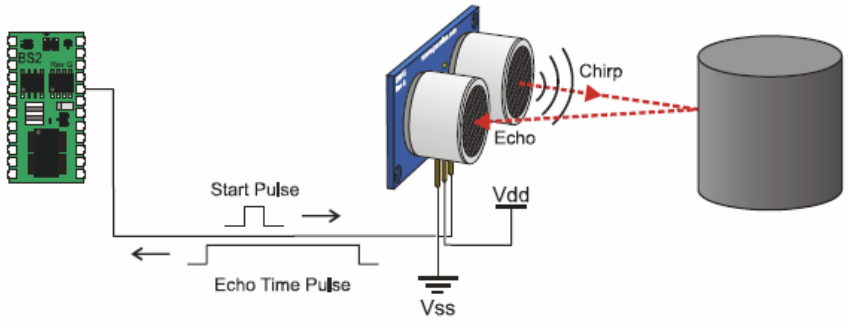
Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: sales@parallax.com
Technical: support@parallax.com

Office: (916) 624-8333
Fax: (916) 624-8003
Sales: (888) 512-1024
Tech Support: (888) 997-8267

PING)))™ Ultrasonic Distance Sensor (#28015)

The Parallax PING))) ultrasonic distance sensor provides precise, non-contact distance measurements from about 2 cm (0.8 inches) to 3 meters (3.3 yards). It is very easy to connect to microcontrollers such as the BASIC Stamp®, SX or Propeller chip, requiring only one I/O pin.

The PING))) sensor works by transmitting an ultrasonic (well above human hearing range) burst and providing an output pulse that corresponds to the time required for the burst echo to return to the sensor. By measuring the echo pulse width, the distance to target can easily be calculated.



Features

- Range: 2 cm to 3 m (0.8 in to 3.3 yd)
- Burst indicator LED shows sensor activity
- Bidirectional TTL pulse interface on a single I/O pin can communicate with 5 V TTL or 3.3 V CMOS microcontrollers
- Input trigger: positive TTL pulse, 2 µs min, 5 µs typ.
- Echo pulse: positive TTL pulse, 115 µs minimum to 18.5 ms maximum.
- RoHS Compliant

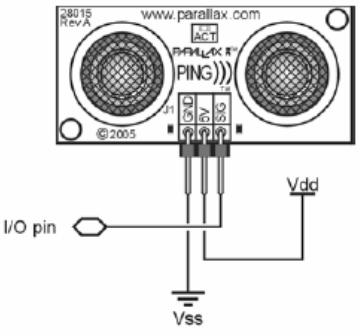
Key Specifications

- Supply voltage: +5 VDC
- Supply current: 30 mA typ; 35 mA max
- Communication: Positive TTL pulse
- Package: 3-pin SIP, 0.1" spacing (ground, power, signal)
- Operating temperature: 0 – 70° C.
- Size: 22 mm H x 46 mm W x 16 mm D (0.84 in x 1.8 in x 0.6 in)
- Weight: 9 g (0.32 oz)

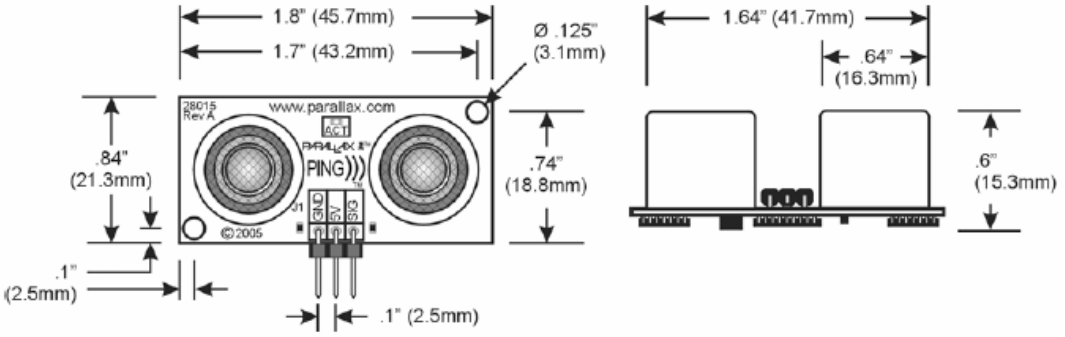
Pin Definitions

GND	Ground (Vss)
5 V	5 VDC (Vdd)
SIG	Signal (I/O pin)

The PING))) sensor has a male 3-pin header used to supply ground, power (+5 VDC) and signal. The header may be plugged into a directly into solderless breadboard, or into a standard 3-wire extension cable (Parallax part #805-000012).

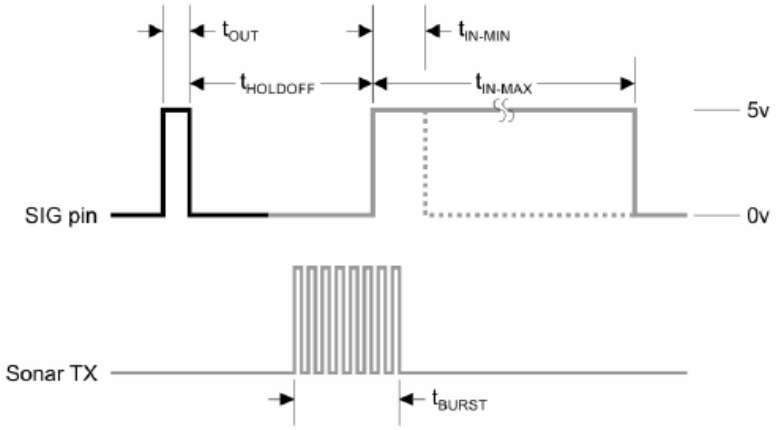


Dimensions



Communication Protocol

The PING))) sensor detects objects by emitting a short ultrasonic burst and then "listening" for the echo. Under control of a host microcontroller (trigger pulse), the sensor emits a short 40 kHz (ultrasonic) burst. This burst travels through the air, hits an object and then bounces back to the sensor. The PING))) sensor provides an output pulse to the host that will terminate when the echo is detected, hence the width of this pulse corresponds to the distance to the target.

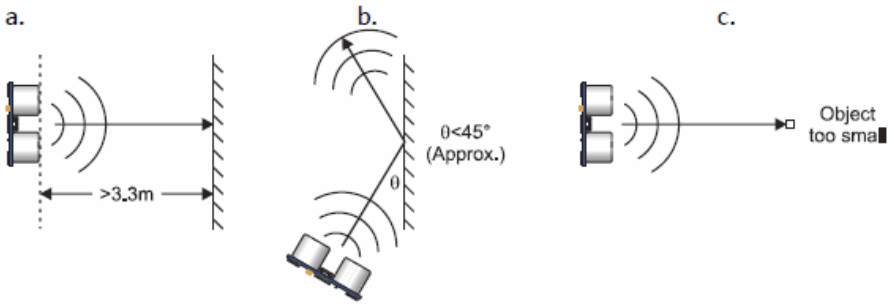


▬	Host Device	Input Trigger Pulse	t_{OUT}	2 µs (min), 5 µs typical
▬	PING))) Sensor	Echo Holdoff	$t_{HOLDOFF}$	750 µs
		Burst Frequency	t_{BURST}	200 µs @ 40 kHz
		Echo Return Pulse Minimum	t_{IN-MIN}	115 µs
		Echo Return Pulse Maximum	t_{IN-MAX}	18.5 ms
		Delay before next measurement		200 µs

Practical Considerations for Use

Object Positioning

The PING))) sensor cannot accurately measure the distance to an object that: a) is more than 3 meters away, b) that has its reflective surface at a shallow angle so that sound will not be reflected back towards the sensor, or c) is too small to reflect enough sound back to the sensor. In addition, if your PING))) sensor is mounted low on your device, you may detect sound reflecting off of the floor.



Target Object Material

In addition, objects that absorb sound or have a soft or irregular surface, such as a stuffed animal, may not reflect enough sound to be detected accurately. The PING))) sensor will detect the surface of water, however it is not rated for outdoor use or continual use in a wet environment. Condensation on its transducers may affect performance and lifespan of the device.

Air Temperature

Temperature has an effect on the speed of sound in air that is measurable by the PING))) sensor. If the temperature (°C) is known, the formula is:

$$C_{air} = 331.5 + (0.6 \times T_C)\ m/s$$

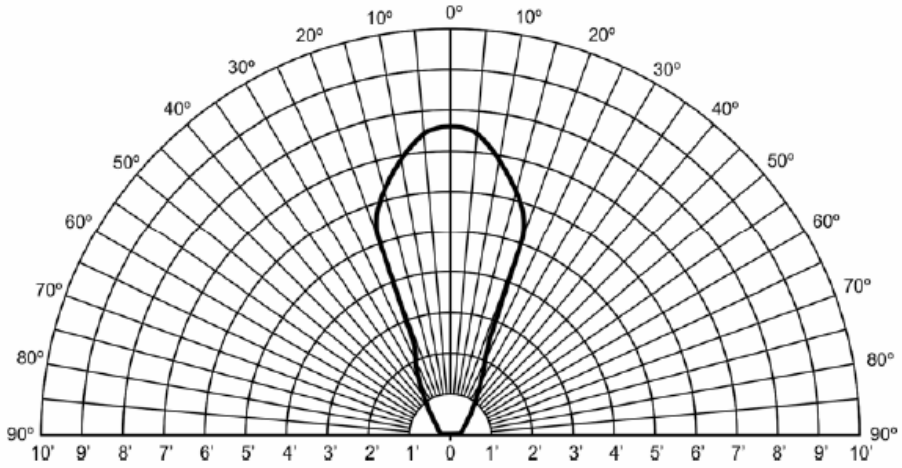
The percent error over the sensor's operating range of 0 to 70 ° C is significant, in the magnitude of 11 to 12 percent. The use of conversion constants to account for air temperature may be incorporated into your program (as is the case in the example BS2 program given in the Example Programs section below). Percent error and conversion constant calculations are introduced in Chapter 2 of *Smart Sensors and Applications*, a Stamps in Class text available for download from the 28029 product page at www.parallax.com.

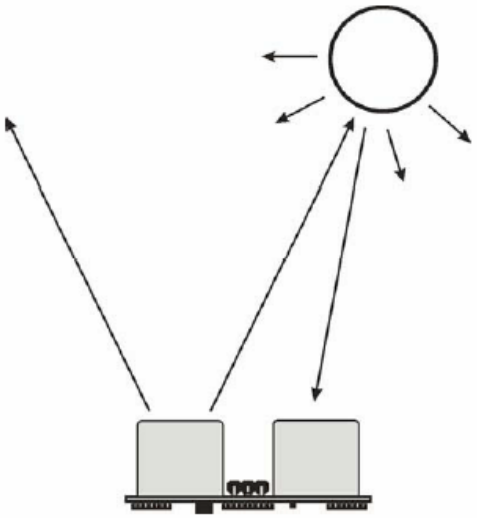
Test Data

The test data on the following pages is based on the PING))) sensor, tested in the Parallax lab, while connected to a BASIC Stamp microcontroller module. The test surface was a linoleum floor, so the sensor was elevated to minimize floor reflections in the data. All tests were conducted at room temperature, indoors, in a protected environment. The target was always centered at the same elevation as the PING))) sensor.

Test 1

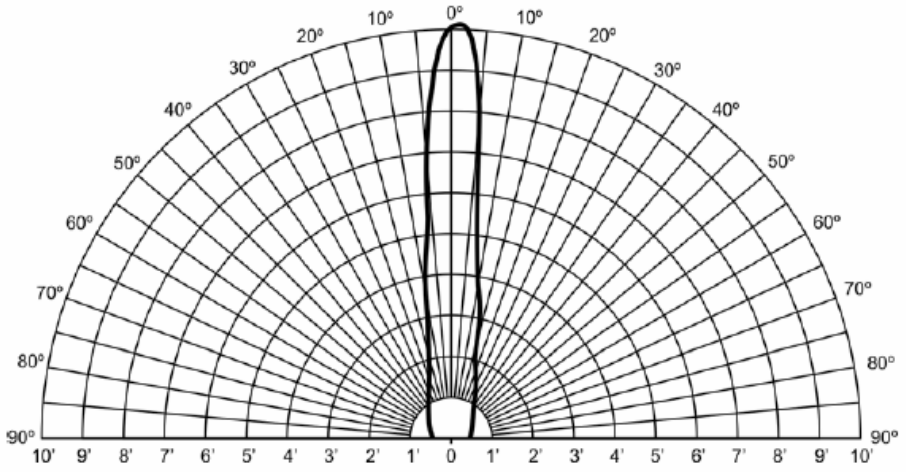
Sensor Elevation: 40 in. (101.6 cm)
Target: 3.5 in. (8.9 cm) diameter cylinder, 4 ft. (121.9 cm) tall – vertical orientation

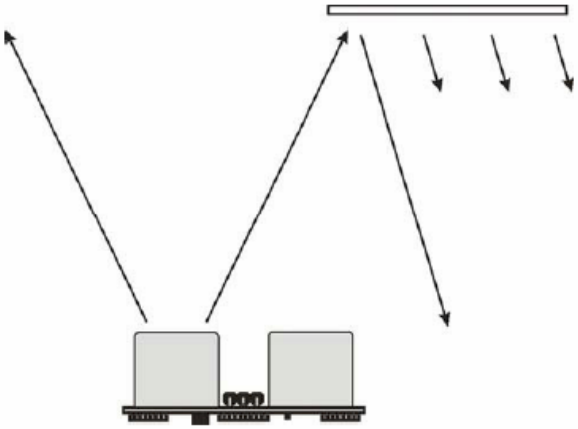




Test 2

Sensor Elevation: 40 in. (101.6 cm)
Target: 12 in. x 12 in. (30.5 cm x 30.5 cm) cardboard, mounted on 1 in. (2.5 cm) pole
Target positioned parallel to backplane of sensor

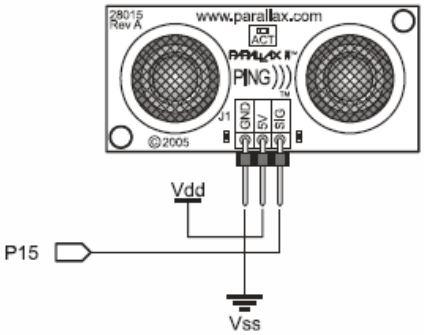


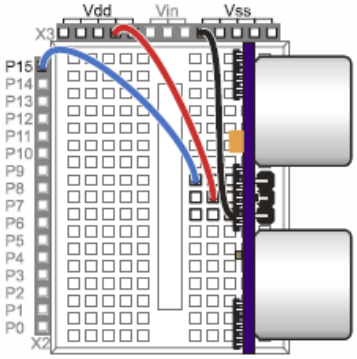


Example Programs and Applications

BASIC Stamp 2

This circuit allows you to quickly connect your PING))) sensor to a BASIC Stamp® 2 via the Board of Education® breadboard area. The PING))) module's GND pin connects to Vss, the 5 V pin connects to Vdd, and the SIG pin connects to I/O pin P15. This circuit will work with the example BASIC Stamp program listed below.

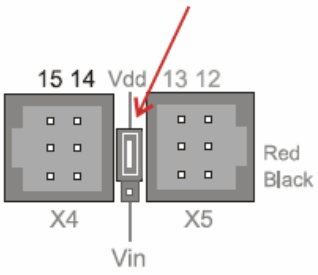




Extension Cable and Port Cautions for the Board of Education

If you are connecting your PING))) sensor to a Board of Education platform using an extension cable, follow these steps:

1. When plugging the cable onto the PING))) sensor, connect Black to GND, Red to 5 V, and White to SIG.
2. Check to see if your Board of Education servo ports have a jumper, as shown at right.
3. If your Board of Education servo ports have a jumper, set it to Vdd as shown. Then plug the cable into the port, matching the wire color to the labels next to the port.
4. If your Board of Education servo ports do not have a jumper, **do not use them with the PING))) sensor**. These ports only provide Vin, not Vdd, and this may damage your PING))) sensor. Go to the next step.
5. Connect the cable directly to the breadboard with a 3-pin header as shown above. Then, use jumper wires to connect Black to Vss, Red to Vdd, and White to I/O pin P15.



Board of Education Servo Port Jumper, Set to Vdd

Example Program: PingMeasureCmAndIn.bs2

This example BS2 program is an excerpt from Chapter 2 of the Stamps in Class text *Smart Sensors and Applications*. Additional PBASIC programs, one for the BS1 and another than runs on any model of BASIC Stamp 2 (BS2, BS2e, BS2sx, BS2p, BS2pe, BS2px) can be downloaded from the 28015 product page.

```
' Smart Sensors and Applications - PingMeasureCmAndIn.bs2
' Measure distance with Ping))) sensor and display in both in & cm

' {$STAMP BS2}
' {$PBASIC 2.5}

' Conversion constants for room temperature measurements.
CmConstant    CON   2260
InConstant    CON   890

cmDistance  VAR   Word
inDistance  VAR   Word
time        VAR   Word

DO

  PULSOUT 15, 5
  PULSIN 15, 1, time

  cmDistance = cmConstant ** time
  inDistance = inConstant ** time

  DEBUG HOME, DEC3 cmDistance, " cm"
  DEBUG CR, DEC3 inDistance, " in"

  PAUSE 100

LOOP
```

Propeller Microcontroller

```
{{
*****************************************
*       Ping))) Object V1.1             *
*     (C) 2006 Parallax, Inc.           *
* Author:  Chris Savage & Jeff Martin   *
* Started: 05-08-2006                   *
*****************************************

Interface to Ping))) sensor and measure its ultrasonic travel time.  Measurements can be in
units of time or distance.  Each method requires one parameter, Pin, that is the I/O pin that
is connected to the Ping)))'s signal line.

  ┌──────────────────┐
  │ ┌──┐  PING)))  ┌──┐ │    Connection To Propeller
  │ │  │           │  │ │    Remember PING))) Requires
  │ └──┘           └──┘ │    +5V Power Supply
  │   GND +5V SIG       │
  └──┬──┬──┬────────────┘
     ⏚  │  ≷ 1K
        │  └ Pin
        ┴

-------------------------REVISION HISTORY-------------------------
 v1.1 - Updated 03/20/2007 to change SIG resistor from 10K to 1K
}}

CON
  TO_IN = 73_746                                                ' Inches
  TO_CM = 29_034                                                ' Centimeters

PUB Ticks(Pin) : Microseconds | cnt1, cnt2
''Return Ping)))'s one-way ultrasonic travel time in microseconds

  outa[Pin]~                                                    ' Clear I/O Pin
  dira[Pin]~~                                                   ' Make Pin Output
  outa[Pin]~~                                                   ' Set I/O Pin
  outa[Pin]~                                                    ' Clear I/O Pin (> 2 µs pulse)
  dira[Pin]~                                                    ' Make I/O Pin Input
  waitpne(0, |< Pin, 0)                                         ' Wait For Pin To Go HIGH
  cnt1 := cnt                                                   ' Store Current Counter Value
  waitpeq(0, |< Pin, 0)                                         ' Wait For Pin To Go LOW
  cnt2 := cnt                                                   ' Store New Counter Value
  Microseconds := (||(cnt1 - cnt2) / (clkfreq / 1_000_000)) >> 1    ' Return Time in µs

PUB Inches(Pin) : Distance
''Measure object distance in inches

  Distance := Ticks(Pin) * 1_000 / TO_IN                         ' Distance In Inches

PUB Centimeters(Pin) : Distance
''Measure object distance in centimeters

  Distance := Millimeters(Pin) / 10                              ' Distance In Centimeters

PUB Millimeters(Pin) : Distance
''Measure object distance in millimeters

  Distance := Ticks(Pin) * 10_000 / TO_CM                        ' Distance In Millimeters
```

Javelin Stamp Microcontroller

This class file implements several methods for using the PING))) sensor with the Javelin Stamp module.

```
package stamp.peripheral.sensor;

import stamp.core.*;


/**
 * This class provides an interface to the Parallax PING))) ultrasonic
 * range finder module.
 * <p>
 * <i>Usage:</i><br>
 * <code>
 *   Ping range = new Ping(CPU.pin0);       // trigger and echo on P0
 * </code>
 * <p>
 * Detailed documentation for the PING))) Sensor can be found at: <br>
 * http://www.parallax.com/detail.asp?product id=28015
 * <p>
 *
 * @version 1.0 03 FEB 2005
 */
public final class Ping {

  private int ioPin;

  /**
   * Creates PING))) range finder object
   *
   * @param ioPin PING))) trigger and echo return pin
   */
  public Ping (int ioPin) {
    this.ioPin = ioPin;
  }


  /**
   * Returns raw distance value from the PING))) sensor.
   *
   * @return Raw distance value from PING)))
   */
  public int getRaw() {

    int echoRaw = 0;

    CPU.writePin(ioPin, false);              // setup for high-going pulse
    CPU.pulseOut(1, ioPin);                  // send trigger pulse
    echoRaw = CPU.pulseIn(2171, ioPin, true); // measure echo return

    // return echo pulse if in range; zero if out-of-range
    return (echoRaw < 2131) ? echoRaw : 0;
  }


  /*
   * The PING))) returns a pulse width of 73.746 uS per inch.  Since the
   * Javelin pulseIn() round-trip echo time is in 8.68 uS units, this is the
   * same as a one-way trip in 4.34 uS units.  Dividing 73.746 by 4.34 we
   * get a time-per-inch conversion factor of 16.9922 (x 0.058851).
   *
```

```
   * Values to derive conversion factors are selected to prevent roll-over
   * past the 15-bit positive values of Javelin Stamp integers.
   */

  /**
   * @return PING))) distance value in inches
   */
  public int getIn() {
    return (getRaw() * 3 / 51);               // raw * 0.058824
  }


  /**
   * @return PING))) distance value in tenths of inches
   */
  public int getIn10() {
    return (getRaw() * 3 / 5);                // raw / 1.6667
  }


  /*
   * The PING))) returns a pulse width of 29.033 uS per centimeter.  As the
   * Javelin pulseIn() round-trip echo time is in 8.68 uS units, this is the
   * same as a one-way trip in 4.34 uS units.  Dividing 29.033 by 4.34 we
   * get a time-per-centimeter conversion factor of 6.6896.
   *
   * Values to derive conversion factors are selected to prevent roll-over
   * past the 15-bit positive values of Javelin Stamp integers.
   */

  /**
   * @return PING))) distance value in centimeters
   */
  public int getCm() {
    return (getRaw() * 3 / 20);               // raw / 6.6667
  }


  /**
   * @return PING))) distance value in millimeters
   */
  public int getMm() {
    return (getRaw() * 3 / 2);                // raw / 0.6667
  }
}
```

This simple demo illustrates the use of the PING))) ultrasonic range finder class with the Javelin Stamp:

```
import stamp.core.*;
import stamp.peripheral.sensor.Ping;


public class testPing {

  public static final char HOME = 0x01;

  public static void main() {

    Ping range = new Ping(CPU.pin0);
    StringBuffer msg = new StringBuffer();

    int distance;
```

```
    while (true) {
      // measure distance to target in inches
      distance = range.getIn();

      // create and display measurement message
      msg.clear();
      msg.append(HOME);
      msg.append(distance);
      msg.append(" \"   \n");
      System.out.print(msg.toString());

      // wait 0.5 seconds between readings
      CPU.delay(5000);
    }
  }
}
```

Resources and Downloads

You can find additional resources for the PING))) sensor by searching the following product pages at www.parallax.com:

- Smart Sensors and Applications (a Stamps in Class text), #28029
- PING))) Mounting Bracket Kit – a servo-driven mount designed to attach to a Boe-Bot robot, #570-28015
- Extension cable with 3-in header, #805-00011 (10-in.) or #805-00012 (14-in.)

A video of a Boe-Bot robot using the PING))) sensor to scan its surroundings then drive to the closest object can be found under Resources > Video Library > Boe-Bot Robot Video Gallery.

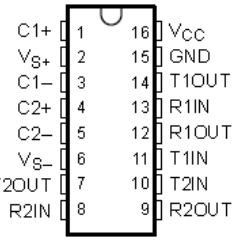
ANEXO A10: DATASHEET MAX232

MAX232, MAX232I
DUAL EIA-232 DRIVERS/RECEIVERS

SLLS047I – FEBRUARY 1989 – REVISED OCTOBER 2002

- Meet or Exceed TIA/EIA-232-F and ITU Recommendation V.28
- Operate With Single 5-V Power Supply
- Operate Up to 120 kbit/s
- Two Drivers and Two Receivers
- ±30-V Input Levels
- Low Supply Current . . . 8 mA Typical
- Designed to be Interchangeable With Maxim MAX232
- ESD Protection Exceeds JESD 22 – 2000-V Human-Body Model (A114-A)
- Applications
  - TIA/EIA-232-F
  - Battery-Powered Systems
  - Terminals
  - Modems
  - Computers

MAX232 . . . D, DW, N, OR NS PACKAGE
MAX232I . . . D, DW, OR N PACKAGE
(TOP VIEW)



description/ordering information

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply EIA-232 voltage levels from a single 5-V supply. Each receiver converts EIA-232 inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V and a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into EIA-232 levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP (N)	Tube	MAX232N	MAX232N
	SOIC (D)	Tube	MAX232D	MAX232
		Tape and reel	MAX232DR	
	SOIC (DW)	Tube	MAX232DW	MAX232
		Tape and reel	MAX232DWR	
	SOP (NS)	Tape and reel	MAX232NSR	MAX232
–40°C to 85°C	PDIP (N)	Tube	MAX232IN	MAX232IN
	SOIC (D)	Tube	MAX232ID	MAX232I
		Tape and reel	MAX232IDR	
	SOIC (DW)	Tube	MAX232IDW	MAX232I
		Tape and reel	MAX232IDWR	

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

Function Tables

EACH DRIVER

INPUT TIN	OUTPUT TOUT
L	H
H	L

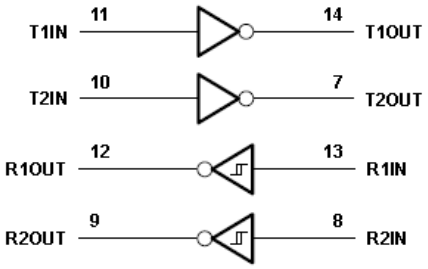
H = high level, L = low level

EACH RECEIVER

INPUT RIN	OUTPUT ROUT
L	H
H	L

H = high level, L = low level

logic diagram (positive logic)



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Input supply voltage range, V_{CC} (see Note 1) . . . . . –0.3 V to 6 V
Positive output supply voltage range, V_{S+} . . . . . V_{CC} – 0.3 V to 15 V
Negative output supply voltage range, V_{S-} . . . . . –0.3 V to –15 V
Input voltage range, V_I: Driver . . . . . –0.3 V to V_{CC} + 0.3 V
Receiver . . . . . ±30 V
Output voltage range, V_O: T1OUT, T2OUT . . . . . V_{S-} – 0.3 V to V_{S+} + 0.3 V
R1OUT, R2OUT . . . . . –0.3 V to V_{CC} + 0.3 V
Short-circuit duration: T1OUT, T2OUT . . . . . Unlimited
Package thermal impedance, θ_{JA} (see Note 2): D package . . . . . 73°C/W
DW package . . . . . 57°C/W
N package . . . . . 67°C/W
NS package . . . . . 64°C/W
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds . . . . . 260°C
Storage temperature range, T_{stg} . . . . . –65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: All voltage values are with respect to network ground terminal.
2. The package thermal impedance is calculated in accordance with JESD 51-7.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	V
V_{IH}	High-level input voltage (T1IN,T2IN)		2			V
V_{IL}	Low-level input voltage (T1IN, T2IN)				0.8	V
R1IN, R2IN	Receiver input voltage				±30	V
T_A	Operating free-air temperature	MAX232	0		70	°C
		MAX232I	–40		85	

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (see Note 3 and Figure 4)

PARAMETER		TEST CONDITIONS	MIN	TYP‡	MAX	UNIT
I_{CC}	Supply current	V_{CC} = 5.5 V, All outputs open, T_A = 25°C		8	10	mA

‡ All typical values are at V_{CC} = 5 V and T_A = 25°C.

NOTE 3: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

MAX232, MAX232I
DUAL EIA-232 DRIVERS/RECEIVERS

SLLS047I – FEBRUARY 1989 – REVISED OCTOBER 2002

DRIVER SECTION

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature range (see Note 3)

PARAMETER			TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage	T1OUT, T2OUT	R_L = 3 kΩ to GND		5	7		V
V_{OL}	Low-level output voltage‡	T1OUT, T2OUT	R_L = 3 kΩ to GND			–7	–5	V
r_o	Output resistance	T1OUT, T2OUT	V_{S+} = V_{S-} = 0,	V_O = ±2 V	300			Ω
I_{OS}§	Short-circuit output current	T1OUT, T2OUT	V_{CC} = 5.5 V,	V_O = 0		±10		mA
I_{IS}	Short-circuit input current	T1IN, T2IN	V_I = 0				200	µA

† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ The algebraic convention, in which the least positive (most negative) value is designated minimum, is used in this data sheet for logic voltage levels only.
§ Not more than one output should be shorted at a time.
NOTE 3: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Note 3)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
SR	Driver slew rate	R_L = 3 kΩ to 7 kΩ, See Figure 2			30	V/µs
SR(t)	Driver transition region slew rate	See Figure 3		3		V/µs
	Data rate	One TOUT switching		120		kbit/s

NOTE 3: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

RECEIVER SECTION

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature range (see Note 3)

PARAMETER			TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage	R1OUT, R2OUT	I_{OH} = –1 mA		3.5			V
V_{OL}	Low-level output voltage‡	R1OUT, R2OUT	I_{OL} = 3.2 mA				0.4	V
V_{IT+}	Receiver positive-going input threshold voltage	R1IN, R2IN	V_{CC} = 5 V,	T_A = 25°C		1.7	2.4	V
V_{IT-}	Receiver negative-going input threshold voltage	R1IN, R2IN	V_{CC} = 5 V,	T_A = 25°C	0.8	1.2		V
V_{hys}	Input hysteresis voltage	R1IN, R2IN	V_{CC} = 5 V		0.2	0.5	1	V
r_i	Receiver input resistance	R1IN, R2IN	V_{CC} = 5,	T_A = 25°C	3	5	7	kΩ

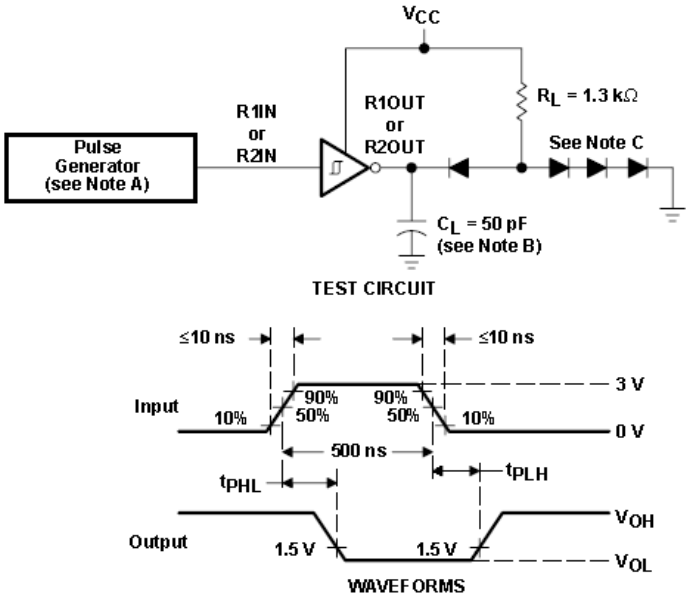
† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ The algebraic convention, in which the least positive (most negative) value is designated minimum, is used in this data sheet for logic voltage levels only.
NOTE 3: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Note 3 and Figure 1)

PARAMETER		TYP	UNIT
$t_{PLH(R)}$	Receiver propagation delay time, low- to high-level output	500	ns
$t_{PHL(R)}$	Receiver propagation delay time, high- to low-level output	500	ns

NOTE 3: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

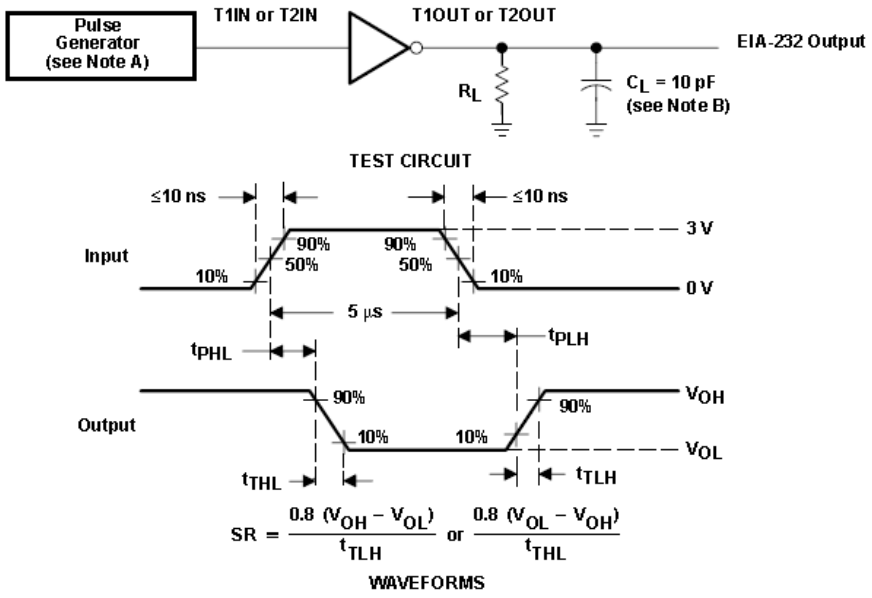
PARAMETER MEASUREMENT INFORMATION



NOTES: A. The pulse generator has the following characteristics: Z_O = 50 Ω, duty cycle ≤ 50%.
B. C_L includes probe and jig capacitance.
C. All diodes are 1N3064 or equivalent.

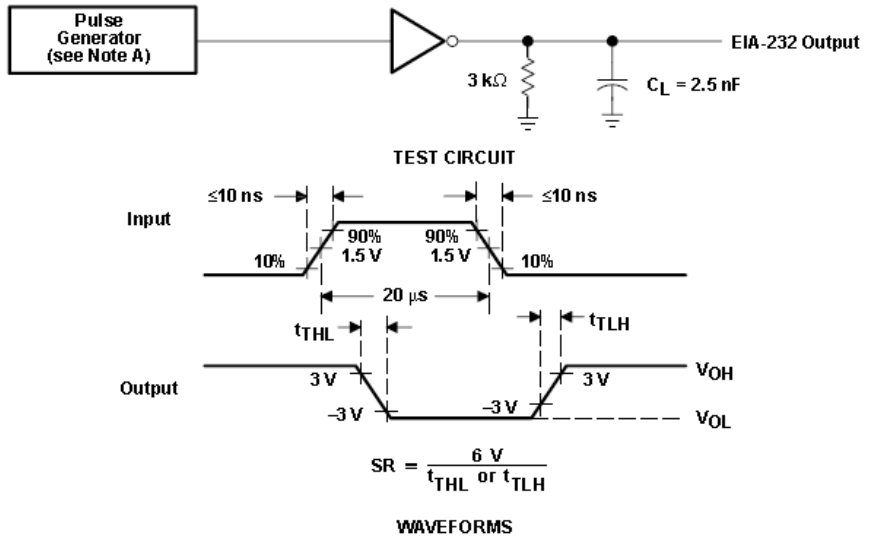
Figure 1. Receiver Test Circuit and Waveforms for t_{PHL} and t_{PLH} Measurements

PARAMETER MEASUREMENT INFORMATION



NOTES: A. The pulse generator has the following characteristics: Z_O = 50 Ω, duty cycle ≤ 50%.
B. C_L includes probe and jig capacitance.

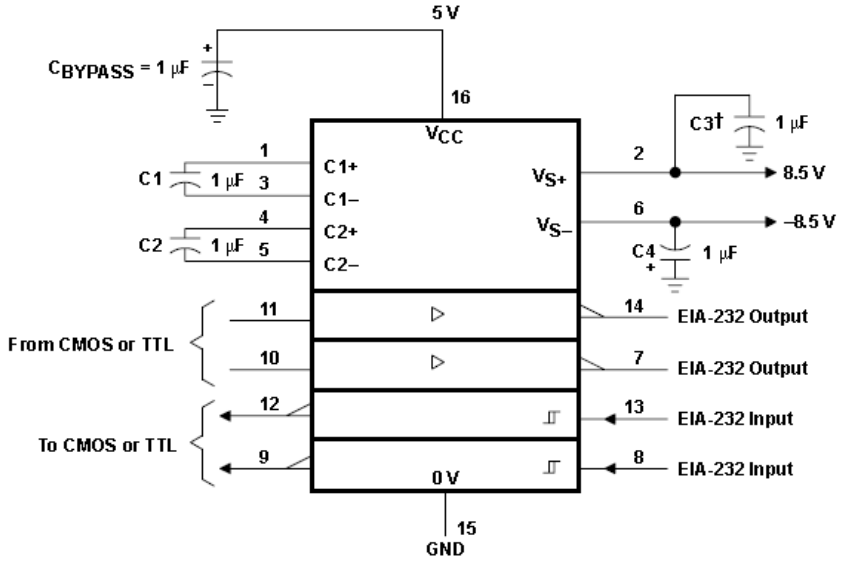
Figure 2. Driver Test Circuit and Waveforms for t_{PHL} and t_{PLH} Measurements (5-µs Input)



NOTE A: The pulse generator has the following characteristics: Z_O = 50 Ω, duty cycle ≤ 50%.

Figure 3. Test Circuit and Waveforms for t_{THL} and t_{TLH} Measurements (20-µs Input)

APPLICATION INFORMATION



† C3 can be connected to V_{CC} or GND.

Figure 4. Typical Operating Circuit

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

REFERENCIAS BIBLIOGRÁFICAS

[1] Figueroa de la Cruz, Mario, Introducción a los sistemas de Telefonía Celular, Primera Edición, Editorial Hispano Americana HASA, Buenos Aires 2008.

[2] Huidrobo Maya, Jose Manuel, Tecnologías de Telecomunicaciones, Primera Edición, Alfa Omega Grupo Editor, Mexico 2000.

[3] Le Bodic, Gwenal, Mobile Messaging Technologies and Service, Segunda Edición, Editorial Wiley, Chichester 2005.

[4] Hillar, Gastón, Visual Basic 2005 y Net 2.0: curso intensivo, Primera Edición, Editorial Hispano Americana HASA, Buenos Aires 2007.

[5] Bus I2C, http://www.comunidadelectronicos.com/articulos/i2c.htm, 03-05-2010

[6] Lenguaje SQL, http://personal.lobocom.es/claudio/sql001.htm, 20-03-2010

[7] Datasheet DS1307, http://www.datasheetcatalog.com/datasheets_pdf/D/S/1/3/DS1307.shtml, 25-03-2010

[8] Sensor de Nivel Ultrasónico PING, http://www.parallax.com/Portals/0/Downloads/docs/prod/.../28015-PING-v1.6.pdf, 10-07-2010

[9] Modem ZTE MG3006, http://www.forwellwireless.com/en/GPRS_Modem_specification.html, 05-03-2010

[10] Modem ZTE MG3006, http://www.forwellwireless.com/download/M1_Usermanual_Eng.pdf, 05- 03-2010

[11] Comandos AT (Modem ZTE MG3006), http://download.maritex.com.pl/pdfs/wi/AT%20ZTE.pdf,10-03-2010

[12] Datasheet ATMEGA8, http://www.datasheetcatalog.net/es/datasheets_pdf/A/T/M/E/ATMEGA8.shtml, 05-02-2010

[13] Microsoft .Net 2005, http://msdn.microsoft.com/es-ec/default.aspx, 10-04-04-2010

[14] Crystal Report, http://www.wiener.edu.pe/manuales2/5to-ciclo/PROGRAMACION-VISUAL-4/Manual-Crystal-Reports-11-XI.pdf, 15-05-2010

[15] Microsotf SQL Server 2000, http://www.snip.gob.ni/Xdc/SQL/SqlServer.htm, 20-05-2010

[16] Compilador BASCOM, http://www.avrhelp.mcselec.com/, 05-01-2010

[17] Ordeño Mecánico, http://ocw.upm.es/produccion-animal/ordeno-mecanico/Tema_2._ORDENO_MECANICO._FUNDAMENTOS_Y_COMPONENTES/tema_02-

2_componentes_basicos_de_una_instalacion_de_ordeno_mecanico.pdf,20-12-2009

[18] Ordeño Mecánico, http://vaca.agro.uncor.edu/~pleche/material/Material%20II/A%20archivos%20internet/Maquinainstala/cap5.pdf

[19] Datasheet MAX232, http://www.datasheetcatalog.net/es/datasheets_pdf/M/A/X/2/MAX232.shtml,05-02-2010

Sangolquí, 25 de Agosto del 2010

_____	_____
Carlos Merchán	Ing. Gonzalo Olmedo
AUTOR	COORDINADOR DE CARRERA