UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERIA Y ARQUITECTURA

ESCUELA DE INGENIERIA ELECTRICA


# "Diseño y construcción de un módulo electrónico programable vía Ethernet desde la Web, para monitorear, controlar y administrar energía en aplicaciones comerciales"

PRESENTADO POR:

**GUSTAVO GOMEZ PEREZ**

**WILMAN ROBERTO VILLATORO RIOS**

PARA OPTAR AL TITULO DE:

**INGENIERO ELECTRICISTA**

CIUDAD UNIVERSITARIA, OCTUBRE DE 2007

# UNIVERSIDAD DE EL SALVADOR

RECTORA :

**DRA. MARÍA ISABEL RODRÍGUEZ**

SECRETARIA GENERAL:

**LICDA. ALICIA MARGARITA RIVAS DE RECINOS**

# FACULTAD DE INGENIERIA Y ARQUITECTURA

DECANO :

**ING. MARIO ROBERTO NIETO LOVO**

SECRETARIO :

**ING. OSCAR EDUARDO MARROQUÍN HERNÁNDEZ**

# ESCUELA DE INGENIERIA ELECTRICA

DIRECTOR :

**ING. LUIS ROBERTO CHÉVEZ PAZ**

UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERIA Y ARQUITECTURA
ESCUELA DE INGENIERIA ELECTRICA

Trabajo de Graduación previo a la opción al Grado de:

**INGENIERO ELECTRICISTA**

Título :

# "Diseño y construcción de un módulo electrónico programable vía Ethernet desde la Web, para monitorear, controlar y administrar energía en aplicaciones comerciales"

Presentado por :

**GUSTAVO GOMEZ PEREZ**
**WILMAN ROBERTO VILLATORO RIOS**

Trabajo de Graduación Aprobado por:

Docente Director :

Ing. Hugo Miguel Colato Rodríguez

San Salvador, Octubre de 2007

Trabajo de Graduación Aprobado por:

Docente Director:

**Ing. Hugo Miguel Colato Rodríguez**

## ACTA DE CONSTANCIA DE NOTA Y DEFENSA FINAL

**En esta fecha, 23 de Octubre de 2007, en La Sala de Lectura de esta Escuela, a las diecisiete horas, en presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:**



**Firma:**

1. **Ing. Luis Roberto Chévez Paz**
   **Director**

2. **Ing. Gerardo Marvin Hernández**
   **Secretario**

**Y, con el Honorable Jurado de Evaluación integrado por las personas siguientes:**

**Firma:**

1. **Ing. Roberto Eduardo Saravia Gutiérrez**

2. **Ing. Carlos Osmin Pocasangre Jiménez**

**Se efectuó la defensa final reglamentaria del Trabajo de Graduación:**

**"Diseño y construcción de un módulo electrónico programable vía Ethernet desde la Web, para monitorear, controlar y administrar energía en aplicaciones comerciales"**

**A cargo de los Bachilleres:**

**GÓMEZ PÉREZ, GUSTAVO**
**VILLATORO RÍOS, WILMAN ROBERTO**

**Habiendo obtenido el presente Trabajo una nota final, global de:** 9.0

( nueve punto cero )

# AGRADECIMIENTOS

Agradezco a Dios todo poderoso, por la vida y la salud, por acompañarme siempre y darme fortaleza en todos los momentos de mi formación académica, haciendo posible culminar mi carrera universitaria.

Agradezco y dedico esta obra, a mis padres Balbino de Jesús Gómez y Deysi Pérez, quienes se han desgastado en esfuerzo y sacrificio sin esperar nada a cambio para apoyarme a mi formación profesional, es el mayor regalo que me han dado, en mi vida no les fallaré. Ojala algún día pueda lograr ser una parte de lo que ahora son ustedes.

A mis Abuelos, Alfonso Gómez (Q.D.D.G) y Mercedes Benítez, gracias por preguntarme siempre ¿Y como saliste?, mostrándome su apoyo y cariño. Este logro es el resultado de tus peticiones fe, esperanza y caridad, gracias Abuela.

A mis hermanos Dilver Adín Gómez Pérez, Roberto Carlos Gómez Pérez, Sergio de Jesús Gómez Pérez y Mayra Milady Gómez Pérez, por su amistad, apoyo invaluable y sobre todo porque hemos sido todos para uno y uno para todos. Gracias mi Familia del alma Gómez Pérez.

A mi prima Yanira Portillo por el apoyo incondicional y motivación para llegar hasta donde estoy. También agradezco a mi cuñada Lilian Gutiérrez, por su colaboración valiosa y animación a seguir en la lucha hasta cumplir esta meta.

A todos mis familiares, que siempre me han dado su apoyo moral en este proceso, y a mi compañero de formula Wilman Roberto Villatoro Ríos por compartir alegrías y sacrificio en la realización de este trabajo de graduación.

A nuestro asesor Ing. Hugo Miguel Colato, por haber creído en nosotros y aportarnos, tiempo, orientación y dedicación.

A mis profesores y laboratoristas que compartieron sus conocimientos con migo.

A mis compañeros con quienes me desvele estudiando, compartí momentos buenos y malos durante mi vida universitaria.

A mis amigos y personas que de una u otra manera han contribuido, ha que hoy sea toda una realidad este proyecto.

Al Padre José Catalino Henríquez por su apoyo moral y espiritual incondicional.

A mi gente del Cantón La Fragua, Moncagua, San Miguel, de donde orgullosamente soy originario, mi fortaleza ha sido la suma de todo el apoyo moral y espiritual que he recibido. Gracias mi gente.

**Gustavo Gómez Pérez**

# AGRADECIMIENTOS

Estoy convencido de que me habría sido imposible alcanzar la culminación de mis estudios sin el apoyo y la motivación que recibí de familiares y amigos que de una u otra forma contribuyeron en mi formación.

En primer lugar quiero agradecer a ese par de ángeles que estuvieron a mi lado todo el tiempo, esforzándose hasta el cansancio pero sin desmayar, con el único propósito de darme la oportunidad de lograr una carrera universitaria, ellos son mis padres Efraín Villatoro y Josefina Ríos, a quienes les debo todo lo que tengo y lo que soy, les agradezco por haber creído en mí en todo momento, porque aunque hubo momentos en que todo parecía imposible, nunca se dieron por vencidos, siempre con la convicción de que la educación es el mejor regalo que se le puede dar a un hijo. Le pido a Dios que me ayude a recordar siempre ese esfuerzo que mis padres hicieron, que aunque con muchas dificultades siempre me dieron su apoyo, espero que cuando tenga mis hijos llegue a comprender completamente el amor y la dedicación que ellos han tenido para conmigo.

Agradezco a mis hermanas por el apoyo y comprensión que en todo momento me dieron a lo largo de los años que me dediqué a estudiar, siempre fueron mi motivación para poder seguir.

Quiero expresar mi más sincero agradecimiento a mi tío German Villatoro, por el apoyo incondicional que siempre me brindó, por sus consejos y palabras de aliento, y por haber estado ahí siempre que lo necesité.

Le doy gracias a todos aquellos parientes y amigos que de una u otra forma contribuyeron para que yo pudiera lograr este titulo, siempre se los voy a agradecer y confío en que todos tendrán su recompensa.

**Wilman Roberto Villatoro Rios.**

# PREFACIO

Una de las situaciones que se da muy a menudo para que exista alto costo de energía eléctrica dentro de una instalación, es la de tener activos algunos servicios eléctricos aun cuando el usuario no los necesita; esto se debe a que el controlador del recurso es manual y depende estrictamente de la imprudencia del mismo usuario, el descontrol se incrementa cuando el usuario es un grupo numeroso de personas donde el uso del servicio es ocasional y nadie es responsable de la administración del servicio.

En cambio, la instalación de sistemas de control automatizado e inteligentes permite al usuario administrar de mejor manera las instalaciones, a la vez que se logra un ahorro significativo de costo de energía, ya que con ello se tendría absoluto control sobre los servicios.

Actualmente existe en el mercado la tecnología necesaria para desarrollar sistemas de control y monitoreo eficientes y a precios razonables, esto ha motivado la creación del sistema presentado en este trabajo de graduación, el cual tiene como carta de presentación un balance entre funcionalidad versus costos de fabricación.

Este sistema tiene como base dos aspectos fundamentales para su construcción y funcionamiento, los cuales son el ahorro de energía, uso de protocolos estándares en la industria y el uso de software de libre distribución en su construcción

En presente trabajo de graduación se muestra el diseño y funcionamiento de un sistema electrónico programable vía Web, el cual sirve para controlar de forma automatizada aplicaciones eléctricas que normalmente requieran de encendido y apagado manual. Con este sistema se pretende ayudar a reducir los costos en el uso de la energía eléctrica en aplicaciones eléctricas comerciales y residenciales, evitando el uso inadecuado administrando el recurso.

# RESUMEN DEL TRABAJO

Este documento presenta el desarrollo del diseño y construcción de un módulo electrónico programable vía ethernet desde la web, para monitorear, controlar y administrar energía eléctrica en aplicaciones comerciales, que en adelante llamaremos SACME[1] con la principal característica de poder administrar recursos que normalmente requieran de encendido y apagado tal como son las luminarias de un edificio, institución o residencia simplemente programando eventos desde cualquier equipo que tenga acceso al sistema a través de un navegador de páginas web.

El módulo electrónico provee las conexiones necesarias para administrar el funcionamiento de al menos 160 dispositivos eléctricos, cuenta con las funciones necesarias para ser utilizado por otros sistemas en forma de esclavo, para ello ha sido dotado de un servidor Modbus, el cual puede ser gestionado por cualquier cliente Modbus, ya sea comercial o embebido en el sistema tal como es el caso de una aplicación en LabView de Nacional Instrument.

Se ha desarrollado una aplicación Web para manejar al módulo electrónico y dotar al usuario de una interfase sencilla y funcional. El sistema reconoce tres tipos de usuario: Administrador, Registrado e Invitado, cada uno de ellos tiene un rol dentro del sistema, estos roles son los que limitan los recursos a los que tiene acceso cada usuario, es responsabilidad del administrador darle mantenimiento y asignarle dichos roles a cada usuario.

Capítulo I, se refiere a la teoría básica que está relacionada con el protocolo de comunicación Modbus sobre la red ethernet que es la base sobre la cual se desarrolla tanto el software como el hardware, además se hace un bosquejo de sistemas existentes en el mercado, similares al desarrollado en el presente trabajo de graduación.

Capítulo II, esta orientado al diseño y la implementación del hardware que el sistema desarrollado requiere, aquí se presentan los criterios de diseño y la tecnología utilizada, también se desarrolla el software que implementa un esclavo del protocolo modbus para hacer que el hardware tenga aplicación en uso comercial.

Capítulo III, se presenta el diseño y la implementación del sitio web que da soporte a recursos remotamente desde un navegador teniendo en cuenta aspectos como la portabilidad, escalabilidad y uso de motor de persistencia para lograr independencia del gestor de bases de datos.

Capítulo IV, aquí se proponen algunas mejoras al sistema que en todo el documento se ha desarrollado para que sea mas robusto y logre ventajas con los sistemas existentes en el mercado.

[1] SACME, Sistema de Administración, Control y Monitoreo de Energía Eléctrica

## TABLA DE CONTENIDOS

**Capitulo** **Página**

## LISTA DE TABLAS

**Tabla** **Página**

## LISTA DE FIGURAS

**Figura** **Página**

# OBJETIVOS

## Objetivo General

Mostrar el diseño y construcción de un sistema de control, monitoreo y administración de energía eléctrica en aplicaciones comerciales, el cual es programado vía ethernet desde un sitio Web creado especialmente para este propósito, o también, desde un programa comercial basado en el mismo protocolo estándar bajo el que ha sido creado el módulo electrónico de este sistema.

## Objetivos Específicos

- Mostrar el diseño y la implementación de un módulo electrónico programable cuya aplicación es la servir de interfase entre los servicios eléctricos y la aplicación Web que el usuario utiliza para el control y monitoreo.

- Mostrar el diseño y la implementación de la aplicación Web utilizada para la administración del sistema, la programación del módulo electrónico así como también el control instantáneo de los servicios eléctricos a cargo del sistema.

- Especificar completamente la tecnología utilizada en la implementación del sistema para determinar los costos económicos derivados de ello.

- Documentar adecuadamente el sistema para su fácil entendimiento y utilización en cualquier aplicación de energía que cuente con las condiciones que este sistema requiere para operar y ser funcional.

# CAPITULO I

# MARCO TEÓRICO

## Introducción.

En el presente capitulo trata algunos aspectos que han sido utilizados en el desarrollo del trabajo de graduación, tales como el protocolo Modbus, sistemas similares en el mercado y otros conceptos que son de importancia para el buen entendimiento y la implementación del sistema sacme.

## 1.1 Protocolo Modbus/TCP y Modbus

Modbus/TCP es un protocolo de comunicación diseñado para permitir a equipo industrial tal como Controladores Lógicos Programables (PLCs), computadores, motores, sensores, medidores y otros tipos de dispositivos físicos de entrada/salida que puedan comunicarse sobre una red. Modbus/TCP. Fue introducido por Schneider Automation como una variante de la familia MODBUS ampliamente usada, siendo protocolos de comunicación simples y abiertos, destinados para la supervisión y el control de equipo de automatización. Específicamente, el protocolo cubre el uso de mensajes MODBUS en un entorno Intranet o Internet usando los protocolos TCP/IP.

La especificación Modbus/TCP define un estándar inter-operable en el campo de la automatización industrial, el cual es mas sencillo de implementar para cualquier dispositivo que soporta sockets TCP/IP.

### 1.1.1 Concepto de Modbus

MODBUS/TCP es un protocolo de mensajería que forma parte de la capa de aplicación, posicionado en la capa 4 del modelo TCP/IP (7 del modelo OSI), que provee comunicación cliente/servidor entre dispositivos conectados sobre diferentes tipos de buses de redes. El protocolo MODBUS/TCP define una estructura de mensajería que los dispositivos reconocerán y usaran, en general sobre cualquier tipo de red, pero para nuestro caso sobre una red Ethernet. Dicho protocolo describe el proceso que un dispositivo usa para realizar peticiones hacia otros dispositivos, ¿cómo este responde a las peticiones de otros dispositivos, y cómo los errores serán detectados y reportados?. Básicamente se establece un formato común para la estructura y contenido del campo de mensaje.

Durante la comunicación sobre una red MODBUS, el protocolo determina la forma en que cada dispositivo reconocerá sus direcciones, reconocerá un mensaje direccionado a éste, determinara el tipo de acción a ser tomada, y extraerá cualquier dato u otra información contenida en el mensaje. Si una replica es requerida, el controlador construirá el mensaje replica y lo enviara usando el protocolo MODBUS.

Cuando MODBUS viaja sobre otras redes, los mensajes conteniendo el protocolo MODBUS son embebidos dentro de la trama o estructura de paquete que es usado en la nueva red. Lo cual permite "hablar un mismo idioma" sobre medios que utilizan protocolos diferentes.

### 1.1.2 Modelo Cliente-Servidor

El servicio de mensajería MODBUS provee comunicación Cliente/Servidor entre dispositivos conectados sobre una red Ethernet. El modelo cliente/servidor es basado sobre cuatro tipos de mensajes:

• Petición MODBUS
• Confirmación MODBUS
• Indicación MODBUS
• Respuesta MODBUS

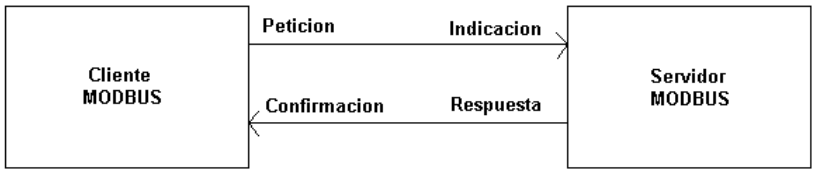


**Figura 1. 1** Mensajes en el Modelo Cliente-Servidor de Modbus.

**Petición MODBUS**
Como se muestra en la figura 1.1, es el mensaje enviado sobre la red por el cliente para inicializar una transacción.

**Indicación MODBUS**
Es básicamente el mensaje de petición hecho por el cliente pero ahora recibido en el lado del servidor.

**Respuesta MODBUS**
Es el mensaje de respuesta enviado por el servidor.

**Confirmación MODBUS**
Es el mensaje de respuesta recibido en el lado del cliente.

Este servicio de mensajería MODBUS es utilizado en intercambios de información en tiempo real.

### 1.1.3 Trama Modbus sobre TCP/IP

Aquí se muestra la forma en que las peticiones y respuestas MODBUS se encapsula cuando son llevadas a través de redes MODBUS TCP/IP.

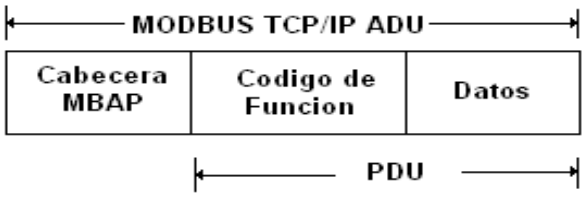


**Figura 1. 2** Trama del Protocolo Modbus sobre TCP/IP.

Como se muestra en la figura 1.2, la trama se divide en tres secciones principales:

**Cabecera MBAP**
Sobre TCP/IP una cabecera es utilizada para identificar la Unidad de Dato de Aplicación (ADU). La cual es llamada Cabecera MBAP (Cabecera del protocolo de aplicación MODBUS). La cabecera MODBUS contiene cuatro campos que se describen el la tabla 1.1, la cabecera tiene 7 bits de longitud.

**Identificador de transacción**
Este es usado en pares de transacciones, el servidor MODBUS copia en la respuesta el identificador de transacción de la petición.

**Identificador de Protocolo**
El protocolo MODBUS es identificado por el valor cero.

**Tabla 1. 1** Cabecera MBAP del Protocolo Modbus sobre TCP/IP.

| Campo | Longitud | Descripción | Cliente | Servidor |
|---|---|---|---|---|
| Identificador de Transacción | 2 Bits | Identificación de una transacción petición/respuesta en Modbus | Inicializada por el cliente | Recopiado por el servidor desde la petición recibida. |
| Identificador de protocolo | 2 Bits | 0 = Protocolo MODBUS | Inicializado por el cliente | Recopiado por el servidor desde la petición recibida. |
| Longitud | 2 Bits | Numero de bites siguientes | Inicializado por el cliente (petición) | Inicializado por el servidor (respuesta) |
| Identificador de unidad | 1 Bit | Identificación de un esclavo remoto conectado sobre una línea serial u otro bus | Inicializado por el cliente | Recopiado por el servidor desde la petición recibida. |

### 1.1.4 Cliente Modbus

El cliente MODBUS le permite a la aplicación usuario explícitamente controlar el intercambio de información con un dispositivo remoto. El cliente MODBUS construye una petición MODBUS de los parámetros contenidos en una demanda enviada por la aplicación usuario a la interfaz cliente MODBUS.

El cliente MODBUS usa una transacción MODBUS cuya administración incluya espera y procesado de una confirmación MODBUS.

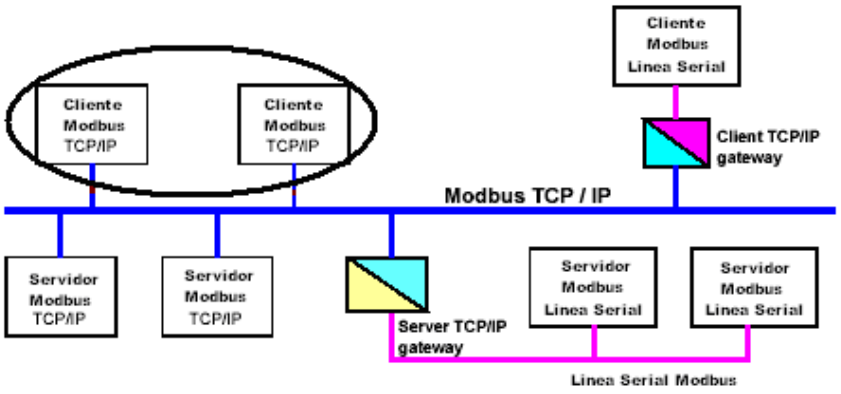


**Figura 1. 3** Unidad cliente Modbus.

**Construcción de una petición MODBUS**

Seguido de la recepción de una demanda desde la aplicación usuario, el cliente tiene que construir una petición MODBUS y enviarla al administrador TCP.

Construir la petición MODBUS puede ser dividido en varias sub-tareas:
La creación de la instancia de una transacción MODBUS que le permita al cliente memorizar toda la información requerida para luego asociar la respuesta a la petición y enviar la confirmación a la aplicación usuario.

El codificado de la petición MODBUS (PDU+cabecera MPAB). La aplicación usuario que inicializa la demanda tiene que proveer toda la información requerida que permita al cliente codificar la petición. El PDU MODBUS es codificado de acuerdo a el "MODBUS Application Protocol Specification [1]". (Código de función MODBUS, parámetros asociados y datos de aplicación). Todos los campos de la cabecera MBAP son llenados. Entonces el ADU de la petición MODBUS es construido prefijando el PDU con la cabecera MBAP.

Los siguientes diagramas de actividades describen más profundamente la fase de construcción de la petición MODBUS.

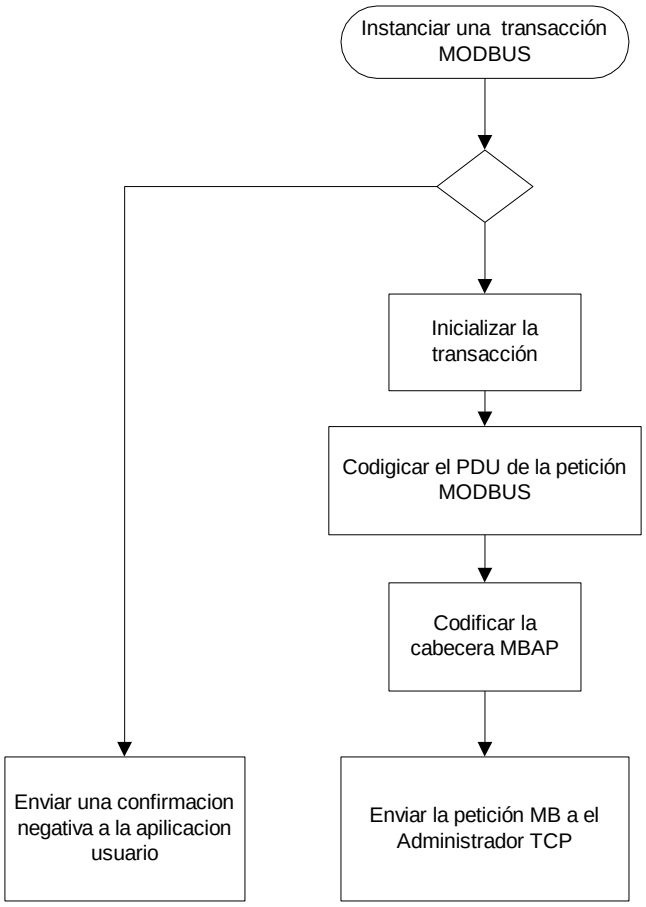


**Figura 1. 4** Diagrama de actividad para la construcción de una Petición Modbus

El siguiente ejemplo describe la codificación del ADU en la petición MODBUS para leer una salida discreta #4 en un servidor remoto:

**Tabla 1. 2** Codificación del ADU en la petición MODBUS

| Campos | Campos Internos | Tamaño | Ejemplo |
|---|---|---|---|
| Cabecera MBAP | Identificador de transacción Hi | 1 | 0x15 |
| | Identificador de transacción Lo | 1 | 0x01 |
| | Identificador de protocolo | 2 | 0x0000 |
| | Longitud | 2 | 0x0006 |
| | Identificador de unidad | 1 | 0xFF |
| Petición MODBUS | Código de función | 1 | 0x02 |
| | Dirección de inicio | 2 | 0x0004 |
| | Cantidad de salida | 2 | 0x0001 |

**Identificador de transacción**
El identificador de transacción es usado para asociar la respuesta futura con la petición. Por lo cual, a un tiempo, sobre una conexión TCP, este identificador debe ser único. Existen diferentes maneras de usar el identificador de transacción:

Por ejemplo, este puede ser usado como un simple "numero de secuencia TCP" con un contador que es incrementado por cada petición.

Este puede también ser juiciosamente usado como un índice elegante o puntero para identificar un contexto de transacción con el objeto de memorizar el servidor remoto actual la petición MODBUS pendiente.

Normalmente, un cliente MODBUS sobre una línea serial debe enviar una petición a la vez. Esto significa que el cliente debe esperara por la respuesta a la primera petición antes de enviar una segunda petición. Sobre TCP/MODBUS, muchas peticiones pueden ser enviadas sin esperar por una confirmación al mismo servidor. El gateway MODBUS/TCP a MODBUS línea serial es el encargado de asegurar la compatibilidad entre estos dos comportamientos.

El número de peticiones aceptadas por un servidor depende sobre su capacidad en término de número de recursos y tamaño de la ventana TCP. En el mismo sentido el número de transacciones inicializadas simultáneamente por un cliente depende también de su capacidad de recursos. Este parámetro de implementación es llamado "NumberMaxOfClientTransaction" y debe ser descrito como una de las características del cliente. Dependiendo del tipo de dispositivo este parámetro puede tomar un valor de 1 a 16.

**Identificador de unidad**
Este campo es usado para propósitos de ruteo cuando se direcciona un dispositivo sobre una subred MODBUS o línea serial MODBUS+. En este caso, el "identificador de unidad" porta la dirección del esclavo MODBUS del dispositivo remoto:

Si el servidor MODBUS es conectado a una subred MODBUS+ o línea serial MODBUS y es direccionado a través de un bridge o gateway, el identificador de unidad MODBUS es necesario para identificar el dispositivo esclavo conectado sobre la subred atrás del bridge o gateway. La dirección de IP destino identifica al bridge adecuado, quien a su vez usa el identificador de unidad MODBUS para enviar la petición a el dispositivo esclavo correcto.

La dirección del dispositivo esclavo MODBUS sobre línea serial son asignados de 1 a 247 (decimal). Dirección 0 es usada como dirección broadcast.

Sobre TCP/IP, el servidor MODBUS es direccionado usando su dirección IP; por lo cual, el identificador de unidad MODBUS no es usado. El valor 0xFF debe ser usado.
Cuando se direcciona un servidor MODBUS conectado directamente a una red TCP/IP. Es recomendable no usar una dirección de esclavo MODBUS significativa en el campo "identificador de unidad". En el evento de una relocalización de la dirección IP dentro de un sistema automático y si una dirección IP previamente asignada a un servidor MODBUS

es entonces asignado a un gateway, usando una dirección de esclavo significativa puede causar problemas debido a un mal ruteo por el gateway.
Usando una dirección de esclavo no significativa, el gateway simplemente descartará el PDU MODBUS sin problemas. 0xFF es recomendado para el "identificador de unidad" con un valor no significante.

*NOTA: El valor 0 es también aceptado para comunicación directa a un dispositivo MODBUS/TCP.*

**Confirmación de procesos MODBUS**
Cuando una trama de respuesta es recibida sobre una conexión TCP, el portador del identificador de transacción en la cabecera MBAP es usada para asociar la respuesta con la petición original previamente enviada sobre la conexión TCP.

Si el identificador de transacción no se refiere a ninguna transacción pendiente MODBUS, la respuesta debe ser descartada.

Si el identificador de transacción se refiere a una transacción pendiente MODBUS, la respuesta debe ser analizada con el objeto de enviar una confirmación MODBUS a la aplicación usuario. (Confirmación positiva o negativa)

Analizar la respuesta consiste en verificar la cabecera MBAP y el PDU de la respuesta MODBUS.

**Cabecera MBAP**
Después de la verificación del identificador de protocolo que debe ser 0x0000, la longitud proporciona el tamaño de la respuesta MODBUS.

Si la respuesta viene desde un dispositivo servidor MODBUS directamente conectado a la red TCP/IP, la identificación de conexión TCP es suficiente para identificar sin ambigüedad el servidor remoto. Por lo tanto, el portador del identificador de unidad en la cabecera MBAP no es significante (valor 0XFF) y debe ser descargado.

Si el servidor remoto es conectado a una subred de línea serial y la respuesta viene desde un bridge, un router o un gateway, entonces el identificador de unidad (!=0xFF) identifica el servidor remoto MODBUS que ha originalmente enviado la respuesta.

**PDU de respuesta MODBUS**
El código de función debe ser verificado y el formato de respuesta MODBUS analizado de acuerdo al protocolo de aplicación MODBUS:

Si el código de función es el mismo que el usado en la petición, y si el formato de respuesta es el correcto, entonces la respuesta MODBUS es proporcionada a la aplicación usuario como una *confirmación positiva*.

Si el código de función es un código de excepción MODBUS (Código de función +80H), la respuesta de excepción MODBUS es dada a la aplicación de usuario como una *confirmación positiva.*

Si el código función es diferente de el usado en la petición ( =código función no esperado), o si el formato de la respuesta es incorrecta, entonces un error es señalizado a la aplicación usuario usando una *confirmación negativa.*

El diagrama de actividad de la figura 1.5 describe, más profundamente el proceso de la fase de confirmación.

**Administrador del time-out**

Deliberadamente no existen especificaciones del tiempo de respuesta requerido para una transacción sobre MODBUS/TCP.

Esto es debido a que MODBUS/TCP se espera que sea usado en una amplia variedad posible de situaciones de comunicación, desde escaneo I/O sujeto a tiempo en milisegundos hasta largas distancias de enlaces de radio con retardos de muchos segundos.

Desde una perspectiva de cliente, el time-out debe tomar en cuenta el retardo esperado en el transporte a través de la red, para determinar un tiempo de respuesta "razonable". Tal retardo de transporte debe ser en milisegundos para una red ruteada Ethernet o cientos de milisegundos para una conexión de red WAN[2].

Hay que tener en mente que todo tiempo "time-out" usado en un cliente para inicializar un re-envió en la aplicación debe ser mayor que el tiempo de respuesta "razonable" máximo esperado. Si esto no es seguido, será un causa potencial para congestión potencial sobe el dispositivito objetivo o sobre la red, que puede por lo cual causar muchos errores. Esto es una situación que siempre debe ser evitada.

Por lo cual en la práctica, el time-out del cliente usado en aplicaciones de alto desempeño son siempre dependientes de la topología de la red y el desempeño esperado del cliente. Aplicaciones que no son críticas en tiempo pueden a menudo permitir valores de time-out por defecto en TCP, que reportaran falla de comunicación después de varios segundos sobre varias plataformas.

[2] Una **red de área amplia**, **WAN**, acrónimo de la expresión en idioma inglés '**Wide Area Network**', es un tipo de red de computadoras capaz de cubrir distancias desde unos 100 hasta unos 1000 km, dando el servicio a un país o un continente

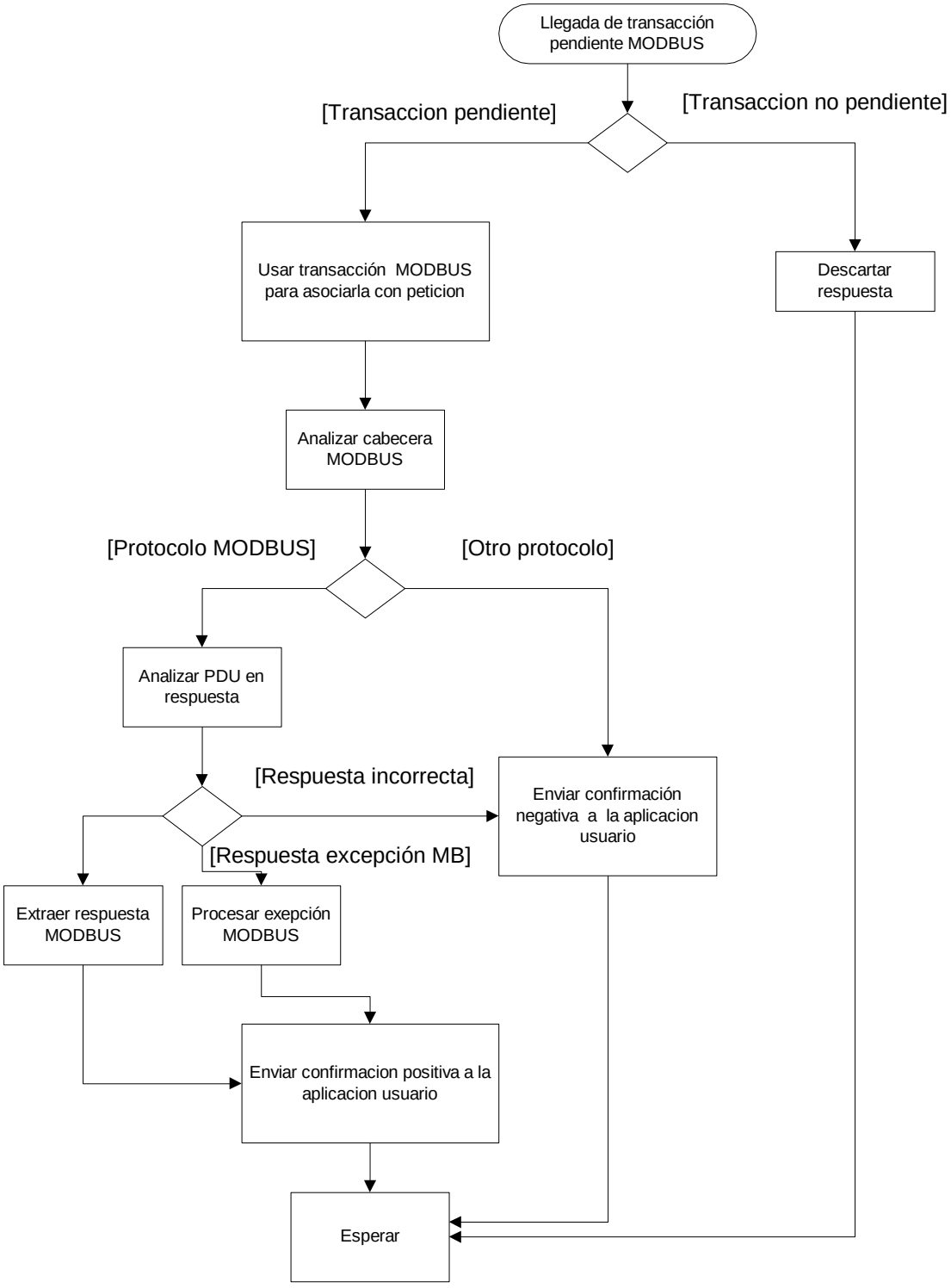


**Figura 1. 5** Análisis del PDU

### 1.1.5 Servidor Modbus.

Sobre la recepción de una petición MODBUS este módulo activa una acción local para leer, escribir o llevar a cabo otras acciones. El procesamiento de estas acciones es hecho totalmente transparente a la aplicación de programador. La principal función del servidor MODBUS es esperar por una petición MODBUS sobre el puerto TCP 502, para tratar esta petición y entonces construir una respuesta MODBUS dependiendo del contexto del dispositivo.
El rol de un servidor MODBUS es proveer acceso al objeto aplicación y servicios al cliente remoto MODBUS.

Diferentes tipos de acceso pueden ser proveídos dependiendo de la aplicación de usuario:

- Acceso simple para obtener y establecer atributos de objetos aplicación.
- Acceso avanzado con el objeto disparar servicios específicos de aplicación.

El servidor MODBUS tiene que:

- Trazar objetos aplicación hacia objetos MODBUS que se puedan leer y escribir, con el propósito de obtener o establecer atributos de los objetos aplicación.
- Proveer un camino para servicios "trigger" sobre objetos aplicación.

En tiempo de corrido el servidor MODBUS tiene que analizar una petición MODBUS, para procesar la acción requerida, y enviar de regreso una respuesta MODBUS.

**Construcción de respuesta MODBUS**
Una vez la petición ha sido procesada, el servidor MODBUS tiene que construir la respuesta usando el adecuado servidor de transacciones MODBUS y tiene que enviar ésta al componente administrador TCP.
Dependiendo de los resultados del procesamiento, dos tipos de respuesta pueden ser construidas:

Una respuesta positiva MODBUS:
El código de función de la respuesta = el código función de la petición.
Una respuesta de excepción MODBUS:
El objetivo es proveer al cliente información relevante concerniente a la detección de errores durante el procesamiento.
El código de función de la respuesta = condigo de función de la petición + 0x80;
El código de función es proveído o indica la razón del error.

**Tabla 1. 3** Excepciones MODBUS.

| Código de excepción | Nombre MODBUS | Comentarios |
|---|---|---|
| 01 | Código de Función ilegal | El código función es desconocido por el servidor |
| 02 | Dirección de datos ilegal | Dependiendo de la petición |
| 03 | Valor de dato ilegal | Dependiendo de la petición |
| 04 | Fallo servidor | El servidor fallo durante la excepción |
| 05 | Acknowledge | El servidor acepta la invocación de servicios pero el servicio requiere un tiempo relativamente largo tiempo ejecutarse. El servidor por lo tanto únicamente retorna una acknowledgement de la recepción de la invocación del servicio. |
| 06 | Servidor trabajando | El servidor es incapaz de aceptar la el PDU de la petición MODBUS. La aplicación cliente tiene la responsabilidad de decidir si y cuando reenviar la petición. |
| 0A | Problema con Gateway | La dirección del gateway no esta disponible |
| 0B | Problema con Gateway | El dispositivo objetivo fallo en la respuesta. El gateway genera esta excepción. |

El PDU de la respuesta MODBUS debe ser prefijado con la cabecera MBAP que es construida usando memoria de datos en el contexto de la transacción.

**Identificador de unidad**
El identificador de unidad es copiado como éste fue dado dentro de la petición MODBUS recibida y memorizada en el contexto de la transacción.

**Longitud**
El servidor calcula el tamaño del PDU MODBUS más el byte del identificador de unidad. Este valor es establecido en el campo "longitud".

**Identificador de protocolo**
El campo del identificador de protocolo es establecido a 0x0000 (protocolo MODBUS), como este fue dado dentro de la petición recibida MODBUS.

**Identificado de transacción**
Este campo es establecido al valor del "Identificador de transacción" que fue asociado con la petición original y memorizada en el contexto de la transacción.

Entonces la respuesta MODBUS debe ser retornada al cliente MODBUS correcto usando la conexión TCP memorizada en el contexto de la transacción. Cuando la respuesta es enviada, el contexto de la transacción debe ser liberado.

## 1.2 Costos del SACME y Sistemas Similares

El SACME es un sistema de control electrónico programable accedido desde un cliente Modbus, al igual que el SACME existen muchos sistemas similares, cada uno con sus particularidades y deficiencias. En esta sección se hace una comparación del SACME con algunos sistemas y determinar las características y deficiencias del mismo.

### 1.2.1 DD-6390 IP RACK CONTROL POR INTERNET

Este sistema es controlado desde Internet a través de páginas Web contenidas en el mismo sistema, a continuación se enlistan sus características:


**Figura 1. 6** Sistema DD-630 rack

- Servidor de páginas incluido
- 8 Entradas de sensores ( 4 entradas por voltaje ON= 4V-24V, OFF= 0V-3V y 4 entradas resistivas ON= 0-200 Ohmios, OFF=500- Infinito Ohmios)
- 8 Salidas relé (4 entradas normalmente abiertas y 4 normalmente cerradas ,240VAC/12A, 60VDC)
- Soporta (WAN/LAN), HTTP, TCP/IP, DHCP, DDNS, WAP por GPRS
- Servidor SMTP, alertas vía e-mail cuando se producen eventos.
- LED,s individuales de estado
- Actualización Online

- Protección contra inversión de polaridad
- SDK VisualBasic y VC posible
- Compatible con IP Video
- WacthDog

Como se muestra en la figura 1.6, este sistema cuenta con una serie de entradas y salidas dispuestas en el mismo módulo, sin la posibilidad de expansión. El precio estimado en el mercado para este sistema es de $575.36 sin costos de instalación.

### 1.2.2 ADAM 5000/TCP

Este sistema es un poco más parecido al SACME, ya que cuenta con módulos que pueden ser agregados a uno ya existente. Las características que tiene son las siguientes:

- ARM 32-bit RISC CPU
- 10/100Base-T auto-negotiation high-speed communication port
- Supports Modbus/TCP for easy integration
- Supports UDP event handling function
- Up to 100 m communication distance w/o repeater
- Allows remote configuration via Ethernet
- Allows concurrent access for 8 host PCs
- I/O slots for up to 64 points and 8 I/O slots for up to 128 points data
- 1500 VDC isolation for Ethernet communication
- Built-in watchdog timer for system auto-reset
- Windows utility
- Provides ActiveX drivers to develop applications

**Figura 1. 7** Sistema ADAM 5000/TCP

La figura 1.7 muestra el módulo principal del ADAM, este módulo tiene capacidad para albergar a 8 sub-módulos de expansión, los cuales pueden ser de entradas o salidas o una combinación de ambas, también trabaja con señales digitales y analógicas. La figura 1.8 muestra un módulo de expansión.

**Figura 1. 8** Módulo de expansión del ADAM 5000/TCP

Este sistema tiene un costo estimado de $717 solo para el módulo electrónico, luego cada bloque de expansión tiene un  costo de $109.

### 1.2.3 SACME

El SACME también tiene sus costos, a continuación presentamos sus características y los cotos de cada módulo.

- 10/100 Mbits Ethernet auto sensig
- 160 Entradas digtales
- 160 Salidas por pulsos a 240V/16A
- Soporta Modbus/TCP
- Permite configuración vía ethernet y serial
- Salidas y entradas personalizables

- Wathdog para auto reset
- Reporte de errores vía Email
- Expansión por módulos

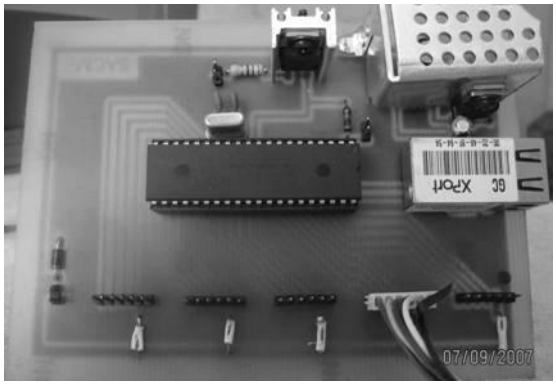


**Figura 1. 9** Módulo electrónico del SACME

El módulo electrónico mostrado en la figura 1.9 es el corazón del SACME, este módulo cuenta con una serie de buses para conectar los bloques de expansión. La figura 1.10 muestra los boques de expansión.

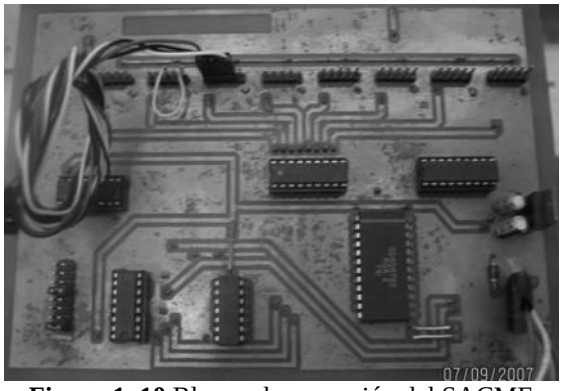


**Figura 1. 10** Bloque de expansión del SACME

El bloque de expansión es el que tiene a cargo un grupo de 8 entradas y 8 salidas. A este bloque se conectan los bloques de manejo de cargas mostrado en la figura 1.11.

Los bloques de manejo de carga tiene como función principal la de manejar las cargas mediante un rele de 16 Amperios activado por pulsos. Otra función que tiene este bloque es la de generar un estado lógico dependiendo del estado de la carga.

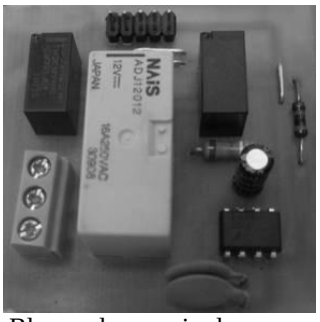


**Figura 1. 11** Bloque de manejo de cargas del SACME.

Los costos de estos módulos del SACME son:

| | |
|---|---|
| Módulo electrónico: | $97.68 |
| Bloques de expansión: | $26.82 |
| Bloques de manejo de cargas: | $27.52 |

El precio un poco elevado de los bloques de manejo de cargas es por el costo del rele de pulsos, ya que es un componente de alta tecnología y eficiencia en el consumo de energía. Así para el manejo de 8 cargas el sistema tiene un costo de $ 344.66

**Tabla 1. 4** Detalle del Costo del Módulo Electrónico.

| Cantidad | Descripción | Precio unitario (US$) | Total (US$) |
|---|---|---|---|
| 1 | Microcontrolador PIC16F877A | $18.09 | $18.09 |
| 1 | GC-XPORT-MBTCP, serial/Ethernet RJ45 | $59.65 | $55.00 |
| 1 | Regulador de voltaje positivo UA-7805 | $0.91 | $0.91 |
| 1 | Regulador de voltaje positivo LM317T | $0.60 | $0.60 |
| 2 | Diodo 1N4004 | $0.15 | $0.30 |
| 2 | Capacitores Electrolitico | $0.30 | $0.60 |
| 2 | Capacitores de pastilla | $0.26 | $0.52 |
| 1 | Resonador 4MHz. | $1.50 | $1.50 |
| 4 | Resistencias ¼ W | $0.25 | $1.00 |
| 1 | Led verde | $0.17 | $0.17 |
| 1 | Push button | $1.00 | $1.00 |
| 2 | Conector mini Din 2 pines macho | $0.64 | $1.28 |
| 5 | Conector mini Din 5 pines macho | $0.64 | $3.20 |
| 1 | Base de 40 pines | $0.45 | $0.45 |
| 1 | Placa de circuito impreso 15x20 cm | $1.99 | $1.99 |
| 1 | Onza de percloruro de hierro. | $1.25 | $1.25 |
| | Costo de fabricación circuito impreso. | $5.00 | $5.00 |
| 1 | Yardas de estaño | $0.17 | $0.17 |
| | | **Total** | **$ 97.68** |

**Tabla 1. 5** Detalle del Costo por Bloque de expansión.

| Cantidad | Descripción | Precio unitario (US$) | Total (US$) |
|---|---|---|---|
| 1 | Regulador de voltaje positivo UA-7805 | $0.91 | $0.91 |
| 1 | CD4051B, Mux/Dem Analogo 8 Canales | $0.79 | $0.79 |
| 1 | CD4067BE, Mux/Dem Analogo 16 Canales | $4.55 | $4.55 |
| 1 | SN74154, Decoder | $1.03 | $1.03 |
| 1 | SN74LS138N, Decoder | $0.91 | $0.91 |
| 1 | Diodo 1N4004 | $0.15 | $0.15 |
| 8 | Conector mini Din 5 pines macho | $0.64 | $5.12 |
| 2 | ULN-2803, buffer de salida | $1.35 | $2.70 |
| 2 | Base de 8 pines | $0.45 | $0.90 |
| 3 | Base de 9 pines | $0.45 | $1.35 |
| 1 | Placa de circuito impreso 15x20 doble cara | $1.99 | $1.99 |
| 1 | Onza de percloruro de hierro. | $1.25 | $1.25 |
| | Costo de fabricación circuito impreso. | $5.00 | $5.00 |
| 1 | Yardas de estaño | $0.17 | $0.17 |
| | | **Total** | **$ 26.82** |

**Tabla 1. 6** Detalle del Costo por bloque de manejo de carga.

| Cantidad | Descripción | Precio unitario (US$) | Total (US$) |
|---|---|---|---|
| 1 | ADJ12012, Latching Relay | $10.70 | $10.70 |
| 1 | HCPL3700, AC/DC to logic optocoupler | $3.28 | $3.28 |
| 2 | Resistencias ¼ W | $0.25 | $0.50 |
| 1 | Conector mini Din 5 pines macho | $0.64 | $0.64 |
| 1 | Mini Terminal de 3 pines 5mm | $1.50 | $1.50 |
| 3 | Capacitores de pastilla | $0.26 | $0.78 |
| 2 | Relay 12V | $2.85 | $5.70 |
| 1 | Placa de circuito impreso 6x6 doble cara | $1.00 | $1.00 |
| 1 | Onza de percloruro de hierro. | $1.25 | $1.25 |
| | Costo de fabricación circuito impreso. | $2.00 | $2.00 |
| 1 | Yardas de estaño | $0.17 | $0.17 |
| | | **Total** | **$ 27.52** |

# CAPITULO II

# DISEÑO E IMPLEMENTACION DEL HARDWARE DEL SACME

## 2.1 Introducción

La implementación de un diseño requiere de una amplia búsqueda de tecnología que se adapte a las necesidades del diseño, esto lleva a muchas situaciones en las que el funcionamiento de los circuitos implementados no es el esperado, aquí se genera una retroalimentación entre la etapa de diseño y el correspondiente cambio en la implementación hasta alcanzar un funcionamiento satisfactorio.

El hardware del SACME esta formado por tres partes que son: el módulo electrónico, los bloques de expansión y los bloques de manejo de carga. En este capitulo se presenta en diseño del hardware como un conjunto, el cual es analizado posteriormente como elementos separados para destacar las características y funcionamiento de cada una de dichas partes.

## 2.2 Criterios de diseño

Para realizar el diseño del hardware se ha tomado en cuenta una serie de criterios a fin de darle el funcionamiento correcto así como también cumplir con los objetivos propuestos para el desarrollo de este trabajo de graduación. Los criterios son los siguientes:

- **Selección de la tecnología**: se ha dado especial atención al bajo costo de los elementos a utilizar, así como también a su consumo de energía.

- **Protocolo de comunicación**: se utiliza un protocolo de comunicación estándar, ampliamente usado en la industria y de fácil implementación.

- **Escalabilidad**: el hardware se implementa en forma modular, lo que permite realizar expansiones futuras, también se ha provisto de la base necesaria para la implementación de nuevas funciones sin la necesidad de modificar el firmware[3].

- **Manejo de las cargas**: el control local de las cargas se ha mantenido además del control establecido por el SACME.

[3] El firmware es el programa en lenguaje ensamblador que reside en el microcontrolador y dicta su funcionamiento.

## 2.3 Diagrama de caso de uso

El funcionamiento del hardware se puede entender mejor mediante un diagrama de caso de uso, en dicho diagrama se muestra las funciones que realiza el hardware y su interacción con el sitio Web. La figura 2.1 muestra el diagrama de caso de uso del hardware del SACME.

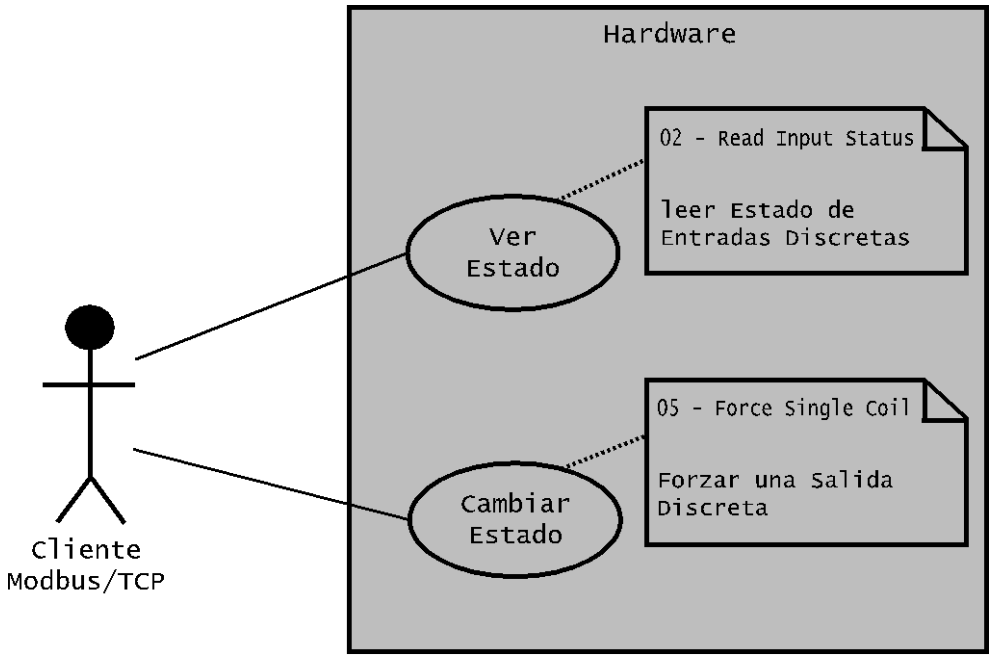


**Figura 2. 1** Diagrama de caso de uso del hardware del SACME.

En la figura anterior se muestra el hardware representado por un cuadro y en su interior las funciones que puede realizar, fuera del cuadro se encuentra un actor, el cual representa ya sea un sitio Web o cualquier aplicación que requiera un uso de las funciones del hardware. Es importante notar que la comunicación entre el cliente y hardware se desarrolla bajo un protocolo, el protocolo Modbus, lo que permite que el hardware pueda ser utilizado en cualquier sistema con Modbus como protocolo de comunicación.

Las funciones que puede realizar el hardware son mostradas en la tabla siguiente.

**Tabla 2. 1** Funciones implementadas en el hardware.

| **Función** | **Propósito** |
|---|---|
| Read input status(0x02): | Esta función permite ver el estado de una o varias entradas digitales consecutivas, las cuales representan el estado de las cargas |
| Write single coil(0x05): | Forzar el estado de una salida, esta salida representa una carga que será encendida o apagada dependiendo del estado especificado en la función |

Estas dos funciones corresponden a las funciones predefinidas del protocolo Modbus y pueden ser agregadas otras similares para manejo de entradas y salidas según sea la aplicación que se le de al hardware de SACME.

## 2.4 Diagrama de bloques del hardware

Ya que se ha visto el funcionamiento del hardware, se presenta los elementos que lo componen mediante un diagrama de bloques. La figura 2.2 muestra el diagrama de bloques del hardware del SACME.

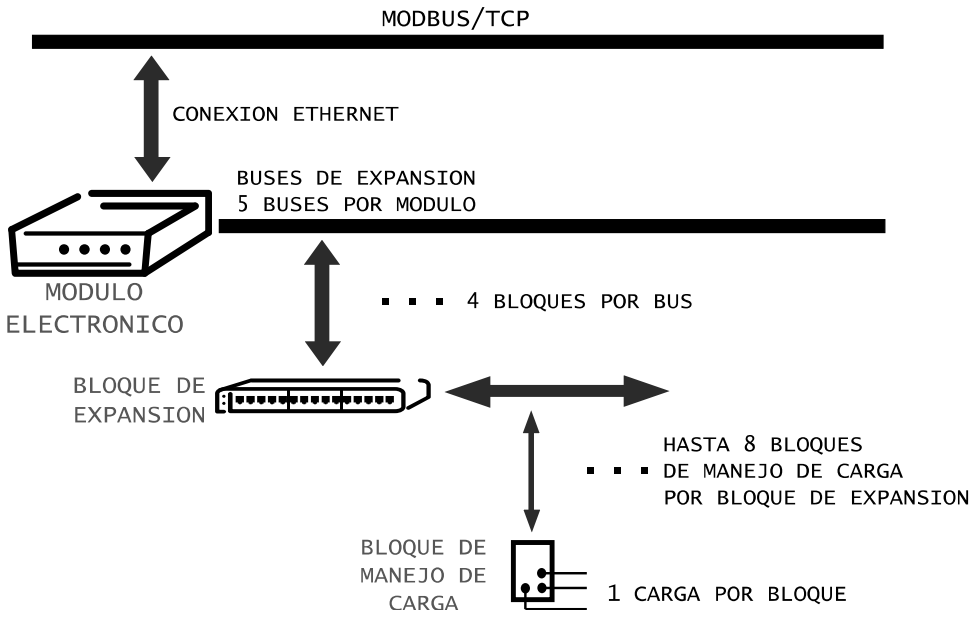


**Figura 2. 2** Diagrama de bloques del hardware del SACME.

El hardware se ha dividido en tres bloques que corresponden a: el módulo electrónico, los bloques de expansión y los bloques de manejo de carga. Esta división se ha hecho siguiendo uno de los criterios de diseño, el cual exige de el hardware pueda ser implementado de forma modular. Esto es importante ya que se puede ir agregando cargas hasta alcanzar la capacidad total. Para que funcione el hardware como mínimo debe tener el módulo electrónico, un bloque de expansión y un bloque de manejo de carga.
El manejo de cargas se hace siguiendo un orden numérico en el que cada carga tiene asignado un número dependiendo del punto en donde es conectada al hardware del SACME. Este orden se muestra en la tabla 2.1.

**Tabla 2. 2** Tabla de direccionamiento de las cargas.

| BUS | PUERTO UTILIZADO | MODULO DE EXPANSION | RANGO DE DIRECCIONES |
|---|---|---|---|
| 1 | PUERTO B NIBBLE BAJO | 1 | 01-08 |
| | | 2 | 09-16 |
| | | 3 | 17-24 |
| | | 4 | 25-32 |
| 2 | PUERTO B NIBBLE ALTO | 5 | 33-40 |
| | | 6 | 41-48 |
| | | 7 | 49-56 |
| | | 8 | 57-64 |
| 3 | PUERTO D NIBBLE BAJO | 9 | 65-72 |
| | | 10 | 73-80 |
| | | 11 | 81-88 |
| | | 12 | 89-96 |
| 4 | PUERTO D NIBBLE ALTO | 13 | 097-104 |
| | | 14 | 105-112 |
| | | 15 | 113-120 |
| | | 16 | 121-128 |
| 5 | PUERTO C NIBBLE BAJO | 17 | 129-136 |
| | | 18 | 137-144 |
| | | 19 | 145-152 |
| | | 20 | 153-160 |

Se manejan en total 160 cargas con un módulo electrónico, para ello se dispone de 5 buses por módulo, en donde se pueden conectar 4 bloques de expansión por bus y finalmente 8 cargas por bloque de expansión. Los puertos indicados son los del microcontrolador pic16f877a.

Para entender mejor el funcionamiento de cada bloque de la figura 2.2 se trata en las secciones siguientes cada uno por separado.

## 2.5 Diseño e implementación del módulo electrónico

El módulo electrónico es la parte central del hardware del SACME, esta basado en un microcontrolador que maneja tanto las entradas y salidas hacia los dispositivos eléctricos mediante buses de expansión como también la comunicación externa o proveniente del cliente Modbus. El módulo electrónico realiza el acceso a los dispositivos mediante un sistema de multiplexación de pines que se implementa en los bloques de expansión, esto debido a que la existencia de pines de entrada/salida en un microcontrolador es limitada. El módulo electrónico implementa un servidor Modbus/TCP para ser accedido desde una red ethernet sin importar si las peticiones las hace un servidor Web que implemente un cliente Modbus personalizado, o un cliente Modbus comercial.

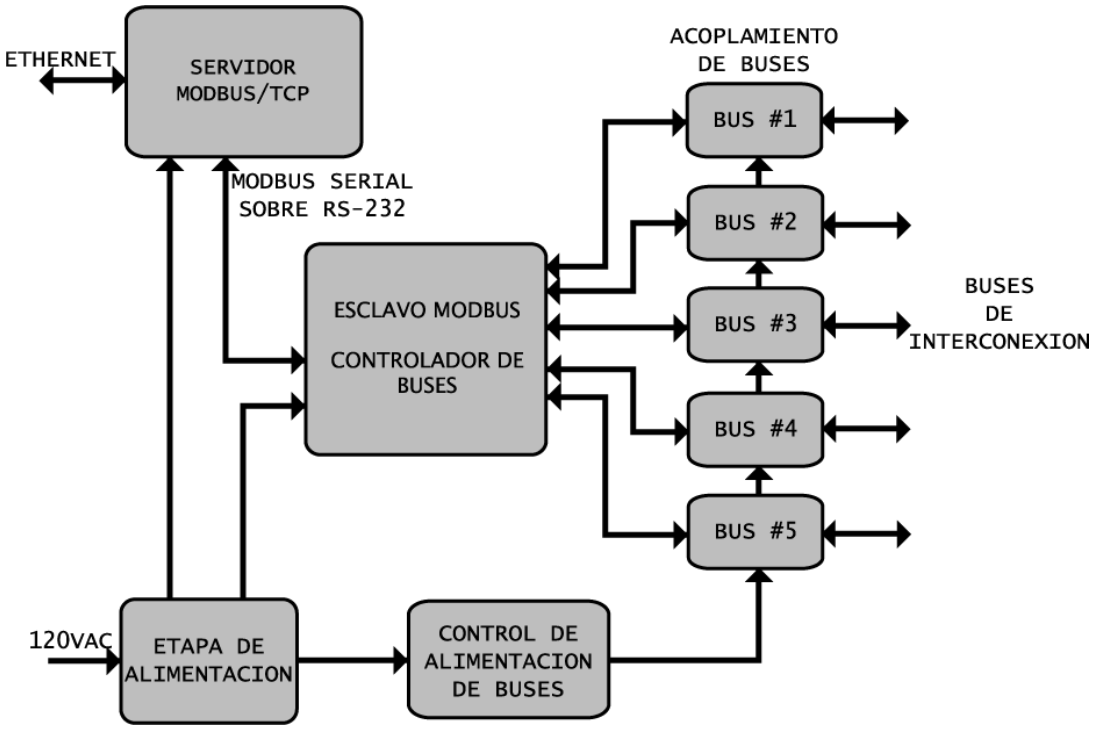


**Figura 2. 3** Diagrama de bloques del Módulo electrónico.

El diagrama de bloques de la figura 2.3 muestra los componentes del módulo electrónico, en el que se destaca dos elementos importantes, el servidor Modbus/TCP y el esclavo Modbus. Estos dos elementos contienen la lógica de funcionamiento de módulo electrónico, a continuación se detalla cada uno.

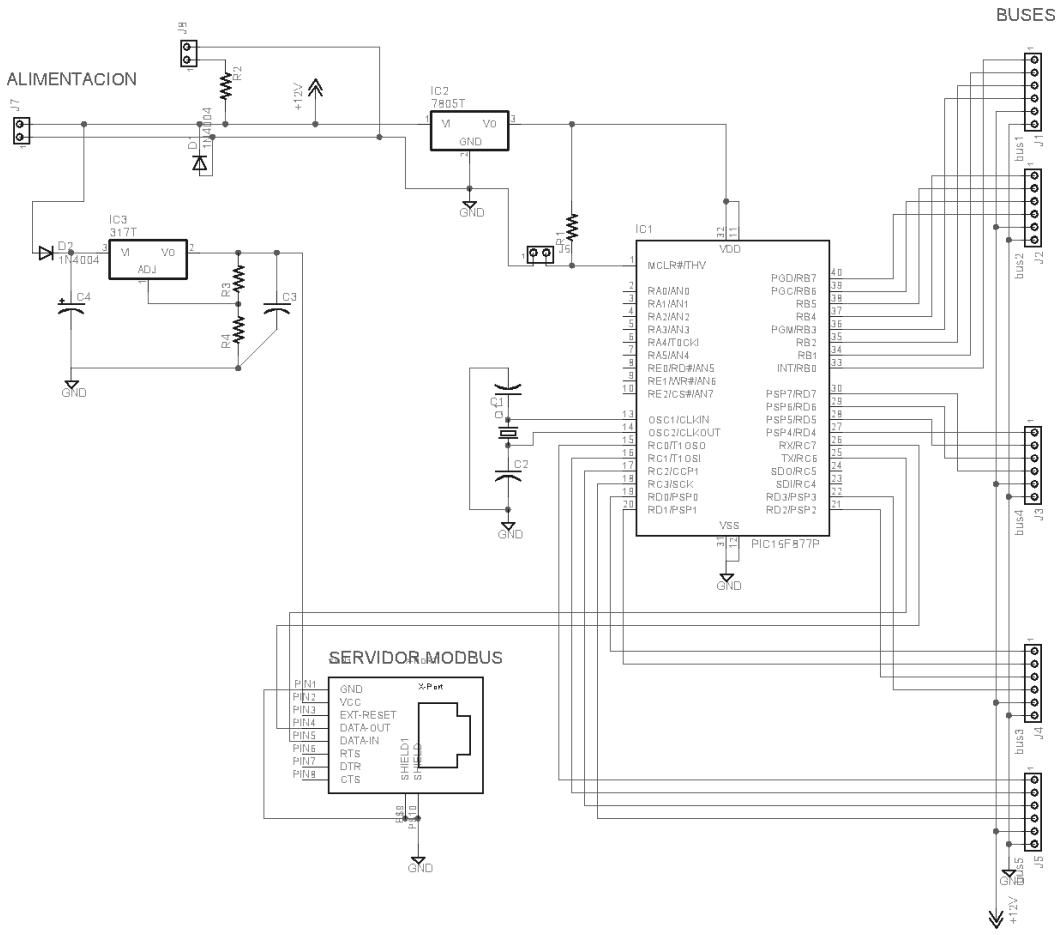


**Figura 2. 4** Diagrama esquemático del Módulo electrónico.

## 2.5.1 Servidor Modbus/TCP

El módulo electrónico trabaja por medio de peticiones, estas peticiones son hechas por un cliente que se comunica vía ethernet, para ello utiliza el protocolo de comunicación Modbus en su modalidad TCP, es decir Modbus a través de una red ethernet.

Para manejar esta comunicación, el módulo electrónico cuenta con un servidor Modbus también sobre TCP, el cual se encarga de recibir todas las peticiones que le haga el cliente y darles tratamiento, ya sea para ejecutar la operación solicitada o bien advertir que dicha función no es soportada en caso de que se solicite una función que no ha sido implementada en el módulo electrónico.

Es importante mencionar que todo el módulo electrónico se visualiza como un servidor Modbus, ya que no existe otra forma de acceder a el que no sea a través del servidor.

Internamente se ve de otra forma, el módulo cuenta con un gateway[4] embebido que hace la conversión de ethernet a serial y viceversa.

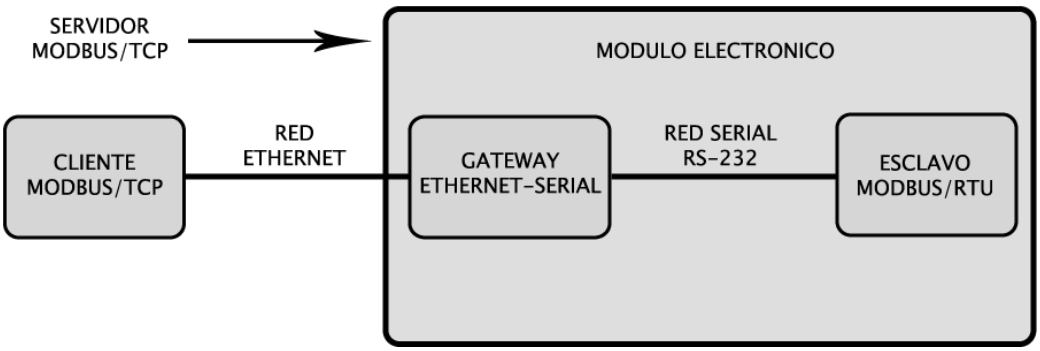


**Figura 2. 5** Representación del módulo electrónico como un servidor Modbus/TCP.

En la figura 2.5 se observa que toda la comunicación con el cliente pasa a través del gateway, este gateway es completamente transparente y soporta todo el stack de funciones estándar de Modbus, por lo que la capacidad del módulo no depende de este (gateway) sino del esclavo modbus implementado en el controlador del módulo.

El gateway esta implementado con un dispositivo de alta tecnología en integración digital, cuenta con su propio microprocesador y un firmware actualizable. En los anexos se muestra los detalles de este dispositivo.

### 2.5.2 Esclavo Modbus y controlador de buses

Toda la lógica de control de buses y direccionamiento de las cargas se realiza en este elemento del módulo electrónico, esta basado en el microcontrolador pic16f877a e implementa un algoritmo de un esclavo Modbus sobre línea serial como marco de funcionamiento. Dentro de este funcionamiento de esclavo Modbus es que se implementa las funciones de direccionamiento de las cargas, lectura y cambio de estados de las cargas y serialización de dichas direcciones para ser enviadas a través de los buses hacia los módulos de expansión.

[4] Es un dispositivo que sirve de Puente entre un medio de transmisión y otro.

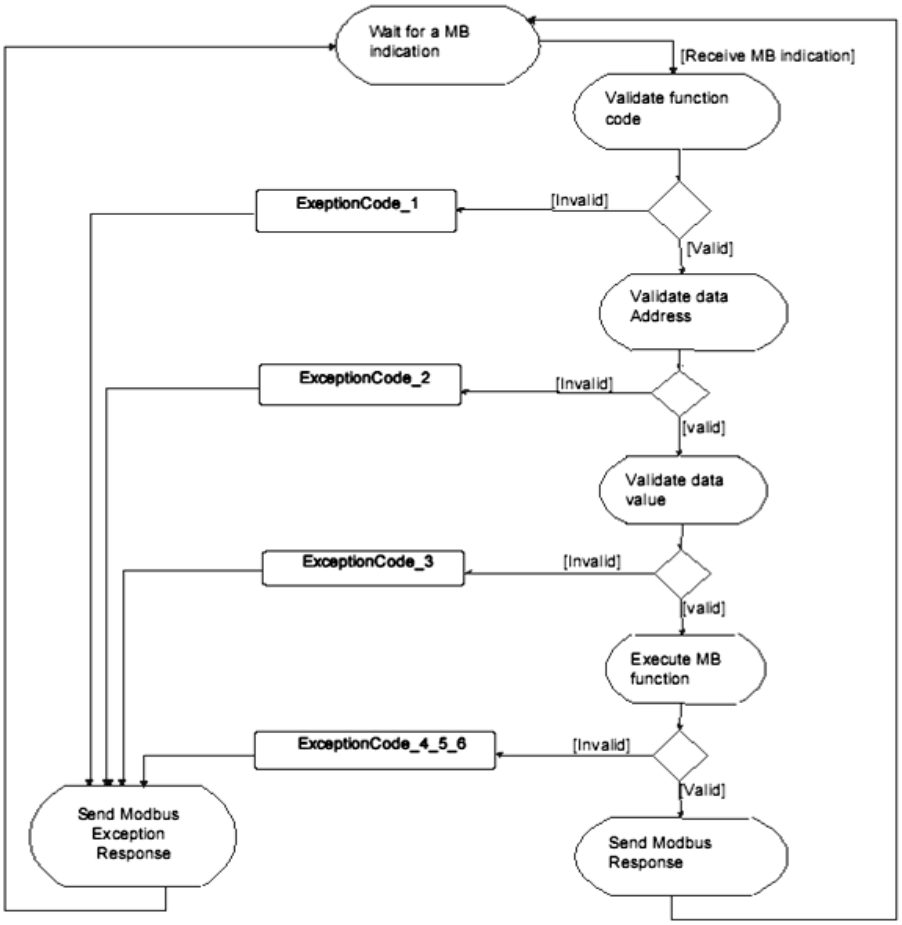


**Figura 2. 6** Diagrama de flujo del esclavo Modbus.

En la figura 2.6 se observa el diagrama de flujo del funcionamiento del esclavo Modbus implementado en el controlador del módulo electrónico.

El funcionamiento es este: el controlador esta siempre alerta esperando una petición Modbus, una vez ha recibido una petición la analiza para determinar si puede ejecutar la operación solicitada o si debe generar una excepción[5]. Cada petición hecha al controlador esta identificada por un número de función, si dicha función esta implementada entonces se pasa al siguiente nivel del flujograma, este consiste en verificar que la dirección de las salidas o entradas especificadas en la petición están dentro del rango de trabajo del módulo electrónico. De ser una dirección valida la solicitada se permite entonces pasar al siguiente nivel, este es el de la validación de los datos a ser usados durante la ejecución de la función como pueden ser estados de las salidas o cantidad de entradas a leer. Si todas estas verificaciones son validas entonces se ejecuta por fin la operación requerida en la petición.

La tabla 2.3 muestra las excepciones posibles durante la ejecución de una petición Modbus.

[5] Una excepción es una forma de respuesta en la cual se da las razones por la cuales no se ha realizado una operación.

**Tabla 2. 3** Excepciones generadas por el esclavo Modbus.

| Código de excepción | Nombre MODBUS | Comentarios |
|---|---|---|
| 01 | Código de Función ilegal | El código función es desconocido por el servidor |
| 02 | Dirección de datos ilegal | La dirección sobrepasa los límites soportados por el módulo electrónico. |
| 03 | Valor de dato ilegal | Depende la función solicitada |
| 04 | Fallo servidor | El servidor fallo durante la ejecución de la función. |

El hardware del SACME maneja cargas en forma de salidas y maneja sus estados (los de las cargas) en forma de entradas, para tener acceso a estas salidas y entradas se ha establecido un orden en cada función Modbus en cuanto al acceso, y las funciones propias del módulo que adecuan el direccionamiento de las funciones Modbus a la ubicación de los buses en el microcontrolador.

**Función 02: Leer Estado de Entrada Discreta (Read Input Status)**
En esta función se accede a los estados de las cargas en bloques de 8, 16, 24 y 32 entradas. Otra condición es que solo se pueden leer entradas del mismo bus durante cada lectura, esto significa que como cada bus tiene 32 entradas, solo se puede acceder a 32 entradas como máximo en cada lectura. Para leer el total de entradas que son 160 se debe hacer 5 lecturas. En la figura 2.7 se muestra el flujograma de la función 02.

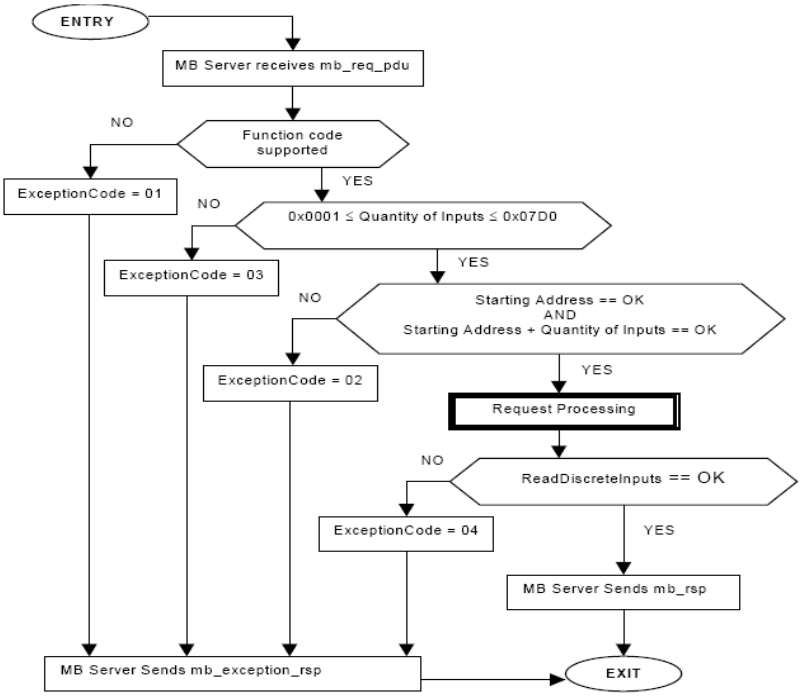


**Figura 2. 7** Diagrama de flujo de la función 02 del esclavo Modbus.

**Función 05: Forzar una Salida Discreta (Write Single Coil)**
Como su nombre lo indica, esta función solo tiene acceso a poner el estado de una única salida en cada ejecución. Lo que significa que para establecer el estado de un grupo de cargas, habrá que hacerlo una por una. La figura 2.8 muestra el diagrama de flujo de la función 05.

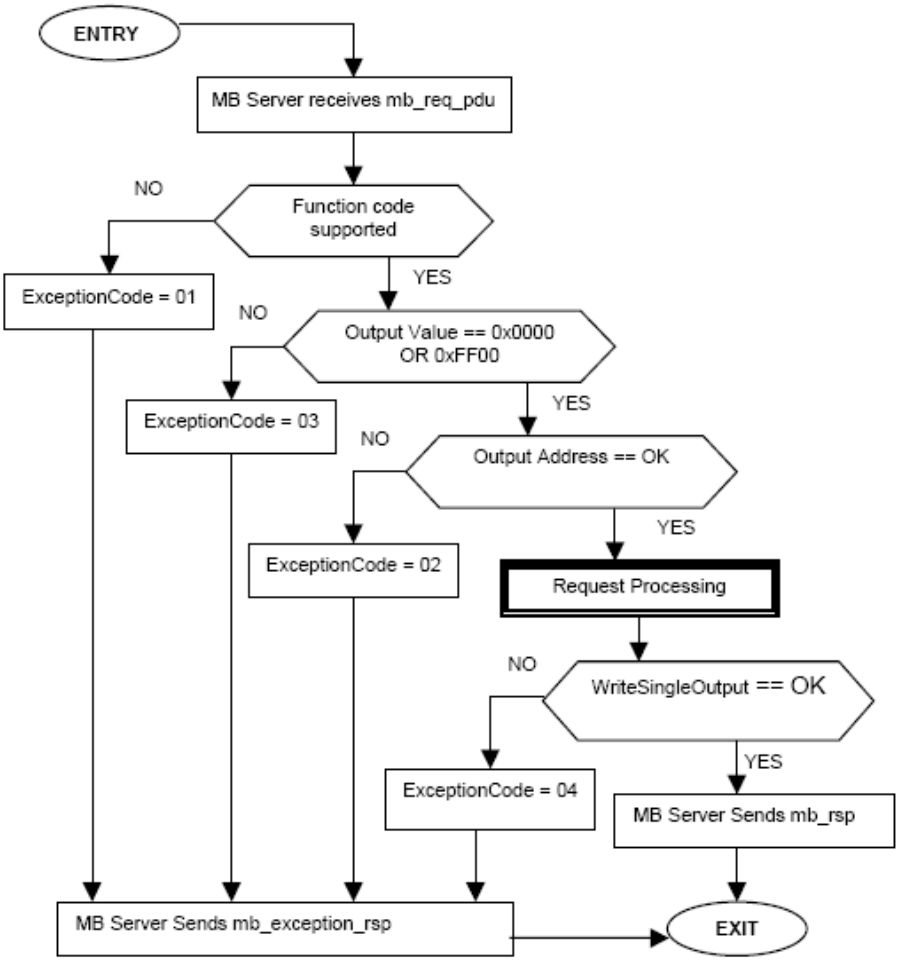


**Figura 2. 8** Diagrama de flujo de la función 05 del esclavo Modbus.

**Rutina de direccionamiento: (de uso interno en el módulo).**

Esta rutina hace una interpretación de la numeración consecutiva que Modbus le da a todas las entradas y salidas y las ubica cada una en su bus correspondiente de acuerdo a la numeración que tengan. En la tabla de direccionamiento mostrada en la tabla 2.2 se observa a que bus pertenece cada salida o entrada (cada salida esta ligada a una entrada del mismo numero).

La figura 2.9 muestra Los cinco buses del módulo electrónico y sus respectivos pines en el microcontrolador.

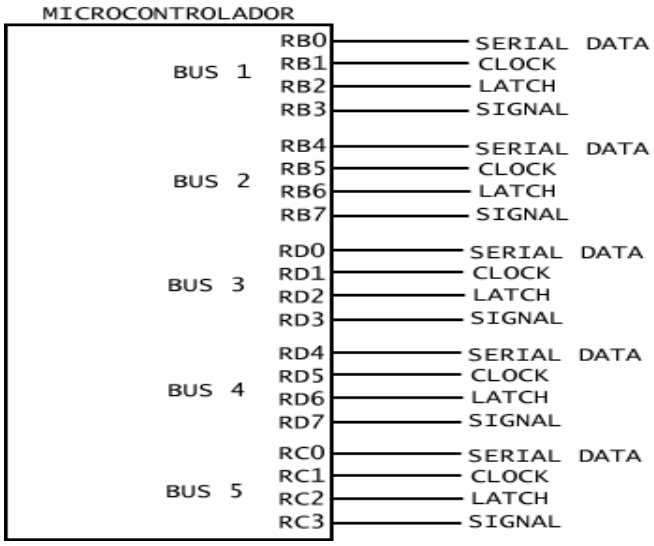


**Figura 2. 9** Pines del microcontrolador para cada bus del módulo electrónico.

El direccionamiento de entradas y salidas se realiza en dos etapas, primero se crea un patrón de direccionamiento, esto es, se crea una tabla de direcciones en la que cada elemento de la tabla representado por un número, hace referencia a una salida o entrada física. Como los pines del microcontrolador han pasado de manejar cargas a ser interfases entre el módulo electrónico y los bloques de expansión, el patrón de direccionamiento permite que cada salida o entrada sea direccionada de forma correcta.

la creación del patrón de direccionamiento depende de la forma en que funciona la multiplexación, en este caso se usa una multiplexación de 16 líneas de salida y 16 de entrada, por lo que se necesita 4 bits para direccionar esta cantidad de salidas y entradas. Aquí es importante notar que las mismas líneas que direccionan las salidas también lo hacen con las entradas.

Una vez que se tiene el patrón de 4 bits se puede direccionar 16 líneas ya sea de entrada o salida, pero es necesario establecer una forma de diferenciar entre salidas y entradas ya que utilizan los mismos 4 bits. Esto se logra al agregar otro bit al patrón, por lo que se tienen 5 bits, 4 para direccionar y uno para diferenciar entre salidas y entradas. La figura 2.10 muestra el patrón de direccionamiento.


**Figura 2. 10** Patrón de direccionamiento.

Aquí entra un punto muy importante en la multiplexación, ya que se utiliza este mismo principio de selección pero un poco mas amplio, esto es, se utiliza 1 bit para seleccionar ya sea las entradas o salidas en dentro del bloque de expansión y 2 bits mas para direccionar entre un bloque de otros tres, teniendo en total cuatro bloques de expansión direccionados en el mismo bus.


**Figura 2. 11** Patrón de direccionamiento completo.

La figura 2.11 muestra El patrón de direccionamiento completo para direccionar las salidas y entradas del módulo electrónico.

En la tabla 2.4 y la tabla 2.5, se muestra la forma de generar los bytes de dirección tanto para salidas como entradas respectivamente.

**Tabla 2. 4** Generación de los bytes de dirección de salidas.

| **Salidas** | **Numero de Bits** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| • 8 pares de salidas por módulo<br>• Si una salida no produce resultado, se utilizara su par, poniendo un uno al bit #3.<br>• 4 módulos por bus | X<br>X<br>X<br>X | 0<br>X<br>X<br>0 | 0<br>X<br>X<br>1 | 0<br>X<br>X<br>1 | 0<br>0<br>1<br>1 | 0<br>0<br>0<br>0 | 0<br>1<br>0<br>1 | 0<br>1<br>0<br>1 |
| | X | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | X | X | X | X | 0 | 1 | 1 | 1 |
| | X | X | X | X | 1 | 0 | 0 | 0 |
| | X | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| | | | | | 1 | 0 | 1 | 0 |
| | | | | | 1 | 0 | 1 | 1 |
| | | | | | 1 | 1 | 0 | 0 |
| | | | | | 1 | 1 | 0 | 1 |
| | | | | | 1 | 1 | 1 | 0 |
| | | | | | 1 | 1 | 1 | 1 |

**Tabla 2. 5** Generación de los bytes de dirección de entradas.

| Entradas | Numero de Bits | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 8 entradas por módulo | | | | | X | 0 | 0 | 0 |
| | | | | | X | 0 | 0 | 1 |
| 4 módulos por bus | X | 1 | 0 | 0 | X | 0 | 1 | 0 |
| | X | X | X | X | X | 0 | 1 | 1 |
| | X | X | X | X | X | 1 | 0 | 0 |
| | X | 1 | 1 | 1 | X | 1 | 0 | 1 |
| | | | | | X | 1 | 1 | 0 |
| | | | | | X | 1 | 1 | 1 |

Una diferencia importante entre las entradas y salidas es que en las entradas no se utiliza el bit 3, esto se debe a que solo se direccional 8 entradas por bloque, mientras que salidas se direccionan 16, en el diseño de los bloques de manejo de carga se explica porque se utiliza el doble de salidas que de entradas.

En cada byte de dirección solo se pueden direccionar 4 bloques de 8 entradas y 16 salidas, y como en cada bus se maneja un byte de dirección, entonces se puede manejar 32 entradas y 64 salidas por bus.

**Rutina de serialización de direcciones: (de uso interno en el módulo)**.

Esta función se utiliza para convertir cada byte de dirección que se genere en la función de direccionamiento, en un tren de bits que son enviados a través de cada bus.

La Serialización de cada byte a ser enviada a través de los buses del módulo electrónico se hace de forma sencilla, se rota cada bit y luego se envía por un pin de salida. La velocidad de transmisión aproximada es de 250 kbps en promedio.

La forma de transmisión se muestra en la figura 2.12.

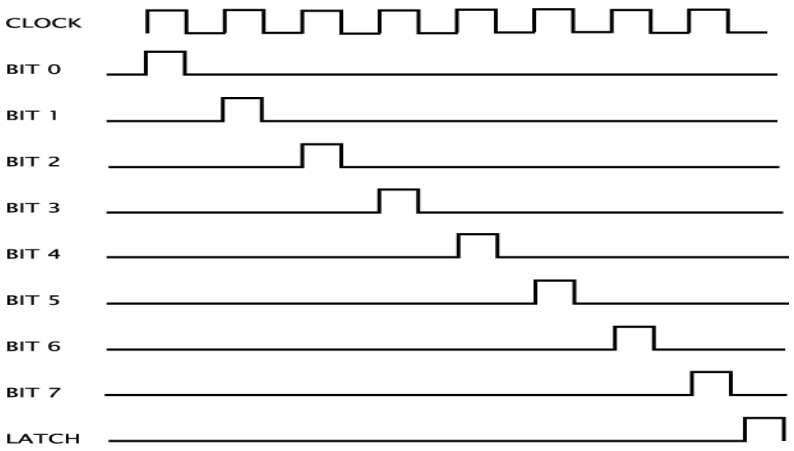


Figura 2. 12 Serialización de los bytes de dirección.

Como puede verse en la figura anterior, hasta que se ha enviado el byte completo se procede a enviar un pulso de habilitación (latch), el cual indica que se ha completado la transmisión y que el byte ya esta disponible para ser utilizado como una dirección.

### 2.5.3 Etapa de alimentación y control de buses

La alimentación del módulo electrónico requiere la generación de tres niveles de voltajes, un nivel para manejar la lógica TTL de 5V y 0V, otro nivel de 12V para manejar los reles y un nivel más que sirve para alimentar el servidor modbus, el cual es de 3.3V.

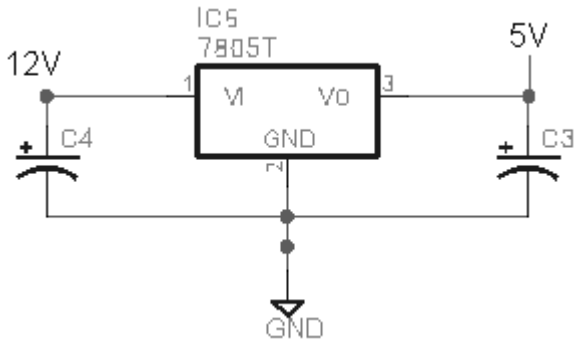


**Figura 2. 13** Generación de 5V para alimentación.

El módulo electrónico es alimentado a través de una fuente externa que provee 12V, y es a partir de este nivel que se utilizan reguladores de voltaje para generar los 5V y 3.3V, la figura 2.13 muestra la generación de 5V.

Los 12V pasan directamente a cada bloque de expansión junto con las líneas de los buses de transmisión de datos, de esta forma se envían a cada bloque 6 líneas, 4 líneas de datos, una línea de tierra y la línea de 12V.

El módulo electrónico esta protegido de errores de conexión de la alimentación tales como colocar al revés las patas de la alimentación, para ello cuenta con un diodo protector que evita que el módulo electrónico sea alimentado al revés.

El consumo de energía se ha reducido de forma considerable al mantener en estado de reposo todos los bloques de expansión y solo se activa el que es diseccionado en un momento determinado. Para lograr esto se ha utilizado los pines de habilitación en los multiplexores y de-multiplexores de forma que con el byte de dirección se genera un bit específico que activa al mux o demux al instante de la lectura de una entrada o al momento de forzar una salida.

El consumo de corriente del módulo electrónico es de 200 mili amperios. Cada módulo de expansión consume en reposo 23 mA.. Este consumo aumenta solo unos 5 mA si el módulo electrónico tiene una demanda de clientes. El total de corriente que consume en el caso mas critco es de 230 mA @ 12 Voltios, esto es, 2.76 Watts y es aproximadamente constante sin importar el número de cargas que se manejen ya que el acceso a cada una se hace una a la vez.

## 2.6 Diseño e implementación de los bloques de expansión

Las entradas y salidas del módulo electrónico se utilizan de una forma diferente a como normalmente se utilizan los pines de un microcontrolador. En un microcontrolador se tiene una reducida cantidad de pines para el manejo de cargas, y utilizándolos uno por cada carga se tendría control de no más de 30 cargas en un microcontrolador de gama media. Es por esta razón que el módulo electrónico se ha diseñado utilizando una sencilla pero eficiente configuración de pines de tal forma que con los pocos pines disponibles en el microcontrolador se puede controlar cientos de cargas.

Esta forma de incrementar la capacidad del módulo electrónico se conoce como multiplexación. Una característica importante del módulo electrónico es que su diseño se basa en dicha multiplexación y por ende todo su funcionamiento.

Existen varias tecnologías para realizar la multiplexación, en este caso se ha usado la mas sencilla y económica, dicha multiplexación se logra utilizando lógica TTL como multiplexores de-multiplexores y de-serializadores que permiten un aceptable incremento de entradas y salidas.

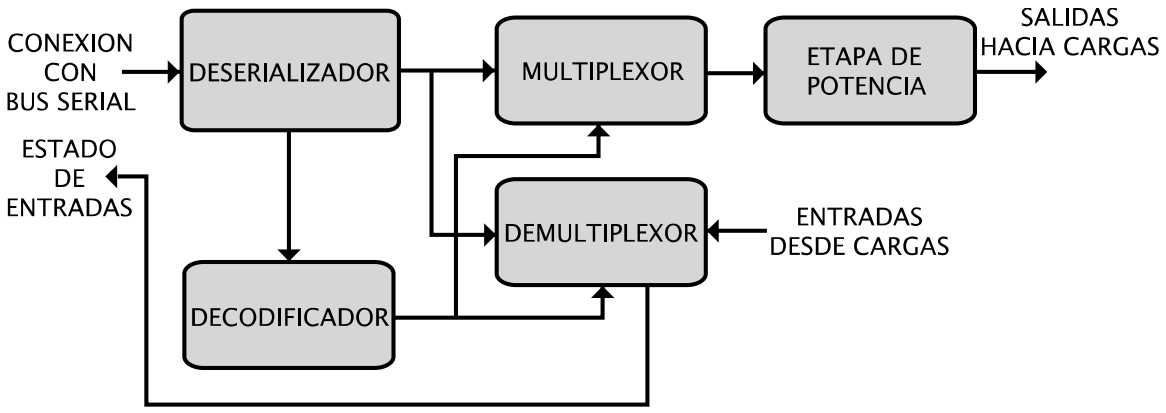


**Figura 2. 14** Diagrama de los bloques de expansión.

Los bloques de expansión están compuestos por varias partes como se muestra en la figura 2.14, donde cada una de ellas realiza tareas específicas como de-serialización de las direcciones recibidas, multiplexación y de-multiplexación y decodificación de bloques. A continuación se detalla cada una de estas partes.

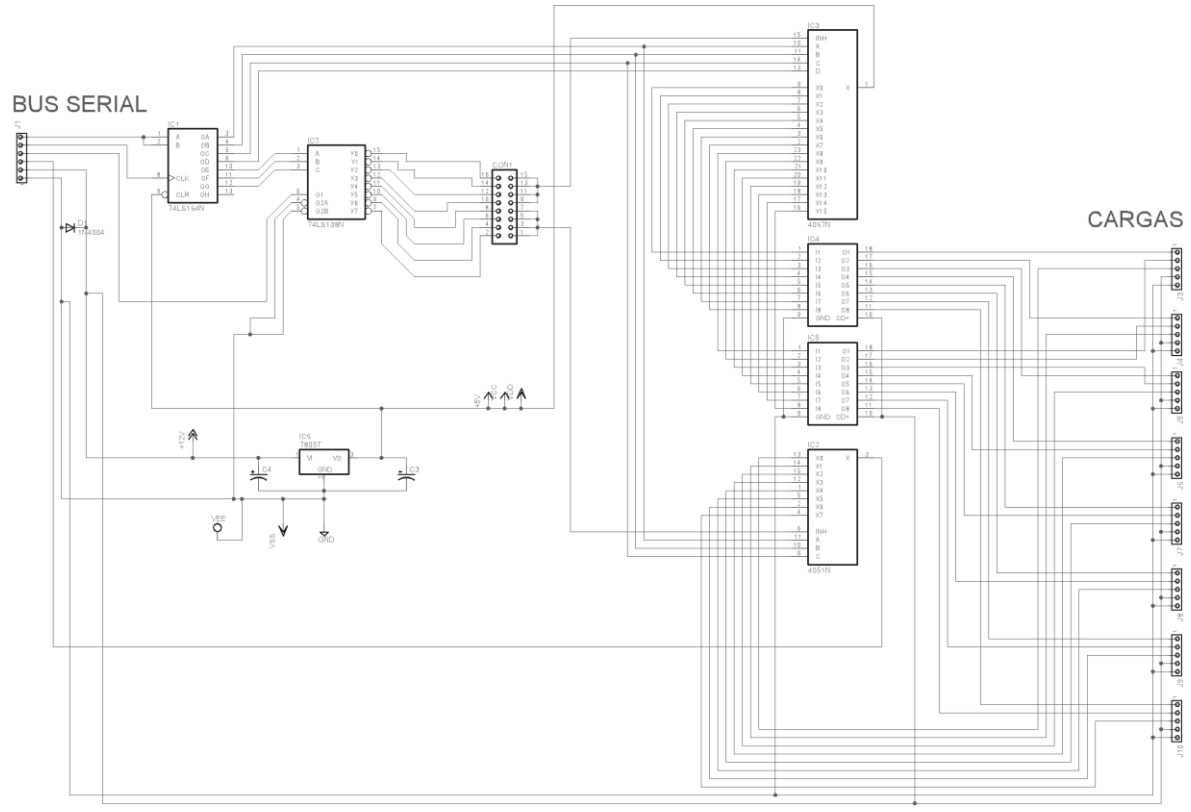


**Figura 2. 15** Diagrama esquemático de los bloques de expansión.

### 2.6.1 Deserializador

Los bloques de expansión reciben patrones de dirección que deben convertir de serie a paralelo para poder usarlas como líneas de control en las etapas de multiplexación y de-multiplexación. La figura 2.16 muestra como se convierten las direcciones a serial para ser enviadas a través de los buses de comunicación.

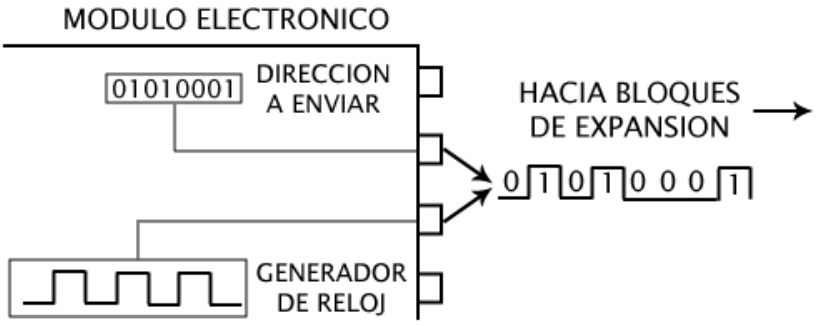


**Figura 2. 16** Conversión de una dirección a forma serial.

Para poder convertir dichos patrones desde la forma serial que se reciben en los buses a forma paralela se utiliza un componente de de-serialización, este componente toma cada bit que recibe y los ordena en forma paralela en sus salidas, cada último bit recibido pasa a ser el menos significativo.

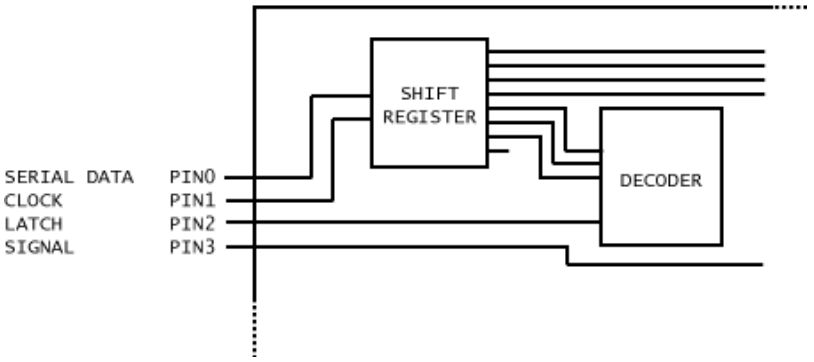


**Figura 2. 17** Identificación de pines en el bloque de expansión.

Como se observa en la figura 2.17, el pin0 se utiliza para transmitir serialmente la dirección que se desea acceder en los bloques de expansión. El pin1 contiene la señal de reloj la cual determina la velocidad a la que se trasmite el tren de bits contenido en el pin0. El pin2 contiene la señal que activa los multiplexores y de-multiplexores permitiendo que sea accedida la salida o entrada especificada en la dirección enviada en el pin0. Por ultimo el pin3 se utiliza para las entradas, toda entrada direccionada dentro del bus envía su estado en este pin3.

Como es obvio, este mecanismo de transmisión permite usar al máximo cada salida o entrada del microcontrolador, pero tiene algunos puntos en contra que vale la pena considerar; a continuación se enlistan las ventajas y desventajas que tiene con respecto a otras formas de transmisión.

**Tabla 2. 6** Ventajas y desventajas de una transmisión serie.

| VENTAJAS | DESVENTAJAS |
|---|---|
| • Expande cada pin del microcontrolador en una relación 1:8, 1:16 o mas dependiendo de la etapa de de-serialización.<br>• Protege las salidas del microcontrolador de sobre-corrientes al mantenerlas a niveles bajos.<br>• Se puede utilizar lazos de corriente para llevar la salida serial a grandes distancias. | • Los pines del microcontroladores dejan de ser bidireccionales.<br>• Si no se polariza adecuadamente puede ser más susceptible al ruido. |

El componente utilizado para este propósito es un shift register, es un circuito electrónico digital de manejo se comunicaciones seriales, este componente convierte un tren de bits entrante digital en una palabra de ocho bits paralelos a la salida. Esto es útil para utilizar una menor cantidad de pines del microcontrolador en el direccionamiento de los bloques de expansión. El componente utilizado es el SN74LS164, en los anexos se dan los detalles de este elemento.

### 2.6.2 Decodificador

La decodificación de bloques se utiliza para seleccionar entre varios bloques de expansión conectados al mismo bus, esto permite que un solo bus pueda contener varios bloques dependiendo del tamaño del decodificador.

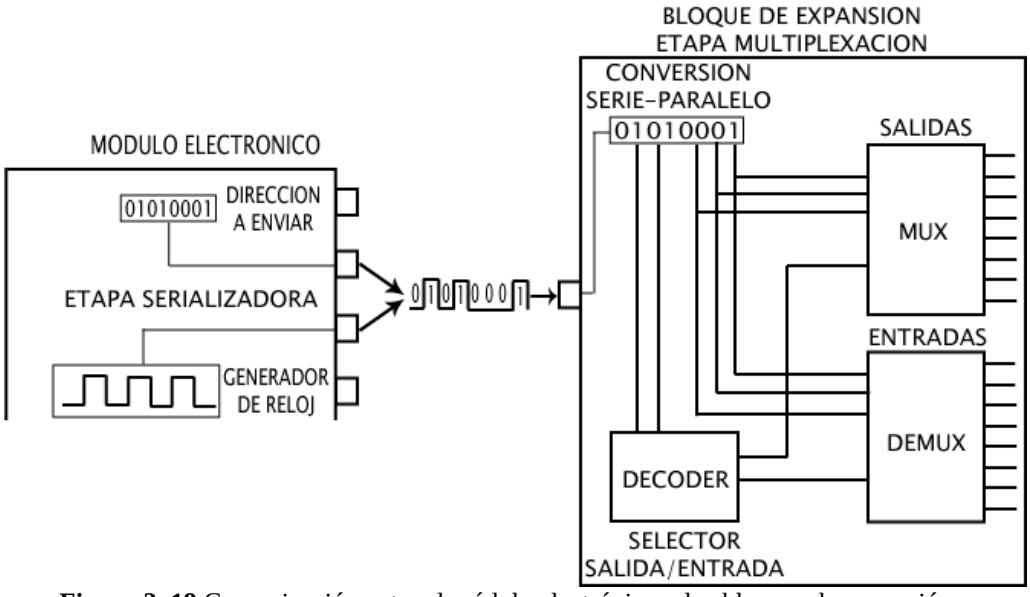


**Figura 2. 18** Comunicación entre el módulo electrónico y los bloques de expansión.

El elemento que se observa en la figura 2.18 llamado "decoder" es en efecto u decodificador de direcciones, usado para seleccionar ya sea al multiplexor (mux) o al de-multiplexor (demux), esto se debe a que las mismas líneas que salen del conversor serie a paralelo son las que van a servir de control tanto para el mux como también para el demux.

De esta forma es como se lleva a cabo la multiplexación de salidas y entradas en el hardware del SACME. Para ver un poco más el mecanismo de selección de bloques se muestra en la figura 2.19 el decoder y junto a cada salida una línea cortada, estas líneas cortadas se pueden unir mediante puentes, pero solamente debe seleccionarse dos de ellas en cada bloque de expansión. Se utiliza las primeras cuatro para seleccionar las salidas, la primera línea para las salidas del primer bloque, la segunda línea para las salidas del segundo bloque y así sucesivamente hasta la cuarta línea. Las líneas quinta hasta la octava se utilizan para seleccionar las entradas en cada bloque de la misma forma en que se hace con las salidas.

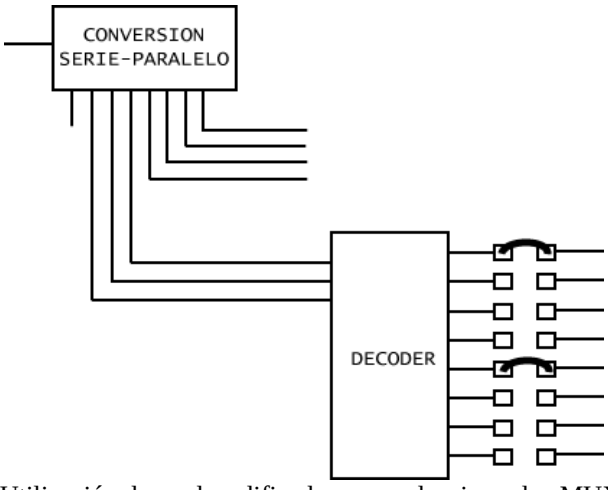


**Figura 2. 19** Utilización de un decodificador para seleccionar los MUX y DEMUX.

Como lo muestra la figura 2.19, se utiliza un decoder de 3 a 8 líneas para seleccionar los mux o demux en un momento determinado, para ello se utilizan los bits mas significativos de la dirección proporcionada desde el módulo electrónico, en la figura se observa también que los primeros 4 bits se envían directamente para controlar los mux y demux, con estas cuatro líneas se direcciona hasta 16 salidas o entradas en cada mux o demux. Otro punto importante es que sobra un bit, el mas significativo del patrón de dirección, este bien podría ser utilizado junto a los otros para controlar un decoder de 16 salidas y así poder direccionar muchos mas elementos.

La colocación de los puentes a las salidas del decoder se debe hacer al momento de anexar el bloque de expansión a un bus del módulo electrónico, esto es muy importante ya que solo debe haber dos puentes en cada bloque y en ningún caso se debe repetir la conexión en otro bloque, de lo contrario se direccionaria dos o mas bloques al mismo tiempo, causando una confusión en los datos y hasta un posible cortocircuito en el bus.

Un detalle muy importante relacionado a la multiplexación de pines es que estos no están permanentemente comunicados con el microcontrolador, y solo reciben atención cuando el microcontrolador hace un cambio sobre ellos, ya sea para establecer un estado diferente a las salidas o para leer el estado de alguna entrada. Esto condiciona la forma en que se manejan las salidas ya que son manejadas de forma momentánea, debiendo estas mantener el estado que se les ha sido asignado. Para lograr este acometido se hace uso de dispositivos de manejo de carga con memoria de estado. Existe en el mercado algunos dispositivos tales como los reles que solo necesitan mantener el estado deseado durante algunos milisegundos, y una ves que lo memorizan lo mantienen aun en ausencia total de electricidad. Estos reles se conocen como “Latching Relays”, y son muy útiles en este tipo de aplicaciones, ya que no es necesario mantenerlos energizados todo el tiempo, por lo que reducen su consumo de energía.

### 2.6.3 Multiplexor

El multiplexor es un elemento que se utiliza para la selección de una de varias salidas, dicha selección se lleva a cabo al poner una palabra de selección en los cuatro bits más significativos del de-serializador, esta conexión se muestra en la figura 2.20.

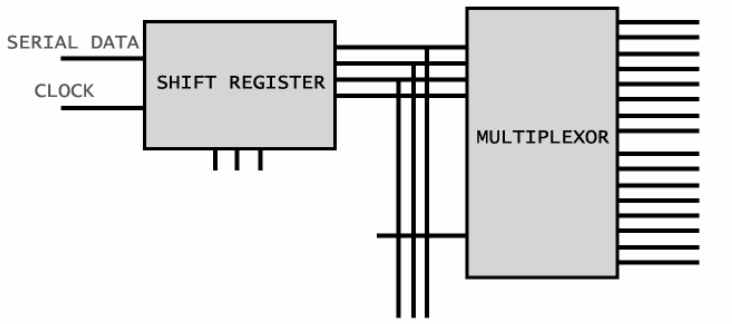


**Figura 2. 20** Salidas del de-serializador como control del multiplexor.

Como lo muestra la figura 2.20, el multiplexor utilizado tiene cuatro líneas de control y puede seleccionar entre 16 salidas.

### 2.6.4 De-multiplexor

El de-multiplexor realiza la función opuesta al multiplexor, este selecciona una de varias entradas y tiene una línea de salida a donde manda en valor de la entrada seleccionada.
El de-multiplexor utilizado en este caso es uno con tres líneas de control y 8 líneas de entrada.

La figura 2.21 muestra el de-multiplexor conectado a las mismas líneas que el mux. Lo que permite diferenciar entre el mux y demux es el decodificador visto anteriormente.

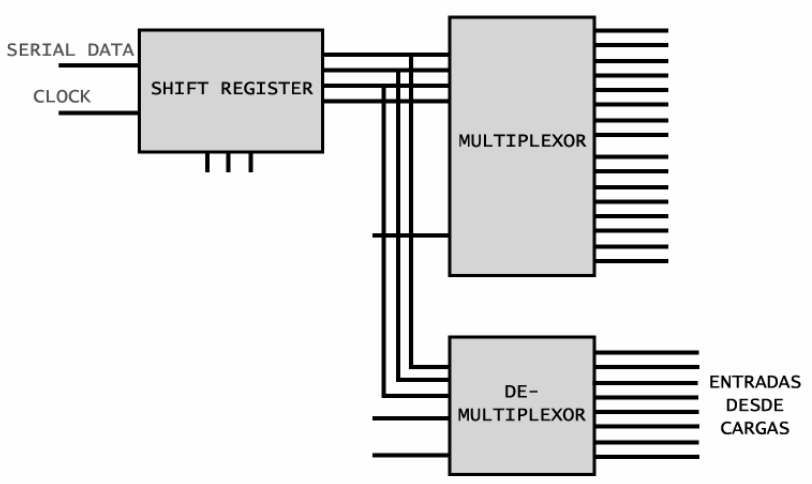


**Figura 2. 21** MUX y DEMUX compartiendo las mismas líneas de control.

### 2.6.5 Etapa de potencia

La etapa de potencia se utiliza para manejar las corrientes de las bobinas de los reles a partir de señales digitales de provienen del mux.

Estas etapas de potencia consisten de arreglos darlington de transistores, capaces de manejar corrientes altas. En los anexos se dan detalles de la capacidad de estos dispositivos.

La figura 2.22 muestra el arreglo darlington con transistores utilizado en cada salida del multiplexor.

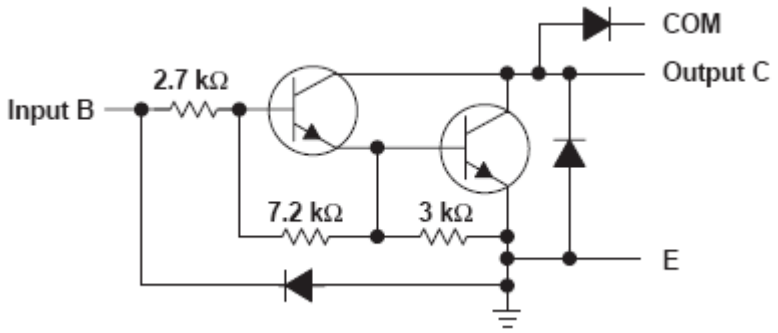


**Figura 2. 22** Arreglo darlington de transistores.

El diseño completo de los bloques de expansión se muestra en la figura 2.23, en el que se observa todos los componentes y su comunicación con el resto del circuito.

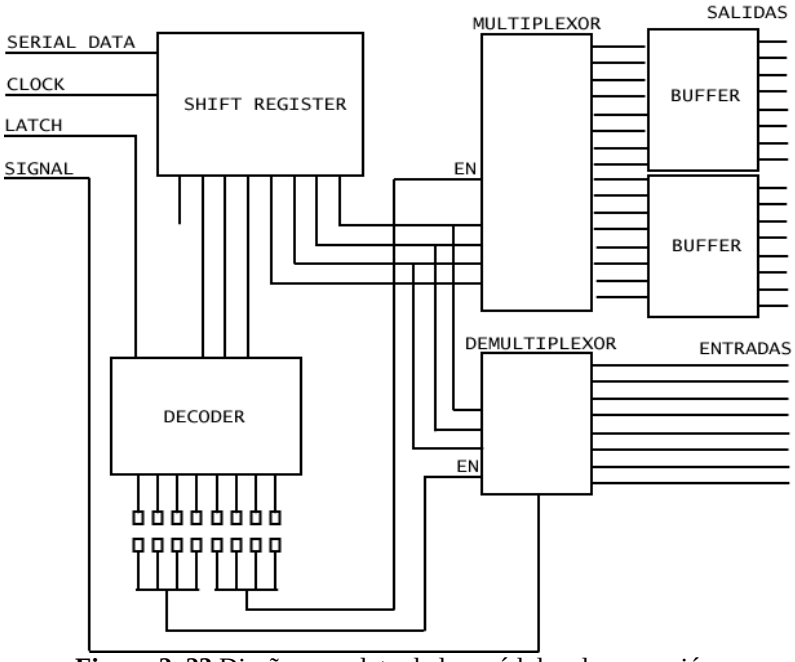


**Figura 2. 23** Diseño completo de los módulos de expansión.

El diagrama muestra una doble cantidad de salidas con respecto a las entradas, esto es así debido a que se utiliza en las salidas reles activados por pulsos, los cuales requieren de dos pulsos para operar, el pulso de set y uno de reset.

## 2.7 Diseño e implementación de los bloques de manejo de carga

Los bloques de manejo de carga son la parte final de la cadena de bloques que componen el hardware del SACME, estos elementos se utilizan para el manejo directo de las cargas, en estos elementos se manejan los reles para activar y desactivar cargas, además contienen la circuiteria necesaria para generar un estado lógico a partir del estado de las cargas, sea este encendido o apagado.

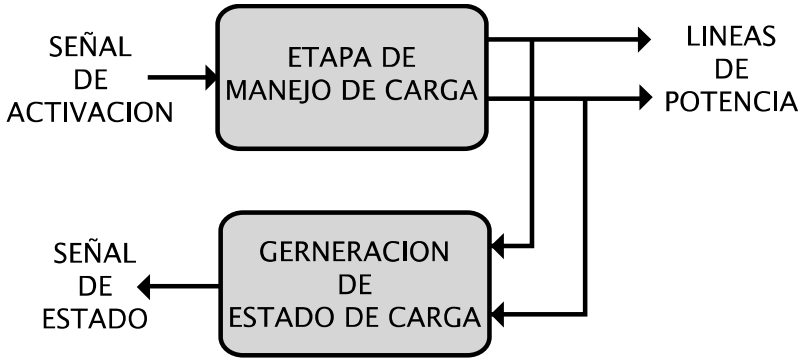


**Figura 2. 24** Diagrama de los bloques de manejo de carga.

Los bloques de manejo de carga se implementan utilizando una tecnología de reles que son activados a través de pulsos de corta duración, esto genera un ahorro de energía, ya que no es necesario mantener energizados los reles todo el tiempo. Esto es inherente a la tecnología de multiplexación ya que no se le puede dar una atención permanente a cada carga sino que se maneja temporalmente y solo cuanto se necesita obtener el estado de dicha carga o se desea forzar su estado.

Debido a que el diseño se ha enfocado en el manejo de sistemas de iluminación, se ha adaptado los bloques de manejo de cargas al funcionamiento de un switch de tres vías, de esta forma se controla una luminaria desde dos puntos, un punto local con interrupción manual y el otro punto se controla remotamente por medio de SACME.

La figura 2.26 muestra un ejemplo de la conexión que se hace para manejar un foco mediante los bloques de manejo de cargas.

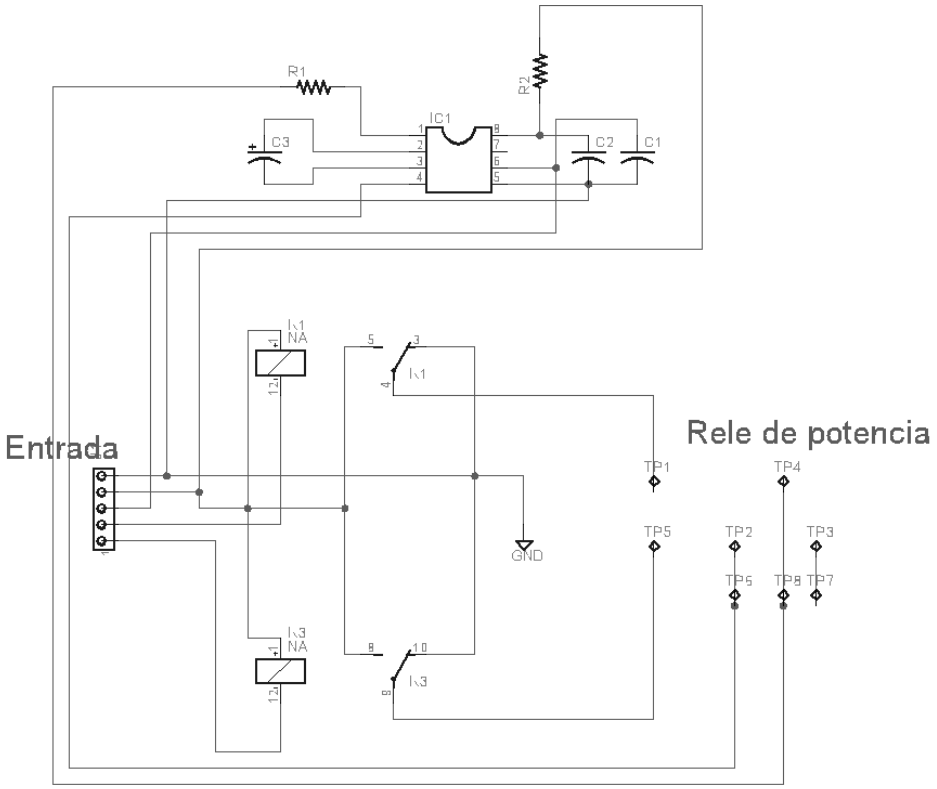


**Figura 2. 25** Diagrama esquemático de los bloques de manejo de carga.

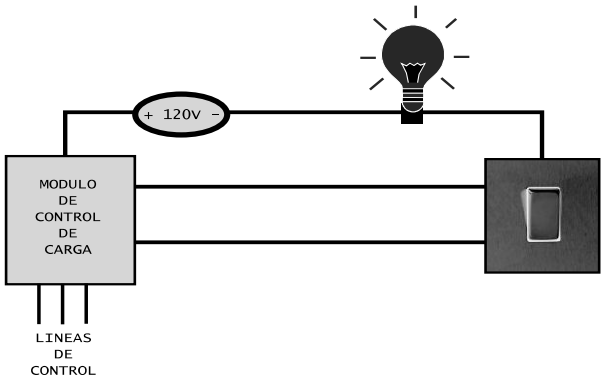


**Figura 2. 26** Ejemplo de una carga controlada por el SACME.

Como se observa en la figura, el control de una carga se hace de dos formas, mediante un switch convencional y mediante un bloque de manejo de carga. Estos bloques tienen la capacidad de manejar cargas de hasta 14 Amperios.

La circuiteria para generar el estado de las cargas se muestra en la figura 2.27, aquí se puede observar que a partir de la alimentación de la carga se coloca un "sensor" de voltaje,

el cual devuelve un estado positivo si la carga esta activada y un estado lógico cero si la carga esta apagada.

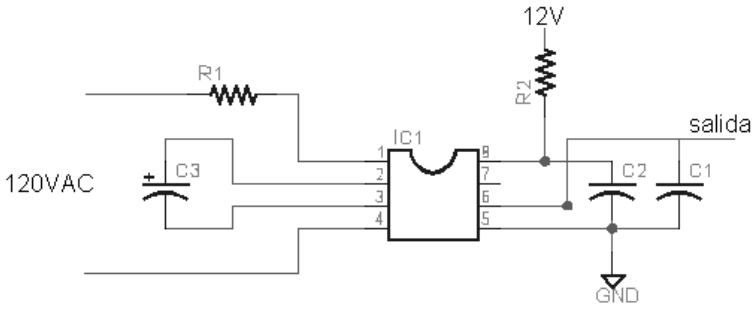


**Figura 2. 27** Circuito de generación de estados de las cargas.

# CAPITULO III

# DISEÑO E IMPLEMENTACION DE SITIO WEB

## Introducción

Los primeros sitios web no eran sino colecciones de páginas web enlazadas entre ellas por el lenguaje html. Hoy en día, los sitios web incluyen multimedia, aplicaciones de comercio electrónico, y otro tipo de sofisticadas aplicaciones basadas en Web como voz IP y banca online. Además de los avances en contenido, se han producido avances en las tecnologías de servidores web, como el desarrollo de servidores de aplicaciones. Dentro de esta familia de aplicaciones o herramientas, hay que destacar el papel que la tecnología java proporciona a este tipo de desarrollo, junto con el apoyo del mundo de condigo abierto (del ingles open-source).
En este capitulo se describe detalladamente las partes involucradas en el diseño y la implementación del Sitio Web tomando criterios de diseño, basados en tecnología java.

La aplicación web, es la que permite a un usuario a través de una interfaz grafica amigable, administrar, controlar y monitorear energía eléctrica que son las diferentes opciones que el sistema SACME proporciona.

## 3.1 Diseño del Sitio Web

### 3.1.1 Aspectos Generales en Arquitectura WEB

A la hora de abordar el desarrollo del sitio web, hay una serie de consideraciones acerca del mismo que hay que tener muy presentes, dado que son claves en el tipo de diseño y metodologías de desarrollo a aplicar.

#### 3.1.1.1 Escalabilidad

Es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos. En general, también se podría definir como la capacidad del sistema de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes. Por ejemplo la implementación de una nueva funcionalidad del sistema o la incorporación de más hardware, o también los servicios pueden separarse en distintos puntos en la red.

#### 3.1.1.2 Separación de responsabilidades

Se logra a través de la separación en capas del sistema. Distintas responsabilidades no deben ser delegadas en la misma clase, y llevado esto algo más allá, en el mismo conjunto de clases. En la actualidad, la tendencia más aceptada es la aplicación de patrones de diseño de arquitectura que dividen la responsabilidad en distintas capas que interaccionan unas con otras a través de sus interfaces. Se trata de los sistemas denominados multicapa o *n*-capas. Aplicados a los proyectos web, el modelo más básico es el de aplicaciones 3 capas: presentación, negocio o dominio y capa de acceso a datos.

#### 3.1.1.3 Portabilidad

En la medida de lo posible, la aplicación web debe poder adaptarse a las distintas posibles arquitecturas físicas susceptibles de ser empleadas para el despliegue del paquete, limitándose en la medida de lo posible el impacto de tal adaptación a tareas de configuración, y evitándose así la necesidad de modificar el código de la misma ante dichas situaciones.

#### 3.1.1.4 Gestión de la sesión del usuario, cacheado de entidades

Con objeto de limitar en la medida de lo posible los accesos innecesarios a memoria secundaria (bases de datos, ficheros externos de configuración, etc), se propone un sistema que se apoya en parte en el empleo de la sesión HTTP(s) para cachear ciertos datos referentes a la sesión del usuario, o bien comunes a todas las sesiones de usuario. Obviamente, la cantidad y naturaleza de las entidades susceptibles de ser cacheadas será determinada teniendo muy presentes aspectos de rendimiento del producto, dado que un empleo abusivo de esta técnica puede y suele llevar a un consumo excesivo de los recursos del sistema (memoria).

#### 3.1.1.5 Aplicación de patrones de diseño

El empleo y aplicación de patrones de diseño facilita el entendimiento del código y, por tanto, reduce considerablemente el coste de mantenimiento, dado que además de aportar soluciones eficientes para problemas comunes, son muy interesantes como medio de entendimiento entre diseñadores e implementadores.

### 3.1.2 Separación Lógica en capas

A la hora de plantear el diseño de la aplicación web, el primer paso es conseguir separar conceptualmente las tareas que el sistema debe desempeñar entre las distintas capas lógicas y en base a la naturaleza de tales tareas se ha de partir de la separación inicial en tres capas, diferenciando que proceso de los que hay que modelar responde a tareas de presentación, cual a negocio y cual a acceso a datos. En caso de identificar algún proceso lógico que abarque responsabilidades adjudicadas a dos o más capas distintas, es probable que dicho

proceso deba ser explotado en subprocesos iterativamente, hasta alcanzar el punto en el que no exista ninguno que abarque más de una capa lógica.

#### 3.1.2.1 Capa de presentación

Es la responsable de todos los aspectos relacionados con la interfaz de usuario de la aplicación. Así, en esta capa de resuelven cuestiones como:

- Navegabilidad del sistema, mapa de navegación, etc.
- Formateo de los datos de salida: Resolución del formato más adecuado para la presentación de resultados. Está relacionado directamente con la internacionalización de la aplicación.
- Internacionalización: Los textos, etiquetas, y datos en general a presentar se obtendrán de uno u otro fichero de recursos en base al idioma preferido del navegador del usuario. En base a esta condición se ven afectadas las representaciones numéricas, las validaciones sobre los datos de entrada (coma decimal o punto decimal) y otros aspectos relativos al idioma del usuario remoto.
- Validación de los datos de entrada, en cuanto a formatos, longitudes máximas, etc.
- Interfaz gráfica con el usuario.
- Multicanalidad de la aplicación: Una misma aplicación web puede contar con varias presentaciones distintas, determinándose el uso de la adecuada en base al dispositivo visualizador desde el que trabaje el usuario. Así, no se representará la misma información con el mismo formato en un navegador web estándar que en un dispositivo móvil provisto de un navegador WAP[6].

#### 3.1.2.2 Capa de negocio

En esta capa es donde se deben implementar todas aquellas reglas obtenidas a partir del análisis funcional del proyecto. Así mismo, debe ser completamente independiente de cualquiera de los aspectos relacionados con la presentación de la misma. De esta forma, la misma capa de negocio debe poder ser empleada para una aplicación web común, una aplicación WAP, o una StandAlone. Por otro lado, la capa de negocio ha de ser también completamente independiente de los mecanismos de persistencia empleados en la capa de acceso a datos. Cuando la capa de negocio requiera recuperar o persistir entidades o cualquier conjunto de información, lo hará siempre apoyándose en los servicios que ofrezca la capa de acceso a datos para ello. De esta forma, la sustitución del motor de persistencia no afecta lo más mínimo a esta parte del sistema. Debería poder reemplazarse el gestor de

---

[6] **Wireless Application Protocol** o **WAP** (protocolo de aplicaciones inalámbricas) es un estándar abierto internacional para aplicaciones que utilizan las comunicaciones inalámbricas, p.ej. acceso a servicios de Internet desde un teléfono móvil.

bases de datos por un conjunto de ficheros de texto sin necesitar tomar ni una línea de código de presentación o negocio. Las responsabilidades que conviene abordar en esta capa son:

#### 3.1.2.2.1 Implementación de los procesos de negocio identificados en el análisis del proyecto.

Como se deduce del párrafo anterior, los procesos de negocio implementados en esta capa son totalmente independientes de cualquier aspecto relativo a la presentación de los mismos. Pongamos por ejemplo la generación de un informe que conste de varias filas las cuales deberán sombrearse con un color determinado por la superación o no de ciertos umbrales para ciertos valores. La aplicación de ciertos umbrales, y la consecuente determinación de la situación (correcta, incorrecta, etc) de cada valor de la fila es un cálculo de negocio que debe afrontarse en esta capa. Sin embargo, sería en un error completar un campo adicional en el que, como resultado del cálculo, se determinara directamente el color de la fila. Lo adecuado es codificar la situación de la misma y, una vez en presentación, determinar el color adecuado en base al código recibido. De esta forma, si el usuario requiere que a partir de cierta fecha, el color rojo sea sustituido por el naranja, la modificación de este aspecto de presentación de información se limitaría en la aplicación a una modificación de la capa de presentación.

#### 3.1.2.2.2 Control de acceso a los servicios de negocio.

Dado que una misma aplicación puede contar con más de una capa de presentación al mismo tiempo, es aconsejable que la responsable última de ejecutar tareas sobre el control de acceso a los servicios del sistema no sea la capa de presentación, sino la de negocio. Los implementadores de la capa de negocio ni pueden ni deben confiar en que las futuras implementaciones de nuevas capas de presentación gestionen adecuadamente el acceso a los servicios de negocio. De esta forma, en cada invocación a cualquier método restringido de negocio se deberá comprobar por medio del sistema de autentificación adecuado, los derechos del usuario actual a realizar tal operación.

#### 3.1.2.2.3 Publicación de servicios de negocio

El lugar adecuado para que dos microaplicaciones o aplicaciones completas interactúen es a nivel de la capa de negocio. Así mismo, el modelo de colaboración recomendado en esta definición de arquitectura es el que se basa en el empleo de servicios web. De esta forma, la capa de negocio ofrecerá dos vistas alternativas, dado que por un lado el conjunto de façades[7] con presentación ofrecerá a esta los servicios que se requieran para el funcionamiento de la microaplicación en sí, y por otro, otro conjunto de servicios serán publicados por medio (recomendablemente) de servicios web. Estos últimos estarán orientados a la colaboración con otros sistemas distintos u otras microaplicaciones pertenecientes al mismo proyecto, y deberán por tanto llevar un control más férreo sobre el acceso al servicio, dado que ahora, además de hacerlo a nivel de usuario, puede que sea

[7] El patrón de diseño *Facade* sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas.

necesario hacerlo a nivel de aplicación, por lo que cada aplicación remota debería estar explícitamente autorizada a invocar el servicio de negocio publicado.

#### 3.1.2.2.4 Invocación a la capa de persistencia

Los procesos de negocio son los que determinan que, como y cuando se debe persistir en el repositorio de información. Los servicios ofertados por la interfaz de la capa de acceso a datos son invocados desde la capa de negocio en base a los requerimientos de los procesos en ella implementados.

### 3.1.2.3 Capa de acceso a datos

La capa de acceso a datos es la responsable de la gestión de la persistencia de la información manejada en las capas superiores. En esta capa, se definen las entidades o tablas para almacenar los datos de forma relacional, En un modelo académicamente purista, la interfaz de esta capa estaría compuesta por vistas de las entidades a persistir, pero a efectos prácticos, y con objeto de aprovechar la habitual potencia de los gestores de bases de datos, la interfaz muestra una serie de servicios que pueden agrupar operaciones en lo que se puede denominar "lógica de persistencia", como *insertar usuario* o *inserción de roles*, en la que podrían darse de alta al mismo tiempo un Rol y todos las entidades que dependan de dicho rol (porque no, el mismo usuario).

### 3.1.2.4 Capa de infraestructura

Esta capa, adyacente a todas las demás de la aplicación, comprende todos aquellos servicios susceptibles de ser requeridos desde cualquiera de las capas lógicas de la aplicación web. La gestión de un servicio y de las clases que lo implementan se realizará desde la concepción de un componente, es decir, una clase totalmente independiente de la aplicación que lo utiliza. La capa de infraestructura estará formada entonces por componentes, y por las clases gestoras necesarias para su configuración y gestión. De esta forma, cuando una clase de cualquiera de las capas lógicas de la aplicación requiera el uso de alguno de los servicios ofrecidos por la capa de infraestructura (por ejemplo, el servicio de Log), no tratará directamente con la clase que implemente tal servicio, sino que lo hará por medio del interfaz que cumpla la misma. Así mismo, la instanciación del servicio es responsabilidad de las clases gestoras de la capa de infraestructura. La clase cliente del servicio le pedirá a la capa de infraestructura que le facilite una instancia de la clase que implementa el servicio que necesita. La relación entre una interfaz que defina un servicio de infraestructura y la clase que implementa el servicio se establece en un fichero externo XML. De esta forma:

• La sustitución de un componente que implemente un servicio de la capa de infraestructura por otro distinto que cumpla la misma interfaz sólo requiere modificar el fichero de configuración.

• La configuración de cada uno de los componentes irá asimismo externalizada en ficheros XML.

• Se consigue desacoplar completamente la aplicación de su entorno de despliegue. En caso de que en un futuro tuviera que ser integrada con otros sistemas, dicha tarea podría llegar a requerir sólo el desarrollo de los componentes adecuados o en encapsulamiento de los ya presentes en el sistema anfitrión. Volviendo al ya citado ejemplo del sistema de Log, es habitual que una compañía tenga normalizado el formato de salida del mismo para todos sus sistemas. Si se necesitará instalar el producto en otra compañía, que probablemente cuente también con su propio formato de trazas de Log, sólo se necesitaría encapsular las clases aportadas por la nueva compañía para la generación de trazas para adaptar su interfaz al que el servicio de la aplicación impone. Si además se tendiera al empleo de interfaces estándar (aunque esto no siempre es posible), esta tarea puede quedar reducida a una simple re-configuración del sistema.

• Las clases gestoras, en caso de que el comportamiento del componente lo permitiera, pueden trabajar con pools[8] de componentes para aquellos cuyo uso no implique un mantenimiento de estado, y sean susceptibles de invocarse con una frecuencia elevada.

• Las clases gestoras de la capa de infraestructura deben permitir establecer períodos de re-configuración, de forma que la alteración del comportamiento del sistema a través de sus componentes (sustitución y configuración de los mismos) se pueda hacer en caliente, evitando una parada en el sistema de producción. La modificación del comportamiento de ciertos componentes, como por ejemplo un pool de conexiones, facilita el sincronismo y dimensionamiento del sistema una vez entre en producción.

[8] En computación, se denomina **pool** (agrupamiento de conexiones) al manejo de una colección de conexiones abiertas a una base de datos de manera que puedan ser reutilizadas al realizar múltiples consultas o actualizaciones.

Casos habituales de servicios que deben según esta filosofía pertenecer a la capa de infraestructura son:

- Servicio de Log.
- Pool de conexiones JDBC[9] (o de cualquier otro sistema de persistencia).
- Sistema de configuración de la aplicación.
- Gestor de accesos/permisos de usuario a los distintos servicios de la aplicación.
- Otros más específicos del entorno del proyecto pero independientes del modelo.

### 3.1.3 Casos de Uso del SACME

Los casos de uso representan lo qué hace el sistema desde el punto de vista del usuario. Es decir, describen un uso del sistema y cómo este interactúa con el usuario. Esto nos permite crear clases especializadas para cada caso de uso y poder independizarlas de las capas adyacentes.

A continuación se presenta el diagrama de casos de uso del SACME.

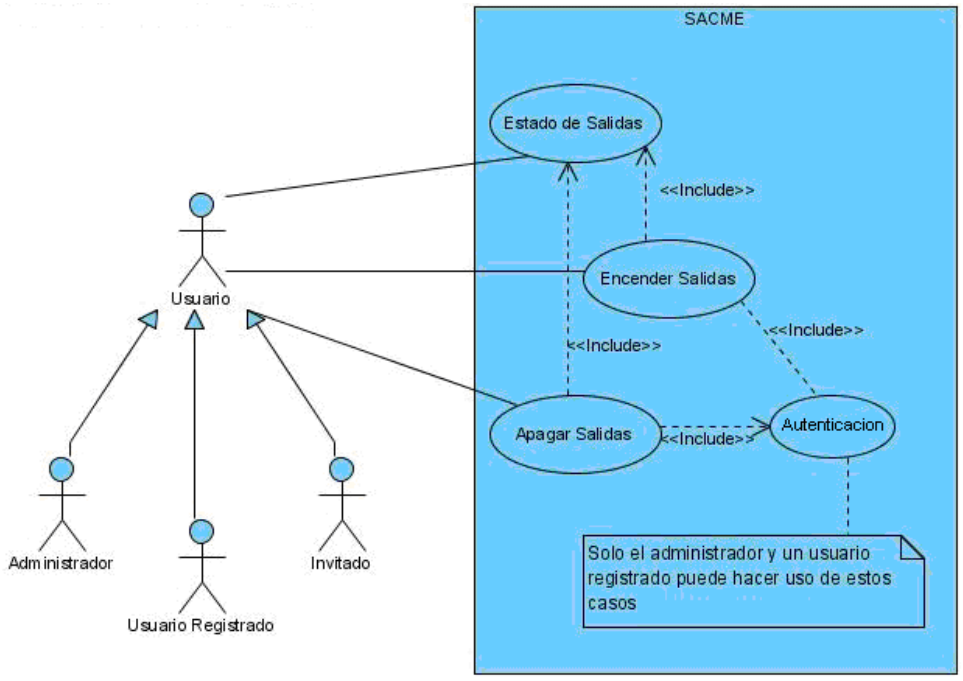


**Figura 3. 1** Casos de Uso del Sacme

A continuación describiremos los posibles casos de uso del sacme.

[9] **JDBC** es el acrónimo de *Java Database Connectivity*, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema de operación donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

#### 3.1.3.1 Caso de Uso: Estado de Salidas y sus escenarios

Continuación se muestra el desarrollo del caso de uso estado de salidas y sus escenarios.

**Tabla 3. 1** Detalle del escenario ES 1.1

| **Identificación escenario** | **ES 1.1** |
|---|---|
| **Nombre del escenario** | Ver el estado de una salida. |
| **Precondición** | El usuario accede al sistema a través de una aplicación web y hace la respectiva selección. |
| **Poscondición:** | Visualiza el estado de una salida. |

**Tabla 3. 2** Detalle del escenario ES 1.2

| **Identificación escenario** | **ES 1.2** |
|---|---|
| **Nombre del escenario** | Ver el estado de un grupo de salidas |
| **Precondición** | El usuario accede al sistema a través de una aplicación web y hace las respectivas selecciones. |
| **Poscondición:** | Visualiza el estado de un grupo de salidas. |

**Tabla 3. 3** Detalle del caso de uso Estado de Salidas.

| **CU: 1** | **Estado de Salidas** | |
|---|---|---|
| **Descripción:** | SACME deberá mostrar al usuario, una lista de salidas disponibles donde se permitirá seleccionar una o mas salidas, las que posteriormente se mostrara su estado. | |
| **Actor:** | Usuario a través de un navegador de páginas web. | |
| **Precondiciones:** | El Usuario debe acceder al sistema a través de la interfaz de usuario. | |
| **Secuencia Normal:** | **Paso** | **Acción** |
| | 1 | El Cliente accede la interfaz de usuario a través de un navegador de páginas web. |
| | 2 | El Cliente selecciona de la lista las salidas necesarias y hace la petición a través de la interfaz de usuario. |
| | 3 | El sistema verifica el estado de cada salida seleccionada. |
| | 4 | Se muestra la respuesta. |
| **Poscondiciones:** | La operación ha sido realizada. | |

**Tabla 3. 4** Detalle del caso de uso Estado de Salidas (Continuación).

| **Excepciones:** | **Paso** | **Acción** |
|---|---|---|
| | 3’ | Connection refused: Error de conexión lo que indica que la conexión con el servicio controlador del módulo electrónico no esta inicializado. |
| | 3’ | Connection time out: Error de conexión, se debe a problema en el módulo electrónico, por ejemplo, no esta conectado a la red, esta apagado, mal configurado, además, de los propios problema de red ethernet. |
| | 3’’ | Modbus Slave Exception: el módulo electrónico se encuentra bloqueado y tendrá que ser reiniciado manualmente. |
| | 4’ | Error Code = 1: Código de Función ilegal, El código función es desconocido por el módulo electrónico. |
| | 4’’ | Error Code = 2: Dirección de datos ilegal, esta dirección no está soportada en el módulo electrónico. |
| | 4’’’ | Error Code = 3: Valor de dato ilegal, depende de la petición y no corresponde a ella. |
| | 4’’’’ | Error Code = 4: Fallo servidor, se intento hacer la operación pero no pudo ser completada. |
| **Importancia:** | Vital | |

### 3.1.3.2 Caso de Uso: Encender Salidas y sus escenarios.

Continuación se muestra el desarrollo del caso de uso encender salidas y sus escenarios.

**Tabla 3. 5** Detalle del escenario ES 2.1

| **Identificación escenario** | **ES 2.1** |
|---|---|
| **Nombre del escenario** | Encender una salida. |
| **Precondición** | El usuario accede al sistema a través de una aplicación web y hace la respectiva selección. |
| **Poscondición:** | Una salida ha sido encendida. |

**Tabla 3. 6** Detalle del escenario ES 2.2

| **Identificación escenario** | **ES 2.2** |
|---|---|
| **Nombre del escenario** | Encender un grupo de salidas |
| **Precondición** | El usuario accede al sistema a través de una aplicación web y hace las respectivas selecciones. |
| **Poscondición:** | El grupo de salidas ha sido encendido. |

**Tabla 3. 7** Detalle del caso de uso Encender Salidas.

| **CU: 2** | **Encender Salidas** | |
|---|---|---|
| **Descripción:** | SACME deberá mostrar al usuario, una lista de salidas disponibles donde se permitirá seleccionar una o mas salidas, las que posteriormente serán encendidas. | |
| **Actor:** | Usuario a través de un navegador de páginas web. | |
| **Precondiciones:** | El Usuario debe estar registrado como usuario Registrado o como Administrador. | |
| **Secuencia Normal:** | **Paso** | **Acción** |
| | 1 | El Cliente accede la interfaz de usuario a través de un navegador de páginas web. |
| | 2 | El Cliente hace la petición a través de la interfaz de usuario. |
| | 3 | El sistema verifica el estado de cada salida seleccionada. |
| | 3.a | Si la salida esta encendida, no se hace ninguna operación. |
| | 3.b | Si la salida no esta encendida, se realiza la operación. |
| | 4 | El sistema verifica el nuevo estado de cada salida en la cual se hizo la operación. |
| | 5 | Se muestra la respuesta. |
| **Poscondiciones:** | La operación ha sido realizada. | |
| **Excepciones:** | **Paso** | **Acción** |
| | 2 | Usuario no registrado. El usuario debe estar registrado en el sistema. |
| | 3’ | Connection refused: Error de conexión lo que indica que la conexión con el servicio controlador del módulo electrónico no esta inicializado. |
| | 3’ | Connection time out: Error de conexión, se debe a problema en el módulo electrónico, por ejemplo, no esta conectado a la red, esta apagado, mal configurado, además, de los propios problema de red ethernet. |
| | 3’’ | Modbus Slave Exception: el módulo electrónico se encuentra bloqueado y tendrá que ser reiniciado manualmente. |
| | 5’ | Error Code = 1: Código de Función ilegal, El código función es desconocido por el módulo electrónico. |
| | 5’’ | Error Code = 2: Dirección de datos ilegal, esta dirección no está soportada en el módulo electrónico. |
| | 5’’’ | Error Code = 3: Valor de dato ilegal, depende de la petición y no corresponde a ella. |
| | 5’’’’ | Error Code = 4: Falló servidor, se intento hacer la operación pero no pudo ser completada. |
| **Importancia:** | Vital | |

### 3.1.3.3 Caso de Uso: Apagar Salidas y sus escenarios.

Continuación se muestra el desarrollo del caso de uso apagar salidas y sus escenarios.

**Tabla 3. 8** Detalle del escenario ES 3.1

| **Identificación escenario** | **ES 3.1** |
|---|---|
| **Nombre del escenario** | Apagar una salida. |
| **Precondición** | El usuario accede al sistema a través de una aplicación web y hace la respectiva selección. |
| **Poscondición:** | Una salida ha sido apagada. |

**Tabla 3. 9** Detalle del escenario ES 3.2

| **Identificación escenario** | **ES 3.2** |
|---|---|
| **Nombre del escenario** | Apagar un grupo de salidas |
| **Precondición** | El usuario accede al sistema a través de una aplicación web y hace las respectivas selecciones. |
| **Poscondición:** | El grupo de salidas ha sido apagado. |

**Tabla 3. 10** Detalle del caso de uso Apagar Salidas.

| **CU: 3** | **Apagar Salidas** | |
|---|---|---|
| **Descripción:** | SACME deberá mostrar al usuario, una lista de salidas disponibles donde se permitirá seleccionar una o mas salidas, las que posteriormente serán apagadas. | |
| **Actor:** | Usuario a través de un navegador de páginas web. | |
| **Precondiciones:** | El Usuario debe estar registrado como usuario Registrado o como Administrador. | |
| **Secuencia Normal:** | **Paso** | **Acción** |
| | 1 | El Cliente accede la interfaz de usuario a través de un navegador de páginas web. |
| | 2 | El Cliente hace la petición a través de la interfaz de usuario. |
| | 3 | El sistema verifica el estado de cada salida seleccionada. |
| | 3.a | Si la salida esta apagada, no se hace ninguna operación. |
| | 3.b | Si la salida no esta apagada, se realiza la operación. |
| | 4 | El sistema verifica el nuevo estado de cada salida en la cual se hizo la operación. |
| | 5 | Se muestra la respuesta. |
| **Poscondiciones:** | La operación ha sido realizada. | |

**Tabla 3. 11** Detalle del caso de uso Apagar Salidas (Continuación).

| **Excepciones:** | **Paso** | **Acción** |
|---|---|---|
| | 2 | Usuario no registrado. El usuario debe estar registrado en el sistema. |
| | 3’ | Connection refused: Error de conexión lo que indica que la conexión con el servicio controlador del módulo electrónico no esta inicializado. |
| | 3’ | Connection time out: Error de conexión, se debe a problema en el módulo electrónico, por ejemplo, no esta conectado a la red, esta apagado, mal configurado, además, de los propios problemas de red ethernet. |
| | 3’’ | Modbus Slave Exception: el módulo electrónico se encuentra bloqueado y tendrá que ser reiniciado manualmente. |
| | 5’ | Error Code = 1: Código de Función ilegal, El código función es desconocido por el módulo electrónico. |
| | 5’’ | Error Code = 2: Dirección de datos ilegal, esta dirección no está soportada en el módulo electrónico. |
| | 5’’’<br>5’’’’ | Error Code = 3: Valor de dato ilegal, depende de la petición y no corresponde a ella.<br>Error Code = 4: Falló servidor, se intento hacer la operación pero no pudo ser completada. |
| **Importancia:** | Vital | |

## 3.2 Implementación del Sitio Web

Para implementar el sacme que tiene las posibilidades de administrar, controlar y monitorear los recursos con funcionalidad encendido-apagado de energía eléctrica en aplicaciones comerciales, una interfaz grafica amigable al usuario es necesaria para interactuar con el hardware que en el capitulo anterior se desarrolló, aplicando las consideraciones del diseño del sitio web para ello, se detalla en esta sección el desarrollo de la aplicación web.

### 3.2.1 Independencia de la plataforma:

El lenguaje Java provee una máquina virtual o "procesador virtual" que ejecuta cualquier código que haya sido escrito en dicho lenguaje. Esto permite que el mismo binario ejecutable se pueda usar en todos los sistemas compatibles con el software Java (windows, linux, mac, solaris, etc.), por ello toda la aplicación esta implementada con tecnología java.

**3.2.2 Generación dinámica de páginas Web:** utilizando código java mediante script hace a las JSPs[10] como parte de la tecnología java, la alternativa para satisfacer el uso de generación de contenido dinámico; La integración de clases de java (.class), hace posible la separación de la lógica del negocio y la presentación de la información.

**3.2.3 Separación de la lógica en capas:** las ventajas que la separación de la lógica en capas ofrece están: aplicaciones mas robustas debido al encapsulamiento, mantenimiento y soporte mas sencillo, mayor flexibilidad, alta escalabilidad, entre otras. La aplicación web esta implementada en capas: Presentación, Negocio, Dominio, Datos y motor de persistencia.

**3.2.4 Uso de componentes:** proporcionan la infraestructura necesaria y la fontanería relacionada que permite a las aplicaciones web operar en un entorno complejo, multiplataforma y con capacidades de computación distribuida, tanto interna como externamente según se requiera en cada caso. Por ejemplo los componentes Java Beans[11] de Sun Microsystems.

**3.2.5 Facilidad de administración y uso:** haciendo uso de la representación del lenguaje visual una interacción amigable entre usuario y la aplicación.

**3.2.6 Independencia de Base de Datos:** El motor de persistencia traduce entre los dos formatos de datos: de registros a objetos y de objetos a registros, haciendo uso de este traductor el gestor de base de datos se hace independiente de la aplicación. La aplicación esta implementada con el gestor de base datos MySQL, pero con la integración del motor de persistencia, la aplicación se acopla a cualquier gestor de bases de datos.

**3.2.7 Código Abierto (open source):** importante respaldo de la sólida tecnología Java™, donde la evolución de la aplicación se hace de manera rápida. Cualquier usuario programador puede tomar el código revisarlo y hacerle mejoras, incorporando nuevas funcionalidades o modificar las ya implementadas y con ello actualizar la versión del mismo.

[10] **JavaServer Pages** (**JSP**) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

[11] Los **JavaBeans** son un modelo de componentes creado por Sun Microsystems para la construcción de aplicaciones en Java.

## 3.3 Esquema del Sitio Web.

Para comprender como esta implementado y ver cuales entidades están involucradas, se presenta a continuación el esquema del sitio web describiendo cada uno de los componentes.

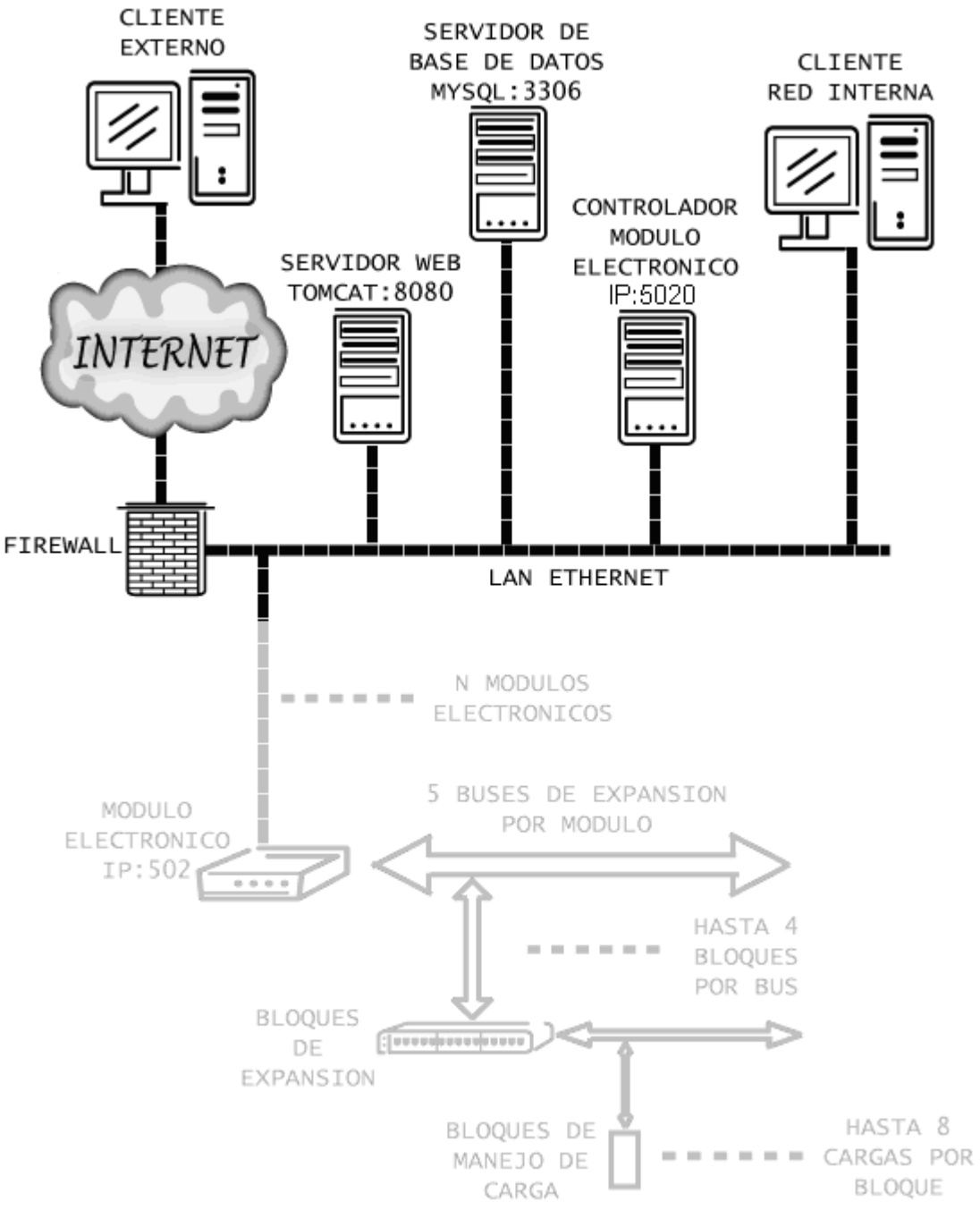


**Figura 3. 2** Esquema del sitio web.

### 3.3.1 El Cliente

Aunque el cliente no forma parte de la implementación como tal en este trabajo de graduación, es necesario tomarlo en cuenta ya que es el agente a quien está destinado el uso del sistema. Se entenderá como cliente al navegador web que direccione la aplicación con la url correspondiente a la aplicación del SACME, así en un mismo equipo pueden haber varios clientes simultáneos, claro que para la aplicación es transparente que se trate de cliente en el mismo equipo, entre los navegadores que pueden servir de clientes se pueden mencionar: Mozilla, FireFox, Internet Explorer, etc.

Dos tipos de clientes son los que el sistema puede darle soporte, uno es el *cliente interno* que es el que se sitúa dentro de la misma red Lan ethernet donde se encuentra el servidor que contiene la aplicación, y el otro es el *cliente externo*, se encuentra situado fuera de la red Lan, que puede ser desde otra red Lan o Internet, siempre y cuando el cortafuegos local esté configurado para permitir clientes externos.

### 3.3.2 Servidor de Servlets (Tomcat).

El servidor de sitios en la Web es un programa que corre como un servicio en un equipo o dispositivo electrónico. Este escucha las peticiones de acuerdo al siguiente formato de dirección de recursos url: http://nombre_del_servidor**:**numero_puerto/nombre_aplicación. El servidor Web buscará una página dentro de un grupo de estas, que son de tipo estáticas o dinámicas; de cualquier modo, siempre devolverá algún tipo de resultado html al cliente o navegador que realizó la solicitud. Este es fundamental en el desarrollo de las aplicaciones del lado del servidor que se implementa, ya que se ejecutarán en él.

Tomcat (también llamado *Jakarta Tomcat* o *Apache Tomcat*), es un servidor web con soporte de servlets[12] y JSPs. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web apache.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la maquina virtual Java; esta es una de las razones por la que se selecciono para la implementación del Sitio Web de este Trabajo de Graduación.

Debidamente instalada la aplicación web sacme en Tomcat, se puede acceder al sistema sacme a través de una interfaz grafica de usuario desde la Web, así un usuario podrá hacer uso del sistema cargando una pagina web a través del navegador con la url: *http://IpHost:8080/sacme*, en la tabla siguiente se describe el significado de cada parámetro

[12] La palabra *servlet* deriva de otra anterior, *applet*, que se refería a pequeños programas escritos en Java que se ejecutan en el contexto de un navegador web. Por contraposición, un *servlet* es un programa que se ejecuta en un servidor.

**Tabla 3. 12** Detalle de la url http://IpHost:8080/sacme

| Parámetro | Descripción |
|---|---|
| **IpHost** | Es la dirección IP de la PC donde esta instalado tomcat. |
| **8080** | Es el puerto por defecto en el cual Tomcat escucha peticiones http. |
| **sacme** | Es el nombre de la aplicación que navega el sitio web del sistema Sacme. |

### 3.3.3 Servicio de Base de Datos (MySQL).

Es el repositorio de información del sistema, este servicio es usado por la aplicación web para leer información, y guardar datos que se pueden persistir de manera directa permitiendo la administración y mantenimiento del sistema. Y también es usado por el servicio controlador administrador de eventos.

**MySQL** es un sistema de gestión de base de datos relacional, multihilo y multiusuario. Funciona sobre multiple plataforma y es de código abierto sencillo de usar y rápido. El valor agregado que mysql le da al sacme es persistir tablas de horarios de los eventos programados.

### 3.3.4 Servicio Controlador del Módulo Electrónico meSacme.

El Servicio Controlador del Módulo Electrónico; Es un software de aplicación de consola creado 100% en java que soporta multihilo, multiplataforma, por medio del cual es posible la comunicación de la aplicación usuario al módulo electrónico.

De aquí en adelante llamaremos meSacme al Servicio Controlador del Módulo Electrónico.

## 3.4 Comunicación con el meSacme

Este tiene la particularidad de ser servidor al lado de la aplicación web y cliente Modbus/tcp al lado del módulo electrónico.

Para entender con claridad como interactúa meSACME tanto con la aplicación web como el mismo módulo electrónico, se presenta a continuación el Diagrama de Secuencia.

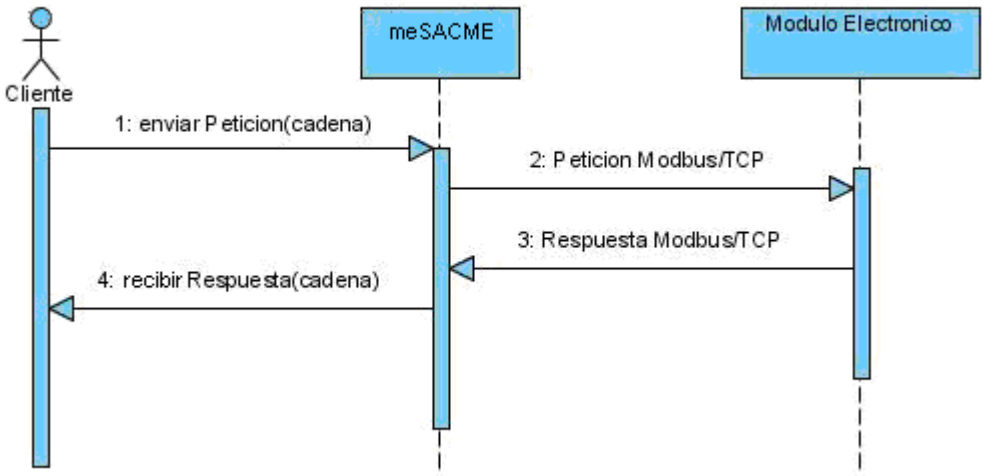


**Figura 3. 3** Diagrama de Secuencia de comunicación con meSACME.

Para establecer comunicación con meSACME, por el lado de la aplicación usuario, se abre una conexión socket a través del puerto 5020 (5020 por parecerse a 502 que es el puerto reservado de Modbus/TCP) para escuchar las peticiones de los clientes.

Cada solicitud del cliente, viene encapsulada en un formato de cadena de caracteres, ésta es transformada en petición Modbus/TCP y enviada a través de un socket por el puerto especificado en la solicitud y con la dirección IP del Módulo electrónico también proporcionada por la solicitud.

La solicitud tiene el formato siguiente:

Dirección IP # puerto # función # dirección de inicio # dato de función

**Tabla 3. 13** Detalle del formato de la solicitud del cliente.

| **Parámetro** | **Descripción** |
|---|---|
| Dirección IP | Es la dirección IP del módulo electrónico. |
| Puerto | Es el puerto de comunicación Modbus/TCP, su valor es 502. |
| función | Es la función a ejecutar y puede ser 02 ó 05. |
| Dirección de inicio | Es la dirección a la cual debe aplicarse la función, debe estar soportada por el hardware de lo contrario se esperará una excepción. |
| Dato de función | Este campo es propio de la función. |
| El caracter "#" | Sirve como separador de parámetros en la cadena solicitud. |

meSACME esta compuesto por un conjunto de tres clases: Sacme_ME.class, NuevoCliente.class y ModbusFunc.class.

Sacme_ME.class, es la clase principal donde se encuentra el método main() que es la que lanza el servicio quedando en estado de espera por un cliente (solicitud); cuando una nueva solicitud llega, llama a la segunda clase que a su ves inicia un nuevo hilo que le da seguimiento a la solicitud. Cuando Sacme_ME ha levantado el hilo de una solicitud vuelve a esperar por otra nueva. Esto hace que se pueda atender las peticiones simultáneamente (multicliente).

NuevoCliente.class, se entiende de la clase Thread de java y tiene sobrescrito el método run() donde la solicitud es capturada, separa los parámetros de la cadena con lo que se obtiene la información necesaria para construir la trama Modbus/TCP. Haciendo uso de los métodos implementados en la tercera clase se despacha la petición en el protocolo que el módulo electrónico entiende.

ModbusFunc.class, es la que hace de cliente Modbus/TCP, está compuesta por dos métodos donde se encuentran desarrolladas las funciones de Modbus/TCP (Read Input Discrete [02] y Write Single Coil [5]) haciendo uso de la librería del proyecto JAMOD[13] y su correspondiente API. Esta clase recibe los parámetros necesarios para construir una petición Modbus/TCP y tiene implementado un cliente Modbus sobre tcp que se encarga de hacer la comunicación a través de socket con el módulo electrónico que tiene un servidor Modbus/TCP embebido.

meSACME puede estar corriendo en el mismo servidor donde se encuentra instalada la aplicación o en un host aparte ya que toda la comunicación es realizada a través de socket.

El diagrama de clases de meSACME es mostrado a continuación.

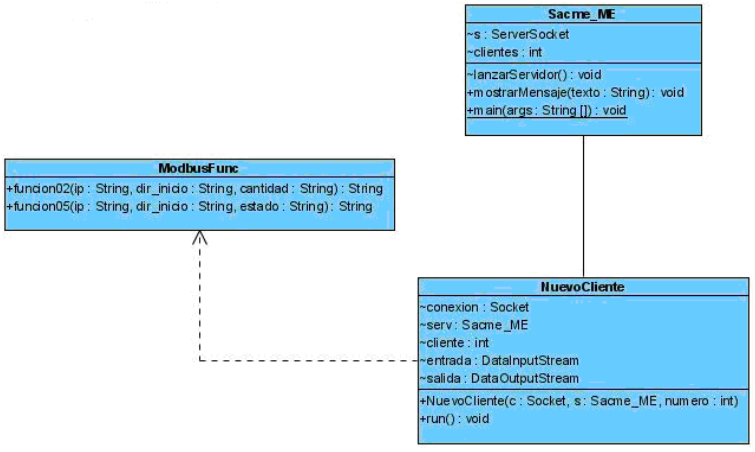


**Figura 3. 4** Diagrama de Clase para meSACME

[13] JAMOD es una implementación orientada a objeto del protocolo Modbus hecho 100% java donde se pueden realizar fácilmente aplicaciones maestro o esclavos en vario medios de transporte (IP y serial)

## 3.5 Separación Lógica en capas.

Distintas responsabilidades no deben ser delegadas en la misma clase, y llevado esto algo más allá, en el mismo conjunto de clases. En la actualidad, la tendencia más aceptada es la aplicación de patrones de diseño de arquitectura que dividen la responsabilidad en distintas capas que interaccionan unas con otras a través de sus interfaces. Se trata de los sistemas denominados multicapa o *n*-capas.

### 3.5.1 Arquitectura N-Capas.

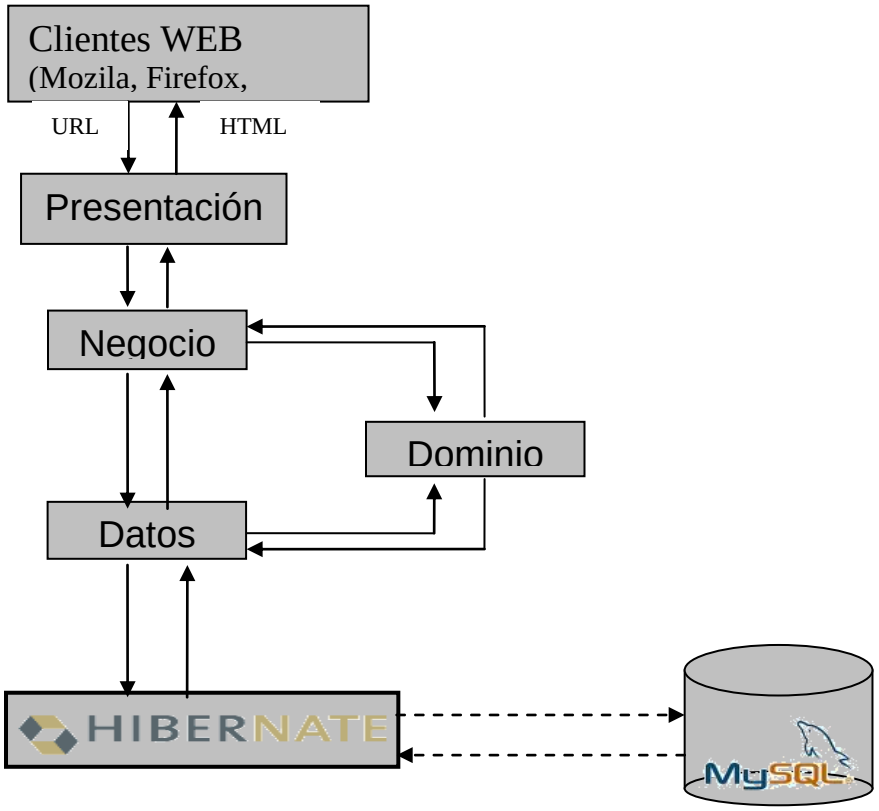


**Figura 3. 5 Arquitectura N-Capas**

#### 3.5.1.1 Clientes Web

Esta capa es importante incluirla dentro de la arquitectura N-capas, porque es el cliente el que hace uso del sistema. Un cliente esta vinculado al cada ventana del navegador en un equipo remoto o local que a través de la url, haga enlace con la aplicación sacme; el sistema podrá responderle al cliente en lenguaje html. Los navegadores disponibles son: Mozila, Firefox, Internet Explorer, etc.

#### 3.5.1.2 Presentación

Como su nombre indica, se limita a la navegabilidad y a gestionar todos aquellos aspectos relacionados con la lógica de presentación de la aplicación, como comprobación de datos de entrada, formatos de salida, internacionalización de la aplicación, autenticación, etc.

Esta capa esta compuesta por páginas estáticas html y dinámicas JSPs, apoyándose Javascript, para validación de datos de entrada y de salida, además, los estilos de las páginas están a cargo de los archivos css.

#### 3.5.1.3 Negocio

El resultado del análisis funcional de la aplicación viene a ser la identificación del conjunto de reglas de negocio que abstraen el problema real a tratar. Estás son las que realmente suponen el motor del sistema, dado que se basan en el funcionamiento del modelo real.

Esta compuesta por clases Java (.class) donde son implementadas las políticas de uso del sistema, el mantenimiento o actualización de las clases se da sin mayor problema debido a que esta capa es independiente de las otras.

#### 3.5.1.4 Dominio

Todos los objetos persistentes son definidos en esta capa, son livianos y no añaden ningún tipo de carga de proceso adicional (gestión de transacciones, gestión de seguridad, control de sesiones, etc.), están compuestos por sus propiedades y métodos que permiten acceder a las ellas. Todos los objetos que la capa de datos persiste en el repositorio, son los que el recipiente de dominio contiene, por ejemplo; Al recuperar datos de la base de datos no se puede hacer con registros ya que la aplicación entiende objetos por lo tanto se recupera con los definidos en esta capa.

#### 3.5.1.5 Acceso a datos

Esta capa es la encargada de persistir las entidades que se manejan en negocio, que están definidas en el dominio; el acceso a los datos almacenados, la actualización, eliminación, etc, también ofrece servicios relacionados con la persistencia o recuperación de información más complejos búsqueda avanzada.

### 3.5.2 El Motor de Persistencia Hibernate.

Hibernate es una herramienta ORM[14] (Object-Relational mapping) completa que ha conseguido en un tiempo record una excelente reputación en la comunidad de desarrollo posicionándose claramente como el producto OpenSource líder en este campo gracias a sus

[14] Mapeo Objeto Relacional es una técnica de programación para convertir datos entre el sistema de tipos. utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos.

prestaciones, buena documentación y estabilidad. Es valorado por muchos incluso como solución superior a productos comerciales dentro de su enfoque, siendo una muestra clara de su reputación y soporte la reciente integración dentro del grupo JBoss[15] que seguramente generará iniciativas muy interesantes para el uso de Hibernate dentro de este servidor de aplicaciones.

Para persistir objetos en hibernate debemos crear los archivos de mapeo, configurar hibernate, crear una fabrica de sesiones, con ella crear una sesión, acceder a los datos con la sesión controlando el manejo de datos con transacciones. Para analizar cada paso lo haremos comenzando creando el archivo de mapeo.

#### 3.5.2.1 Archivos de Correspondencia

Los archivos de correspondencia son archivos XML, con este se describe DTD de hibernate que una clase es una determinada tabla y que propiedades del objeto son persistentes y a que campo de la tabla se corresponden. Además se define la relación entre los objetos, dependencias, si un objeto va a ser contenedor de otro y como se refleja en la estructuras de tablas.

Por ejemplo, el archivo de mapeo para los módulos electrónicos que son adoptados por sacme es presentado a continuación.

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
        <class name="tesis.dominio.Modulo" table="modulos" >
                <id name="id" type="long" column="id_modulo" >
                <generator class="increment"/></id>
                <property name="nombre">
                  <column name="nombre"/>
                </property>
                <property name="descripcion">
                  <column name="descripcion" />
                </property>
                <property name="ip">
                  <column name="ip"/>
                </property>
                <set name="pines" lazy="false" access="field"
            cascade ="all,delete-orphan">
            <key column="id_modulo_FK"/>
            <one-to-many class="tesis.dominio.Pines"/>
          </set>
        </class>
</hibernate-mapping>
```

**Figura 3. 6** Archivo de correspondencia modulo.hbm.xml

[15] **JBoss** es un servidor de aplicaciones J2EE de código abierto implementado en Java puro.

Para cada clase que representa una tabla en la base de datos, un archivo de mapeo es necesario y toma una similitud como el mostrado en la figura 4.6, estos archivos son: usuario.hbm.xml, pines.hbm.xml, evento.hbm.xml, roles.hbm.xml y modulo.hbm.xml.

#### 3.5.2.2 Configuración de Hibernate

Hibernate esta diseñado para operar en muchos ambientes diferentes, por lo tanto hay un número grande de parámetros de la configuración. Afortunadamente, la mayoría de ellos tienen valores por defecto sensatos.

Hibernate puede configurarse por líneas de código con lo cual si queremos hacer una modificación debemos compilar nuevamente. Esto no es muy beneficioso por lo que hibernate nos permite configurarlo con un archivo de configuración el cual puede ser XML o texto llano que su extensión es .properties

#### 3.5.2.3 Crear una fabrica de Sesiones

Luego de crear el objeto Configuración y setear todas las clases persistentes se debe crear una fábrica de sesiones con la cual se crean las sesiones para persistir los objetos de esta forma:

```
SessionFactory fabrica =cfg. buildSessionFactory();
```

Hibernate permite tener más de una fábrica de sesiones esto sirve para cuando la aplicación se conecta con más de una base de datos. Todo esto esta implementado en las clases java dentro de la capa de Datos.

#### 3.5.2.4 Conectarnos a la base de datos

Nos conectamos a una base de datos por medio de una sesión en el momento que creamos la sesión nos conectamos a la base de datos, creamos la sesión con la fábrica de sesiones ya que la sesión hereda las configuraciones que seteamos en el objeto de configuración con el cual creamos la fabrica de sesiones.

Para Crear una sesión se hace de la forma:

```
Session session = fabrica.openSession(); // abrir una sesión.
```

Cuando creamos esta sesión se disparara una excepción (error) dado que no hemos especificado todavía la base de datos donde vamos a trabajar. Para crear satisfactoriamente una sesión debemos especificar:

**Tabla 3. 14** Propiedades que se deben especificar a hibernate.

| Nombre de la propiedad | Propósito |
|---|---|
| hibernate.connection.driver_class | Driver jdbc |
| hibernate.connection.url | Url de la base de datos |
| hibernate.connection.username | Usuario de la base de datos |
| hibernate.connection.password | Password del usuario de la base de datos |
| hibernate.connection.pool_size | Numero máximo de conexiones concentradas |

Estas configuraciones se pueden setear por medio de un archivo hibernate.cfg.xml. Este archivo puede usarse como un reemplazo para el hibernate.properties o, si los dos están presentes, el archivo XML sobrescribe las propiedades. Por ejemplo. Un extracto de este archivo para la aplicación del sacme se muestra en la figura siguiente.

```
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.url">jdbc:mysql://localhost/sacme</property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password">root</property>
<property name="hibernate.connection.pool_size">100</property>
<property name="show_sql">false</property>
<property name="dialect">org.hibernate.dialect.MySQLInnoDBDialect</property>
<property name="hibernate.hbm2ddl.auto">Update</property>
```

**Figura 3. 7** Extracto del archivo de configuración de hibernate hibernate.cfg.xml.

Como estamos trabajando con Mysql el driver jdbc es "com.mysql.jdbc.Driver", este es proporcionado gratuitamente y puede bajarse de internet o también se encuentra dentro del cd que se anexa a este documento.

El dialecto es especificado como "org.hibernate.dialect.MySQL o por el compatible con la configuración de la base de datos que estemos utilizando.

Los demás parámetros son fácilmente comprensibles y los cambios necesarios para adaptarse a sistemas ya implementados no requieren de mayor explicación.

### 3.5.2.5 Los Dialectos de SQL

SQL trata de ser un estándar pero no lo logra, según la base de datos que estemos usando se llamaran de un modo u otro las funciones y la sintaxis no es igual en todas; por lo que se denomina escribir en un dialecto cuando escribimos SQL que solo entiende un motor de

base de dato determinado. Como hibernate crea SQL en tiempo de ejecución debe saber el Dialecto que debe usar, Hibernate soporta la siguiente lista de Dialectos:

**Tabla 3. 15** Dialectos de hibernate.

| RDBMS | Dialecto |
|---|---|
| DB2 | org.hibernate.dialect.DB2Dialect |
| DB2 AS/400 | org.hibernate.dialect.DB2400Dialect |
| DB2 OS390 | org.hibernate.dialect.DB2390Dialect |
| PostgreSQL | org.hibernate.dialect.PostgreSQLDialect |
| MySQL | org.hibernate.dialect.MySQLDialect |
| MySQL with InnoDB | org.hibernate.dialect.MySQLInnoDBDialect |
| MySQL with MyISAM | org.hibernate.dialect.MySQLMyISAMDialect |
| Oracle (any version) | org.hibernate.dialect.OracleDialect |
| Oracle 9i/10g | org.hibernate.dialect.Oracle9Dialect |
| Sybase | org.hibernate.dialect.SybaseDialect |
| Sybase Anywhere | org.hibernate.dialect.SybaseAnywhereDialect |
| Microsoft SQL Server | org.hibernate.dialect.SQLServerDialect |
| SAP DB | org.hibernate.dialect.SAPDBDialect |
| Informix | org.hibernate.dialect.InformixDialect |
| HypersonicSQL | org.hibernate.dialect.HSQLDialect |
| Ingres | org.hibernate.dialect.IngresDialect |
| Progress | org.hibernate.dialect.ProgressDialect |
| Mckoi SQL | org.hibernate.dialect.MckoiDialect |
| Interbase | org.hibernate.dialect.InterbaseDialect |
| Pointbase | org.hibernate.dialect.PointbaseDialect |
| FrontBase | org.hibernate.dialect.FrontbaseDialect |
| Firebird | org.hibernate.dialect.FirebirdDialect |

### 3.5.2.6 Abrir una Sesión

Luego de haber completado toda la configuración podemos crear la sesión. La creamos de esta forma:

```
Session session = fabrica.openSession(); // abrir una sesión.
```

Abrimos una sesión dentro de ella existe un objeto encargado de controlar las transacciones llamado Transaction se debe referenciarlo para tener control sobre este objeto. Se hace de la siguiente manera:

```
Transaction tx= session.beginTransaction();
```

Al iniciar la transacción ya se está en condiciones de guardar nuestro objeto con el método de la sesión save(objeto) y se puede recuperar un objeto con el método get(objeto.class,id) donde id es el id del objeto y objeto.class es la clase del objeto. Se puede borrar con el método delete (objeto) y eliminar el objeto de la base de datos. Modificar el objeto persistente que se encuentra en la base de datos con el método upDate(objeto).

Luego de realizar todas las acciones que se desea solo se debe invocar el método del objeto Transaction commit() y se aplican los cambios, si surgiera algún error o las modificaciones no debieran ser aplicadas y se llama al método rollback() de el objeto Transaction

### 3.5.2.7 El Lenguaje de Interrogación del Mundo Objectual: Hql

El HQL (*Hibernate Query Language*) es un lenguaje de interrogación. En el mundo relacional disponemos del SQL (*Structured Query Language*) que nos permite obtener información haciendo preguntas basadas en las tablas y sus columnas. El equivalente en el mundo objectual es el HQL, que nos permite hacer preguntas basadas en los objetos y sus propiedades.

Hibernate se encarga de enlazar los dos mundos el relacional con el objectual. Traduce las consultas que se hacen desde el mundo objectual en HQL al lenguaje de interrogación del mundo relacional, el SQL, y transforma los resultados obtenidos en el mundo relacional (filas y columnas) en aquello que tiene sentido en el mundo objectual: objetos.

### 3.5.2.8 Ejecución de consultas

Existen diversos métodos para ejecutar consultas.

**El método “Session.find()”**

Este devuelve el resultado de la consulta en una java.util.List. Es bastante práctico si se devuelven pocos resultados, ya que los tiene que mantener en memoria:

**El método "Session.iterate()"**
Este método devuelve un java.util.Iterator y es práctico si la consulta nos proporciona un gran número de resultados.
El iterador se encarga de cargar los objetos resultantes de la consulta uno a uno, a media que los vamos pidiendo

**La interfase "Query"**
La interfase Query nos permite ejecutar consultas pero aporta algunas ventajas:

- Podemos especificar el número máximo de registros que queremos que se nos devuelvan.
- Podemos especificar el primer registro que queremos obtener
- Permite el uso de parámetros con nombre

## 3.5.3 La Interfaz de Usuario

Cuando un usuario accede al sitio web del sistema sacme a través de un navegador de paginas web, la primer pagina que visualiza es la de inicio de identificación del sistema, a partir de aquí el usuario se comunica con el sistema a través de la navegabilidad del sitio web con lenguaje visual amigable, la pagina de inicio del sacme se muestra a continuación.


**Figura 3. 8** Página de inicio del SACME.

El usuario tiene la opción de identificarse o entrar como invitado pulsando el botón "Entrar como no Registrado", el invitado esta limitado en recursos y solo puede monitorear dichos recursos. Para identificarse es necesario que el usuario introduzca la dirección de correo

electrónico con que se registro en la base de datos, así se evita de tener coincidencia de usuarios ya que no existen duplicados de correos electrónicos, la contraseña debe ser la proporcionada por el administrador, esta se encuentra encriptada como texto plano en la base de datos.

Una ves identificado el usuario, se le presentan de manera gráfica las opciones con las que puede interactuar, estas opciones son los mismos casos de uso con los cual cuenta sacme, a continuación se presenta el diagrama de casos de uso del sitio web.

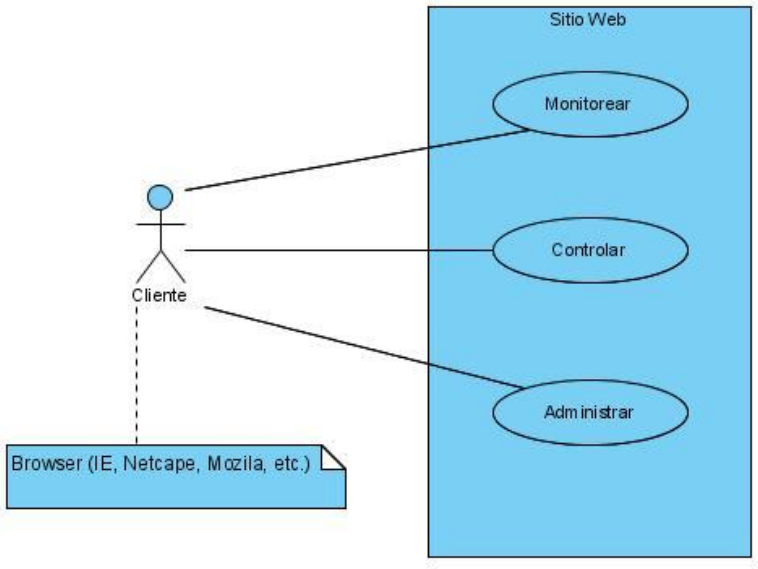


**Figura 3. 9** Diagrama de Caso de Uso del Sito Web.

### 3.5.3.1 Monitorear

Permite ver el estado de los recursos que nos interese saber en un determinado tiempo. Un usuario del sistema puede seleccionar el recurso de una lista a la cual tiene acceso, luego pulsa el botón “ver estado” y se le presenta el estado actual de dichos recursos.

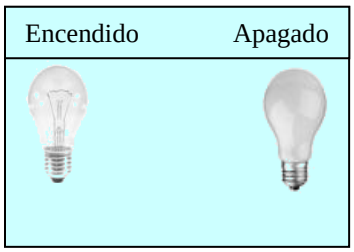


**Figura 3. 10** Monitorear.

### 3.5.3.2 Controlar

Permite encender o apagar uno o un grupo de recursos que deseamos manipular en cualquier instante. Esta caso va acompañado del anterior Monitorear, así si se requiere encender es porque esta apagado, en el caso que al momento de hacer la solicitud de cambiar el estado, este ya ha sido cambiado de manera local a través del interruptor mecánico, el sistema realiza la petición y es el módulo electrónico quien toma la decisión si hace o no el cambio.

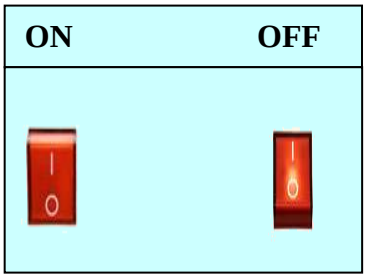


**Figura 3. 11** Controlar.

### 3.5.3.3 Administrar

Esta opción solo esta habilitada para el administrador y los usuarios registrados, no así para el invitado, porque requiere de manipulación de los recursos y esta restringido por los roles asignados por el administrador.

Los eventos programados y las operaciones de mantenimiento son realizados en la administración.

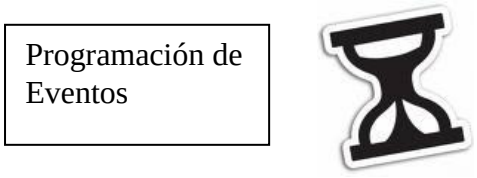


**Figura 3. 12** Administrar.

Para las tres opciones antes descritas existe una comunicación interna que es transparente para el usuario, ese funcionamiento hace que cada solicitud del usuario se refleje en una acción física sobre los recursos. Cada acción comienza desde que un usuario a través de un navegador accede al sitio web sacme, posicionándose en la pagina de inicio en donde debe

identificarse, la aplicación valida los datos de entrada de información de usuario y muestra los casos de uso a los cual tiene acceso a su correspondiente rol, si por ejemplo se requiere monitorear un recurso se selecciona de la lista disponible, luego se presiona el botón "Ver Estado" que esta ubicado en la parte inferior de la lista de recursos. La dirección IP, el puerto Modbus/TCP y la función que se va a ejercer sobre la dirección especifica del recurso son encapsulada en formato de cadena de caracteres y enviada a través de un socket por el puerto 5020 que es donde meSACME escucha, este a su ves levanta un hilo para darle seguimiento a esa solicitud, se construye la petición Modbus/TCP con los datos que proporciona la cadena solicitud y se envía por otro socket por el puerto Modbus/TCP (502) para que el módulo electrónico haga la acción sobre el recurso especificado, la respuesta es presentada al usuario en la interfaz grafica, al ser esta satisfactoria se represente con una imagen que distingue entre encendido y apagado o en caso de ser no satisfactoria se presenta el error ocurrido, para saber de que error se trata la interfaz proporciona un enlace de información de errores el cual puede ser consultado en cualquier instante.

El diagrama de secuencia para monitorear es presentado a continuación.

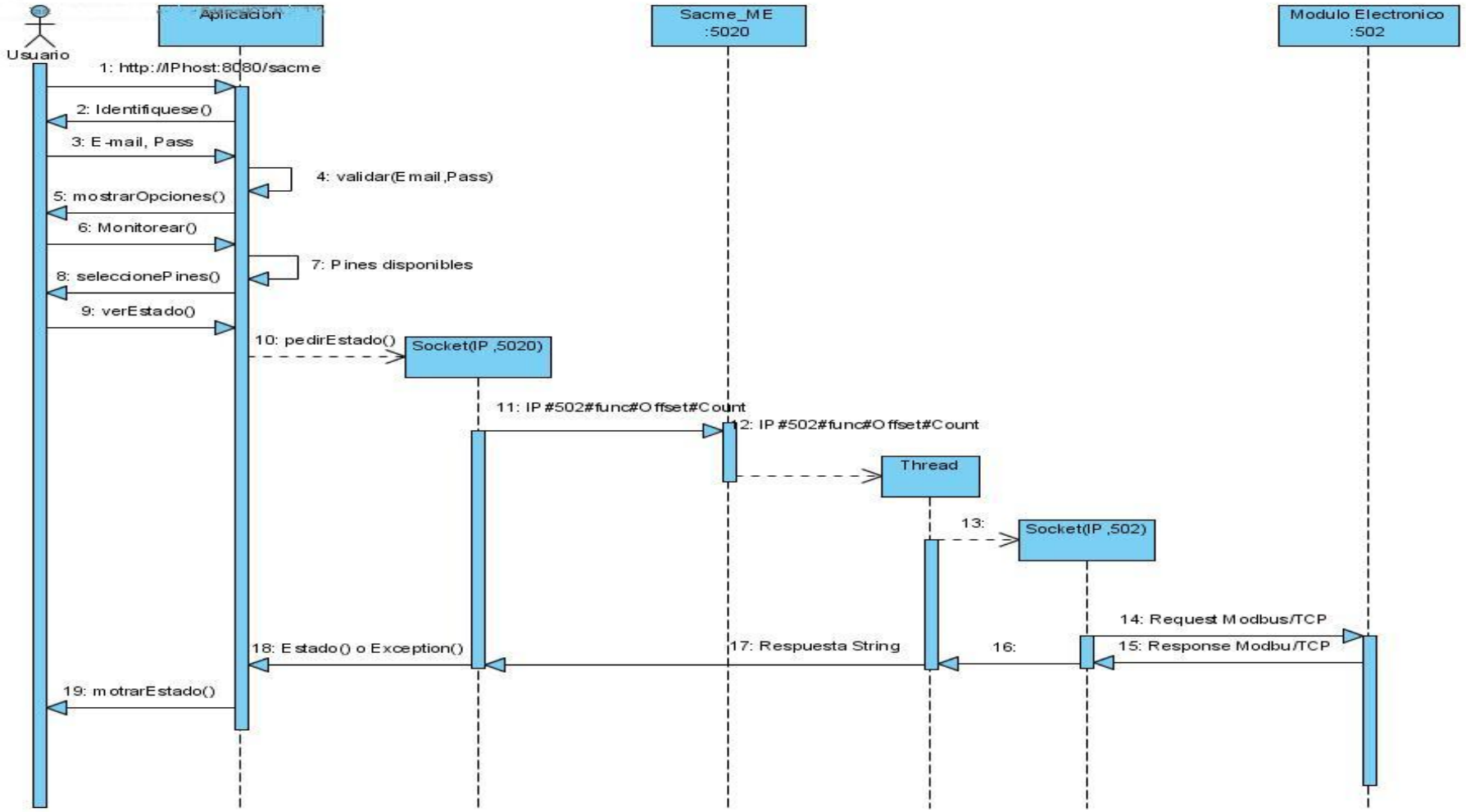


**Figura 3. 13** Diagrama de Secuencia Monitorear.

Para el caso de Controlar, tiene una similitud con el diagrama anterior por lo que no es relevante mostrar su diagrama de secuencia.

### 3.5.4 Administración de Eventos Programados

Para la administración de eventos programados es necesario que el servicio administrador de eventos, al cual en adelante llamaremos aeSACME[16], sea levantado de la misma manera que meSACME. Un evento puede estar relacionado a un solo recurso o a un grupo de ellos.

**aeSACME**. Es una aplicación de consola puramente desarrollada con java, que debe estar corriendo en el servidor donde esta la aplicación web. El intervalo de tiempo es programado y es en rangos de minutos, por ejemplo cada 5 minutos lee la base de datos.

aeSACME está periódicamente revisando la base de datos donde se encuentra la información de los eventos, la información contiene, titulo, descripción, fecha, hora, estado, repetición y acción de los eventos.

**Tabla 3. 16** Detalle de la Entidad donde se almacena la información del evento

| Nombre | Propósito |
|---|---|
| Titulo | Es un nombre único designado por el usuario del sacme, que representa a un evento, es recomendable que este nombre sea descriptivo con la función y localización del recurso o recursos. |
| Descripción | La información adicional necesaria para distinguir los eventos, este es opcional. |
| Fecha | Es la fecha que el evento se dio o dará por primera ves y tiene el formato dd/mm/aaaa. |
| Hora | Es la hora del evento y tiene el formato hh:mm:ss, aunque los segundos son ignorados en el calculo es obligatorio el formato. |
| Estado | Puede tener uno de dos opciones, activado o desactivado. Lo que significa es que un evento puede estar programado pero no se lleva a cabo si esta desactivado, pero si esta activado el evento se da. |
| Repetición | Tiene tres opciones, Solo una Ves, Diario o Semanal. La repetición se hace a partir de la fecha del evento, en el caso de Semanal, el evento se lleva a cabo cada coincidencia múltiple de siete del día que se da el evento por primera vez. |
| Acción | Encender o Apagar el recurso o los recursos. |

Primero se obtiene una lista de los eventos que tengan el Estado = Activado, ellos son los candidatos en primera instancia a ser ejecutados, luego esta lista pasa por un filtro donde es reducida a solo los que coinciden con la fecha actual del host, la lista sufre otra reducción cuando se verifica la coincidencia con la hora actual y finalmente los eventos candidatos

[16] Sacme_AE: Servicio Administrador de Eventos del Sacme.

son los que estrictamente coinciden con los el minuto actual mas el intervalo de tiempo que se programó a aeSACME para que revise la base de datos. Cada evento es ejecutado en la hora exacta con que fueron programados.

Todo lo que en el párrafo anterior se describe se aprecia también en el diagrama de estado de aeSACME que se muestra a continuación.

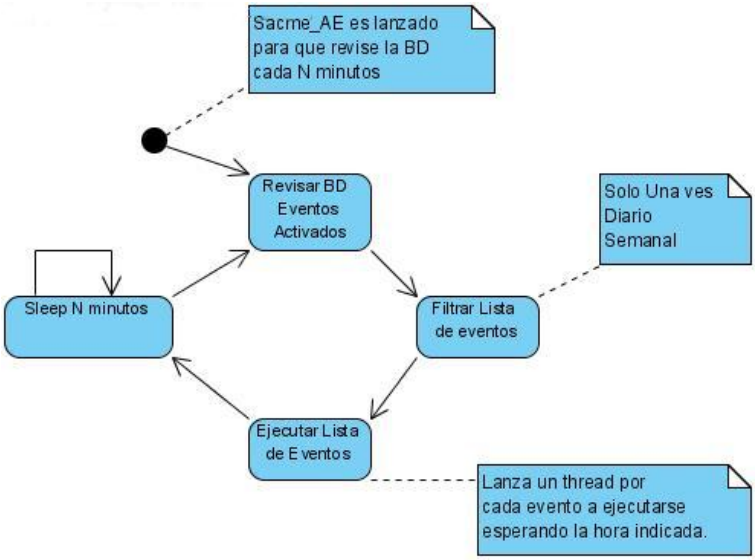


**Figura 3. 14 Diagra**ma de Estado para la administración de eventos.

## 3.5.5 Diagrama Entidad/Relación

La información que se guarda en la base de datos necesaria para el buen funcionamiento de sacme, esta distribuida en entidades que se encuentran relacionadas entre si. Las asociación de las entidades van de acuerdo a las necesidades con que fue diseñado sacme, un usuario por ejemplo solo puede tener un solo rol, pero un rol puede pertenecerle a varios usuarios, esta asociación se conoce como uno a muchos se implementa en las entidades usuario y roles respectivamente. Todas estas relaciones están definidas tanto en el motor de persistencia que ahí es vital y en la base de datos. El diagrama de entidad relación se muestra a continuación.

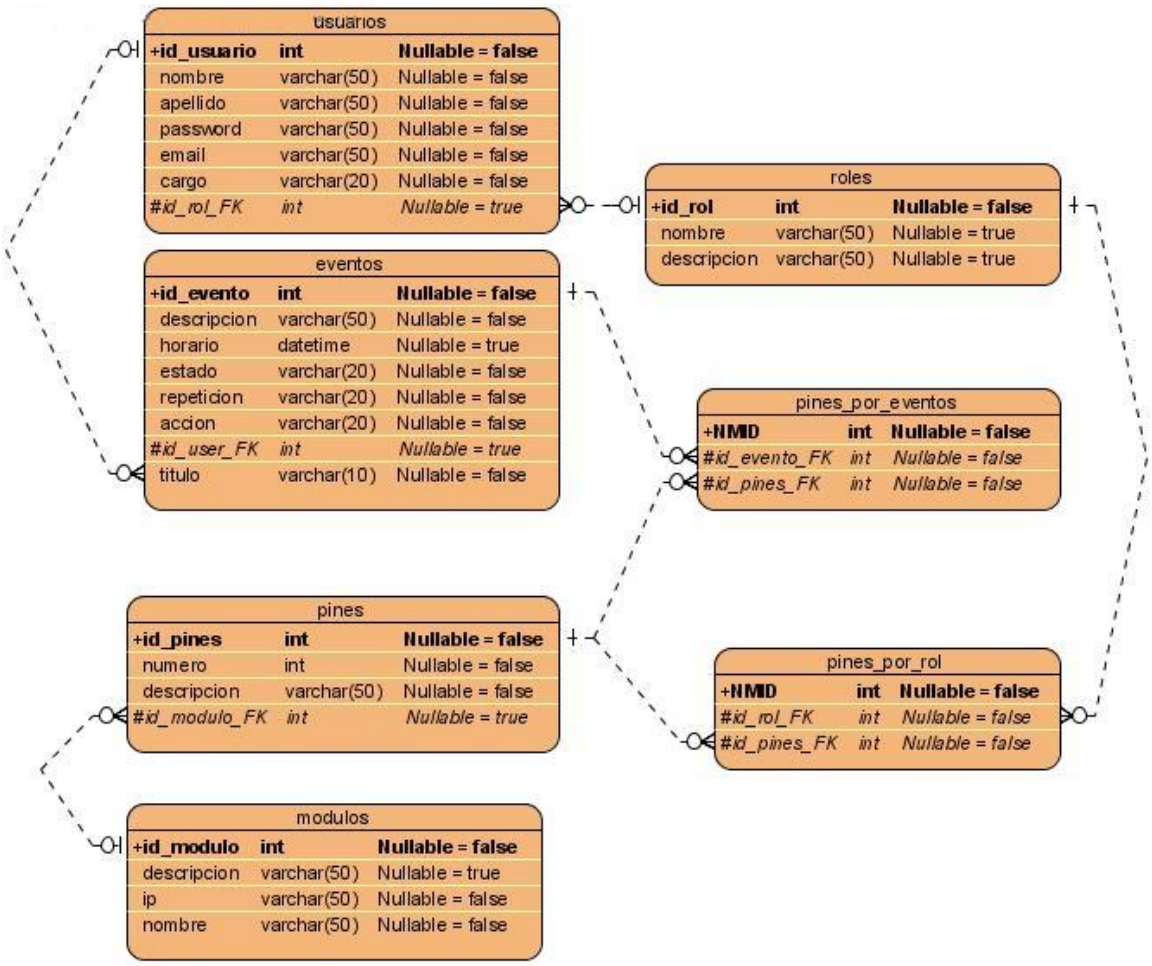


**Figura 3. 15** Diagrama Entidad Relación.

El archivo .sql donde contiene el script de las tablas que deben ser creadas en la base de datos se encuentra en el anexo y también está disponible en el cd adjunto a este documento.

# CAPITULO IV

# PROPUESTAS DE MEJORAS AL SISTEMA SACME

## Introducción

En los capítulos anteriores está desarrollado el sistema SACME, tanto su diseño como la implementación, a partir de ello, en este capitulo se presentan algunas propuestas de mejoras al sistema para que contribuyan a ser mas atractivo y competitivo con los sistemas similares que existen en el mercado, ya que se limita al monitoreo, control y la administración a través de la programación de eventos de forma remota, teniendo la desventaja de no ser inteligente, pero tiene toda la base para llegar a ser usado en domótica, claro que para ello se necesita la incorporación de sensores para que SACME se convierta en sistema inteligente. Las mejoras propuestas están separadas en hardware y software respectivamente y son independiente cada una, así, cada mejora se puede llevar a cabo sin tener que modificar las otras partes involucradas.

## 4.1 Mejoras Al Hardware

El hardware del SACME cumple con todos los requerimientos establecidos en la formulación de este trabajo de graduación, este es programable, controlado desde la Web desde cualquier cliente Modbus. No obstante, la posible comercialización requiere de ciertos cambios y mejoras al hardware para darle mayor robustez y darle mayor rapidez en los procesos.

### 4.1.1 El módulo electrónico

El módulo electrónico cuenta con 5 buses seriales en los que se conectan todos los bloques de expansión, estos buses permiten la conexión de hasta 4 módulos de expansión en cada uno.

Estos buses transmiten en señales lógicas digitales de 0 y 5V. Esto representa una limitante si se desea extender los bloques cientos de metros, ya que las señales digitales no tienen la suficiente potencia para ello.

Una forma de resolver esta limitante es la de implementar los buses con un protocolo estándar para largas distancias, como el RS485 que es precisamente para estos propósitos, y que daría la ventaja de estar trabajando sobre un bus estándar.

Otra forma de mejorar los buses es la implementación de lazos de corriente, estos lazos permiten extender las distancias hasta kilómetros ya que las caídas de voltaje no son importantes sino solo la corriente que se maneja en dichos buses.

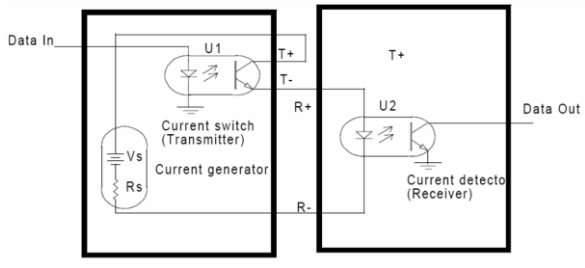


**Figura 4. 1** Diseño de los lazos de corriente.

La figura 4.1 muestra la forma en que se desarrollan los lazos de corriente, para ello se utilizan optoacopladores.

### 4.1.2 Boques de expansión

Los bloques de expansión se desarrollan utilizando multiplexación para lograr direccionar una cantidad importante de salidas y entradas. Se utiliza multiplexores de 16 salidas y de-multiplexores de 8 entradas, además se utiliza un decoder para direccionar los 4 bloques conectados al bus. Con esto se genera un total de 160 direcciones posibles de entrada y salida.

Una forma de mejorar estos bloques es la de sustituir los multiplexores por un microcontrolador y que este controle la ampliación de los pines y también la comunicación serial.

Si se desea incrementar el número de entradas y salidas simplemente se debe cambiar el decodificador por uno más potente. Con lo cual se permitirá tener mas bloques conectados a los buses.

### 4.1.3 Boques de manejo de cargas

Los bloques de manejo de cargas son los que conectan directamente a las cargas, estos tienen una limitante de capacidad de manejo de corriente que depende del tipo de carga que se desee manejar.

Otro aspecto importante de los bloques de manejo de carga es que están diseñados para manejar circuitos de iluminación, por lo que tanto el rele de control como la generación de estados de las cargas se manejan como interruptores de tres vías. Si se desea controlar otro tipo de carga habrá que modificar el bloque para adecuarse a la carga.

## 4.2 Mejoras al Software

El diseño y por lo tanto la implementación de este trabajo de graduación hace posible que se puedan hacer mejoras al software, ya que su implementación esta desarrollada con la arquitectura n-capas y también aprovechando la aportación del uso de código abierto, se espera que la evolución de nuevas versiones se realicen sin mayor dificultad.

### 4.2.1 La Interfaz Gráfica

La interfaz gráfica forma parte de la capa de presentación dentro de la arquitectura n-capas y tiene que ver con el lenguaje visual que el usuario percibe, están involucradas páginas estáticas y dinámicas; estas son las que se deben de modificar o rediseñar para que el usuario pueda interactuar con SACME con la mayor facilidad.

El uso de marcos (Frames) no son recomendables para un sitio Web, algunos cuelgan los navegadores. La pagina principal de la interfaz de usuario esta hecha con marcos, estas están ubicadas en de la carpeta "principal" dentro de la carpeta contenedora de la aplicación sacme; solo la página del marco llamado "main", es la que se va actualizando según sea el caso correspondiente a la navegabilidad del sitio. Una buena mejora seria trabajar con estilos de paginas (archivos .css) donde se puede definir los marcos de manera mas moderna. La pagina que define los marcos se llama "marco_general.jsp", ésta es la que debe ser cambiada por una pagina hecha con estilo css.

```
<html>
<head>  <title>Principal</title> </head>
<frameset rows="75,*" border="0">
    <frame name="heading" src="paginaCabecera"  scrolling="no" frameborder="no" noresize>
    <frameset cols="150,*" border="0">
        <frame name="menu" src="paginaMenu"  scrolling="no" frameborder="no" noresize>
        <frame name="main" src="paginaPrincipal"  scrolling="yes" frameborder="no" noresize>
    </frameset>
</frameset>
  <noframes>
        <body marginwidth="0" marginheight="0" leftmargin="0" topmargin="0">
        </body>
   </noframes>
</frameset>
</html>
```

**Figura 4. 2** Extracto de código de la página marco_general.jsp

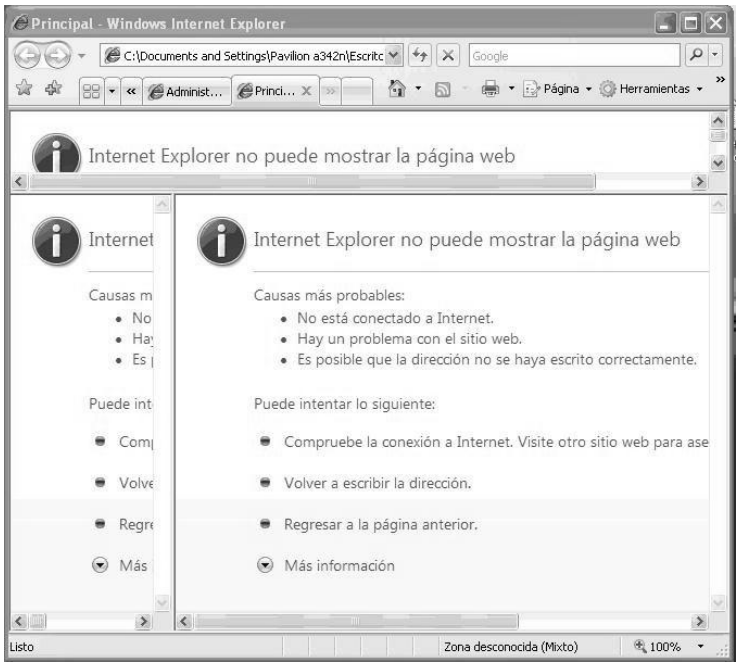


**Figura 4. 3** Vista de los marcos con "marco_general.jsp"

Cuando se trata de encender y apagar o monitorear recursos a los que el sistema tiene control, el usuario tiene que seleccionar de una lista proporcionada el la interfaz de usuario; es responsabilidad del usuario si selecciona el recurso correcto. Para evitar una equivocación por falta de percepción visual de parte del usuario, la interfaz grafica debe ser amigable y fácilmente manejable como un simple clic.

La manera como se selecciona con la actual interfaz de usuario se muestra en la figura 4.1. Las listas de recursos son mostradas en forma de tabla donde cada fila representa un recurso y cada columna su respectiva información. El primer objeto de cada lista es un selector checkbox que al seleccionarlo, se ejercerá sobre el la acción que el usuario desea.


**Figura 4. 4** Pantalla de selección de recursos para monitorear.

Por ejemplo, en la figura 4.1 muestra el estado de todos los recursos disponibles como desconocido, esto se debe a que no se ha ejecutado el monitoreo, para hacerlo es necesario que se seleccionen los recursos de los que se tiene interés saber su estado, chequeando la caja correspondiente a cada recurso, en la figura 4.1 se ha seleccionado los recursos: Luz Media, Sala de Lectura y Luz de Pasio, ahora solo faltaría hacer clic en el botón "Ver Estado".

Ahora esto podría tener mejor atractivo y mejor manejo si en lugar de cajas de selección, se le incorpora a la lista, imágenes que simulen un interruptor convencional de apagado y encendido, y dicha imagen dará la sensación que el recurso esta apagado o encendido según sea la situación.

La idea a la que se pretende llegar se muestra en la figura 4.2, teniendo en cuenta que los colores, la posición, entre otros aspectos de diseño gráfico serán determinados a la hora de implementar dicha mejora.

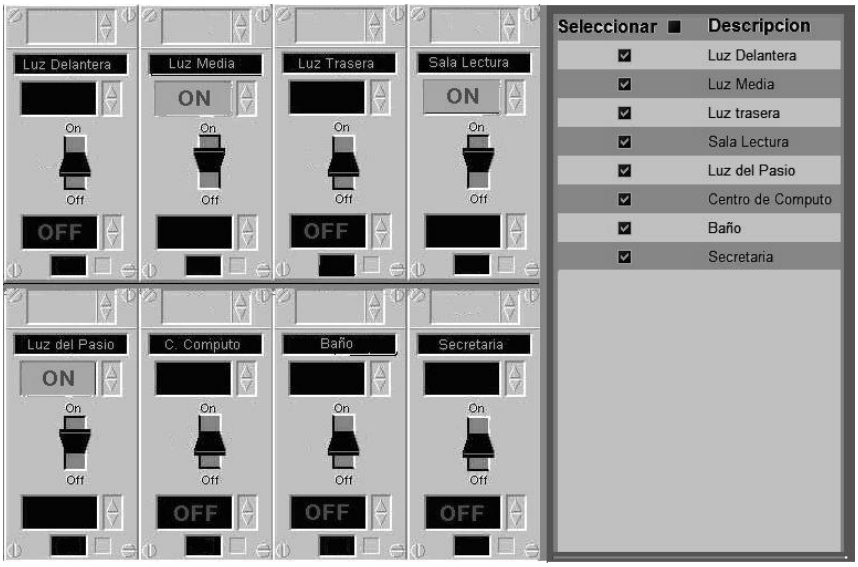


**Figura 4. 5** Vista previa de la pantalla de control y monitoreo.

La idea trata tal y como se muestra en la figura anterior, en la parte derecha se puede seleccionar los recursos a los que se quiere controlar o monitorear, y en la parte izquierda cada recurso debe representarse con un interruptor correspondiente a cada recurso seleccionado y se pueda apreciar visualmente el estado del recurso (apagado o encendido), así como también, se podrá hacer clic sobre el interruptor para que de alguna manera se realice la solicitud que se requiera.

Los archivos a modificarse son: listar_pinesRoot.jsp, listar_pinesXusuario.jsp y consultar_estado.jsp que se encuentran en la carpeta con nombre "jsp" que se encuentra en la carpeta contenedora de la aplicación sacme también con el nombre "sacme". El contenido o el rol que los archivos tienen en la capa de presentación se presentan en la tabla siguiente.

**Tabla 4. 1** Páginas encargadas de visualizar los recursos.

| Página | Propósito |
|---|---|
| listar_pinesRoot.jsp | Visualiza en forma de lista los recursos que el sistema tiene registrado en la base de datos, esta lista es completa y solo debe ser accedida por el administrador de sacme. Cuando se trata del usuario administrador, éste tiene todos los derechos de uso, por lo tanto los recursos son mostrados a través de esta página |
| listar_pinesXusuario.jsp | Al igual que el anterior, visualiza en forma de lista los recursos, solo si se trata de un usuario registrado, o sea distinto de administrador y de invitado, por lo tanto solo se muestran los recursos correspondientes al rol que el administrador le asigne previamente a dicho usuario. |
| consultar_estado.jsp | Monitorea el estado de una lista de recursos como primer paso y si se trata de un usuario diferente al invitado esta misma página puede llevar a cabo el control del recurso. Esta página debe ser llamada una ves se haya seleccionado los recursos en base a la lista mostrada con cualquiera de las dos paginas que se describen anteriormente en esta tabla. |

### 4.2.2 Paginación

SACME está diseñado para dar soporte a N módulos electrónicos, y cada módulo tiene la capacidad de controlar 160 recursos, esto hace que la lista de recursos sea cada ves mas grande cuanto mas módulos electrónicos estén adjuntos al sistema, de aquí que es necesario hacer uso de la paginación de registros.

¿Que es la paginación? Es la propiedad de mostrar los resultados de n registros en una página de una consulta hecha a la base de datos, sin importar el número de registros que devuelva dicha consulta. Al tener algún conocimiento mas avanzado sobre paginación, es posible parametrizar el número de registros que se desea mostrar por página y muchas otras cosas, como hacer ordenación por un campo que seleccione el usuario o hacer clic en un campo y que se redireccione a otra página incluso pasándole parámetros de la misma consulta.

El desarrollo consiste en crear primeramente un Bean llamado por ejemplo BeanPaginacion.java que tendrá todos los métodos y propiedades necesarias para ser utilizado desde una pagina jsp. Este bean debe ser estándar y debe funcionar con cualquier consulta que le pasemos y con cualquier número de campos y registros preferiblemente.

El bean debe tener las siguientes funciones generales que resumen en la tabla siguiente.

**Tabla 4. 2** Funciones que debe tener el bean de paginación.

| Función | Propósito |
|---|---|
| cadenaNoHayDatos | Esta función debe crear un String con la cadena que queramos que salga en pantalla cuando no hay resultados en la consulta. |
| cadenaSiHayRegistros | Esta función tiene como finalidad, devolvernos true en caso que existan registros en la consulta o false, en caso contrario. |
| ConectarBD | esta dedicada a la conexión a la base de datos apoyándose con hibernate. |
| CerrarConexion | Simplemente cierra la conexión |

**Propiedades**

Las propiedades son utilizadas para pasarle al Bean los parámetros necesarios antes de que muestre los registros en pantalla. Como por ejemplo el título del listado, el color de las filas impares, el de las pares, o bien el tipo de fuente o el tamaño que usará al imprimir los resultados en pantalla.

Las propiedades cargan en las variables globales para el Bean, los datos que le pasemos desde la página JSP. Las posibles propiedades se muestran a continuación debidamente comentadas, puede guiarse por ellas para saber que hace cada propiedad.

```
public void setTextoNoHayResultados(String v_mensaje){
strMensaje_no_hay_resultados = v_mensaje;
}
public void setAnchoTabla(String strAncho){
strAnchoTabla = strAncho;
}
public void setColorFondoTabla (String RGBColor){
ColorFondoTabla = RGBColor;
}
public void setTipoLetra (String strTipoLetra){
strFaceLetra = strTipoLetra;
}
public void setTamanoLetra (String tamanio){
strTamanoLetra = tamanio;
}
public void setColorFilaImpar (String RGBColor){
ColorFilaImpar = RGBColor;
}
public void setColorFilaPar (String RGBColor){
ColorFilaPar = RGBColor;
}
public void setColorTextoFilas (String RGBColor){
ColorTextoFilas = RGBColor;
}
public void setColorTextoTitulo (String RGBColor){
ColorTextoTitulo = RGBColor;
}
public void setRegistros_Pagina(int intRegistros){
registros = intRegistros;
}
```

```
public void setRegistroActual(int intRegistro){
RegistroActual = intRegistro;
}
public void setTitulos_Campos(String array_titulos_campos[]){
arTitulo_Campos = array_titulos_campos;
}
public void setPagina(String strPagina){
que_pagina = strPagina;
}
public void setMostrar_paginaXdeY(boolean SioNo){
bl_mostrar_paginaXdeY = SioNo;
}
public void setMostrar_Paginas(boolean SioNo){
bl_mostrar_paginas = SioNo;
}
public void setSelect(String strSelect1){
strSelect = strSelect1;
}
public void setOrderBy(String strOrderBy1){
strOrderBy = strOrderBy1;
}
```

**Figura 4. 6** Posibles propiedades para el bean de paginación.

**Método**

El más importante es el método que se encarga de imprimir los resultados y además de evaluar todos los parámetros que han sido pasados previamente. Por lo tanto es el que pondrá los datos en pantalla. Posiblemente se le puede llama Mostrar_Registros()

**La página jsp**

El código de la página JSP, no es especialmente difícil, si se conoce cómo funciona el Bean, ya que únicamente es rellenar las variables, establecerlas y llamar al método Mostrar_Registros().

Las partes de la página jsp que debe tener se resume en la tabla a continuación.

**Tabla 4. 3** Partes de la página jsp para la paginación.

| Sección | Propósito |
|---|---|
| Cabecera de página. | Aquí se le especifica a la página jsp para que utilice el bean. <jsp:useBean> |
| JavaScript | Una parte fundamental de la página JSP para que el efecto de resaltado de cada línea tenga efecto. debe ir al principio de cada página, entre los tags, <head> y </head> |
| Posición y dirección del cursor | Debemos recoger los parámetros que el propio Bean nos envía. Estos parámetros son el Registro por dónde se quiere empezar a mostrar registros y otro es la dirección a dónde queremos dirigirnos, es decir hacia delante o hacia atrás. |

**Tabla 4. 4** Partes de la página jsp para la paginación (Continuación).

| Sección | Propósito |
|---|---|
| Tipo de consulta | Select, from, where, orderby |
| Nombres y Títulos de los Campos | Para que se haga el tipo de consulta de solo los campos que se requiere que aparezcan en pantalla es necesario especificar los nombres de los mismos. |
| Campo clave | Es muy importante, tener un campo clave único para que el Bean pueda saber cuantos registros tiene la consulta y así saber cuantas páginas tiene que mostrar en los resultados. Para ello se debe establecer cual es campo clave o Primary Key de tu consulta. |
| Otros Datos importantes | Otros datos importantes serán el número de registros que deseas poner en cada pantalla, el ancho de la tabla con los resultados y el nombre de la página JSP. |
| Mostrar Registro y cierre | Por último debemos llamar al método que hace que todo funcione *Mostrar_Registros().* |

Toda esta sección de paginación solo tiene como objeto dar una orientación de lo que se necesita para poder tener una mejor visualización de los registros en pantalla para un gran número de recursos, no se pretende que lo anterior sea suficiente par desarrollar la paginación que se requiere, pero si mostrar donde se necesita aplicarlo, y es precisamente cuando la interfaz de usuario imprime en pantalla la lista de recursos disponibles al usuario y esta al ser grande tendríamos problema para visualizar cada recurso. La paginación ayudará cuando los recursos vayan aumentando de números mediante mas módulos electrónicos estén soportados por sacme.

### 4.2.3 Seguridad

La seguridad en un sitio web es muy importante ya que de ello depende la integridad de los datos y la confiabilidad del usuario, SACME esta implementado con lo mas mínimo en lo que a seguridad se refiere.

**Encriptación de la contraseña**

La contraseña en la base de datos esta encriptada con el algoritmo SHA-1[17] que produce una salida resumida de 160 bits de un mensaje que puede tener un tamaño máximo de $2^{64}$ bits, es muy útil ya que por mas grande que sea la cadena a encriptar, el resultado de la encriptación no supera nunca los 40 caracteres, pero la cadena encriptada puede que no sea caracteres imprimibles, para ello se utiliza el modo de expresión de secuencias de bytes denominado "Base 64", que consiste en la asociación de cada 6 bits de la secuencia a un elemento de un "alfabeto" que consta de 64 caracteres imprimibles. Toda esta encriptación esta responsabilizada por la clase SHA1BASE64.class que esta en la capa de negocio

[17] SHA-1 *Secure Hash Algorithm*, Algoritmo de Hash Seguro, es un sistema de funciones hash criptográficas relacionadas de la *Agencia de Seguridad Nacional de los Estados Unidos* y publicadas por el *National Institute of Standards and Technology* (NIST).

donde solo se le pasa la contraseña como parámetro y devuelve una cadena encriptada o sea caracteres imprimibles,; esta cadena es está almacenada en la columna "password" de la tabla "usuario" en la base de datos.

**Deficiencia de seguridad en SACME**
Cuando un usuario se identifica en el sistema los datos son enviados en forma plana haciéndolos vulnerable a cualquier hacker, aquí es donde se debe de emplear alguna forma de encriptación u otro tipo de seguridad.

Para que el sistema SACME funcione adecuadamente, por defecto en la base de datos debe tener ingresado dos registros obligatorios, estos son el usuario administrador y el usuario invitado, en las paginas de navegabilidad del sitio el usuario siempre es reconocido mientras permanece en la misma ventana, gracias a la variable de sesión de la clase Session y para saber si se trata de de administrador o invitado se compara literalmente con admin@admin y invitado@invitado respectivamente. Esto debe ser rediseñado para poder tener mayor flexibilidad en cuanto al nombre y contraseña de dichos usuarios.
Los registros obligatorios que deben estar en la base de datos para que sacme funcione están descritos en la tabla siguiente.

**Tabla 4. 5** Registros obligatorios por defecto en la tabla usuarios en la base de datos.

| Usuario | E-mail | Contraseña | Contraseña Encriptada |
|---|---|---|---|
| Administrador | admin@admin | root | 3Hbp8MAAbo+RngxRXGbbujmC94U= |
| Invitado | invitado@invitado | "no tiene" | DAQ4otdwBReJy6/dR/4lqdf3RYc= |

**Características de usabilidad que se debe mejorar en SACME para mayor seguridad.**

- Permitir a los usuarios escoger su propio nombre de usuario.
- Permitir a los usuarios escoger su propia contraseña.
- Las contraseñas deben tener como mínimo 6 caracteres.
- Facilitar el cambio de contraseña.
- No cambiar el sistema de contraseñas a menos que sea indispensable.
- Permitir el almacenamiento en cookies para operaciones de baja seguridad.
- Permitir confirmación de una correcta operación, Ejemplo un E-mail.
- Incluir botón de LOGOUT para usuarios.
- Identificación de usuarios conectados.

### 4.2.4 El controlador meSACME

El controlador de meSACME, es el que se encargado de recibir las solicitudes que la aplicación hace en formato cadena y las convierte en peticiones Modbus/TCP para llevar a cabo la comunicación con el módulo electrónico que habla el mismo lenguaje, es decir, es el interprete bidireccional entre la aplicación usuario y el módulo electrónico.

La manera en la que el controlador está implementado en este trabajo de graduación como primera versión, hace lento el proceso de respuesta cuando se trata de una lista grande de recursos a los que se requiera intervenir. Si por ejemplo se ha seleccionado una lista de n recursos, cada recurso abre una conexión socket a través del puerto 5020 y envía una única solicitud y se espera la respuesta, luego cierra el socket, el siguiente recurso vuelve a abrir la conexión y realiza la misma situación anterior y de manera sucesiva los demás recursos hasta tener la respuesta de la n solicitud, para luego presentar en pantalla la información completa de todos los resultados. Esto puede ser impaciente para el usuario y no tendría forma de saber si se esta llevando a cabo el procedimiento o simplemente quedo colgada la pagina.

Para evitar tal situación es necesario que una nueva versión de meSACME sea implementado, pero ¿cual seria su modificación? Primero habrá que tener en cuenta la identidad del cliente que hace la petición, porque el controlador debe mantener la característica de multi hilo, de tal manera que este servicio esperaría que un cliente le envía una lista de recursos así como la acción que se va a ejercer sobre él, y toda la información necesaria para llevar a cabo la solicitud, luego el servicio le responderá inmediatamente al cliente con una página html indicándole que la solicitud se esta procesando para que el usuario este informado que ha sido aceptada su solicitud, toda esta lista debe ser enviada como petición en el protocolo conocido por el módulo electrónico, luego una ves que el controlador haya recibido la respuesta completa para toda la lista por parte del módulo electrónico, el mismo servicio le avisará al correspondiente cliente que la solicitud esta procesada, por ultimo la respuesta debe ser presentada en pantalla del usuario ya sea porque el mismo servicio se lo ha enviado directamente o porque el cliente tiene la llave para revisar el paquete de respuesta en la base de datos.

Resumiendo lo que anteriormente se describe, meSACME tendrá que:

- Asignar un código a cada cliente para identificar su correspondiente solicitud.
- Responderle inmediatamente con una pagina html que le indique espera o procesando al usuario.
- Almacenar en la base de datos el resultado de las solicitudes vinculándole su correspondiente código de cliente.
- Avisarle al cliente cuando la respuesta se encuentre terminada.
- Despachar la respuesta al cliente.

Los archivos que están involucrados en el servicio controlador del módulo electrónico se encuentran en la ruta sacme\WEB-INF\classes\tesis\servicios\modbus y son especificados en la tabla siguiente.

**Tabla 4. 6** Clases del servicio meSACME.

| Clase | Propósito |
|---|---|
| Sacme_ME.class | Clase principal que tiene la responsabilidad atender múltiples clientes simultáneamente y obtener los parámetros necesarios para construir la petición Modbus/TCP |
| ModbusFunc.cass | Recibe como parámetro los datos necesarios para construir la petición e implementa un cliente Modbus/TCP |

# CONCLUSIONES

- El diseño implementado del hardware del SACME tiene las características que le permiten ser utilizado en cualquier aplicación que contenga un cliente Modbus/TCP.

- El hardware sigue una implementación de forma modular o en bloques, esto facilita el uso del mismo y le da una amplia gama de aplicaciones en cuanto a cantidad de cargas se necesite controlar ya que puede funcionar perfectamente con una carga como con todas las que permite su capacidad.

- La multiplexación de los pines del microcontrolador es una excelente forma de aumentar la capacidad del hardware.

- La comunicación serial del módulo electrónico con las demás partes del hardware es otra parte importante de la implementación, pero esta debe ser cuidadosamente tratada ya que una transmisión serial es más vulnerable al ruido.

- Con la tecnología java, se logran diversas ventajas para desarrollar un software flexible e independiente de la plataforma, esto hace portable al sistema desarrollado. También java está diseñado específicamente para trabajar sobre una red, de modo que incorpora objetos que permiten acceder a archivos en forma remota.

- El uso de un motor de persistencia hace posible la independencia de base de datos que estemos utilizando, aunque el desarrollo se hizo en MySQL, el motor de persistencia hibernate nos permite fácilmente trabajar con otro gestor de bases de datos ya sea de código abierto o comercial.

- La arquitectura de n-capas nos da mayor flexibilidad al software desarrollado, ya que permite un mantenimiento fácil, una adaptación a sistemas ya establecidos y una mayor depuración de errores.

- Con la utilización de software de código abierto nos despreocupamos por el uso de licencia.

- Aprovechando las ventajas de la red ethernet y el uso de protocolos sobre tcp/ip como es el caso de Modbus/TCP nuestro sistema es mas robusto ya que puede ser adaptado fácilmente a terceros como por ejemplo Labview de Nacional Instruments

# REFERENCIAS BIBLIOGRAFICAS

Definición de una arquitectura software para el diseño de aplicaciones web basadas en Tecnología Java-J2EE, Daniel Fernández Lanvin, Universidad de Oviedo, España.

"Persistencia de los Objetos Java utilizando Base de Datos relacionales", Pablo Emanuel Goette. y Cristian Galeano.

Tutorial de JavaServer Pages, Miguel Angel García, Edición y distribución: javaHispano.

Manual Hibernate, Héctor Suárez González, Edición y distribución: javaHispano.

## Recursos encontrados en Internet

http://javahispano.org/

http://www.java.net

http://java.sun.com/

http://es.wikipedia.org/

http://www.programacion.com/java/tutorial/j2ee/

http://www.hibernate.org

# ANEXOS

# A 1 MANUAL DE USUARIO SACME

## A 1.1 ACERCA DE ESTA GUIA

### A 1.1.1 Propósito de la Guía

Esta Guía contiene toda la información necesaria para configurar y usar el sistema SACME.

### A 1.1.2 Resumen de secciones.

El contenido de esta guía se encuentra resumido en la tabla siguiente.

**Tabla A. 1** Resumen de contenidos en el Manual de Usuario.

| Sección | Propósito |
|---|---|
| *Configuración del Módulo Electrónico.* | Provee la forma como se configura un módulo electrónico en la red ethernet. |
| *Instalación de la aplicación SACME en Tomcat.* | Provee la forma como se instala la aplicación en el servidor Tomcat |
| *Puesta n Marcha la Aplicación SACME.* | Provee los pasos para levantar los servicios necesarios para que SACME funcione. |
| *La Interfaz del Sitio Web SACME.* | Provee la información descriptiva de las pantallas principales visualizadas par el usuario administrador. |

### A 1.1.3 Información adicional.

Toda la documentación completa se encuentra disponible en el CD adjunto a este documento incluyendo herramientas de software importante, código fuente de programas utilizados y todos los archivos necesarios que comprende SACME.

## A 1.2 CONFIGURACION DEL MODULO ELECTRONICO

El servidor embebido en el módulo electrónico usa el protocolo TCP/IP para comunicarse en la red ethernet, soporta ARP, UDP, TCP, ICMP, Telnet, TFTP, DHCP, y SNMP. Toda la información a cerca del servidor Modbus/TCP se encuentra en el CD adjunto a este trabajo de graduación. Es necesario configurarlo manualmente para personalizar algunos

parámetros y ajustarlo a la aplicación SACME; para hacerlo es necesario usar el modo de configuración que el servidor tiene disponible de fábrica.
Se hace a través de una conexión TELNET sobre la red ethernet[18]; toda la información es guardada en la memoria no volátil que posee el módulo Xport.

### A 1.2.1 Buscando un Módulo Electrónico en la Red

Cuando existe un nuevo módulo electrónico es necesario configurarle algunos parámetros, pero para ello se necesita saber que IP tiene asignado en ese preciso momento. La clave es revisar el hardware en el módulo electrónico, ahí esta embebido el dispositivo Xport y en la etiqueta tiene impreso su dirección MAC.

El Xport ocupado en este Trabajo de Graduación tiene la dirección MAC.

MAC: 02-20-4A-8F-64-5A

Se tiene dos opciones para satisfacer el problema.

- Utilizando el software *Device Installer* de Latronix (disponible en el CD adjunto), leer la documentación para el uso del mismo.

- Ocupando un analizador de protocolos en redes Ethernet, tal es el caso de Ethereal solo basta capturar en modo promiscuo y revisar que IP le corresponde el dispositivo con la dirección MAC que se esta buscando.

### A 1.2.2 Accesando a Modo configuración

Para configurar el módulo electrónico sobre la red es necesario establecer una conexión telnet por el puerto 9999.

**Para establecer una conexión telnet**

1. Del menú Inicio de Windows, hacer click en *Ejecutar* y escriba el siguiente comando, donde x.x.x.x es la dirección IP y 9999 es el puerto fijo de configuración del módulo electrónico.

| | |
|---|---|
| Windows: | telnet x.x.x.x 9999 |
| UNIX: | telnet x.x.x.x:9999 |

   La IP que en este trabajo de graduación se ocupó es IP: 192.168.1.50

2. Hacer Click en *Ok*, la siguiente información aparecerá en pantalla.

[18] Información detallada a cerca de la configuración del servidor Modbus/TCP embebido esta disponible en los documentos adjuntos a este trabajo de graduación en la carpeta XPORT.

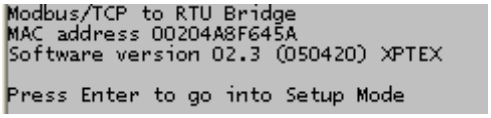


**Figura A 1. 1** Pantalla de inicio en Modo Configuración del Módulo Electrónico.

3. Para entrar en modo configuración, presione *Enter* antes de 5 segundos. Aquí se cambian los parámetros que se necesitan personalizar.

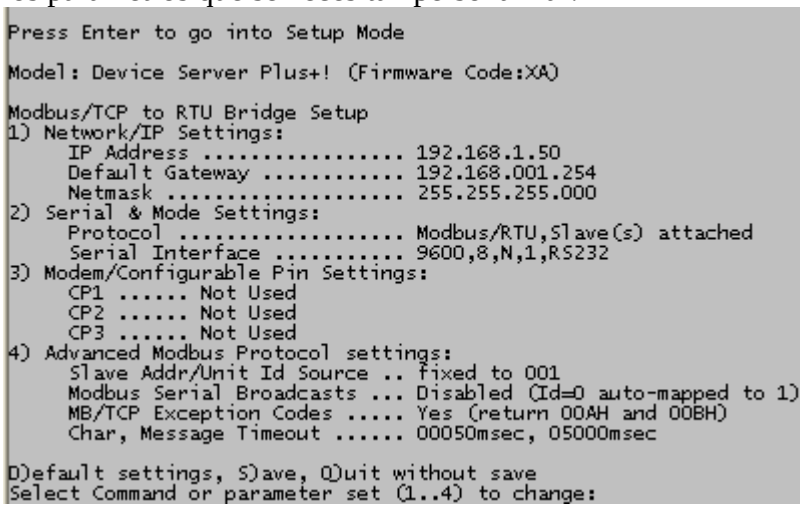


**Figura A 1. 2** Pantalla de Opciones en Modo Configuración del Módulo electrónico.

### A 1.2.3 Configuración IP del servidor Modbus/TCP

Seleccione 1 para configurar los parámetros de red del módulo electrónico y siga las instrucciones de la pantalla.
La IP debe ser un valor único dentro de la red. Si la IP que se le asigne ya esta siendo ocupada por otro dispositivo se presenta en los leds un código de Error que puede ser consultado en la guía de usuario del Xport en el CD adjunto. *No debe ser activado en modo DHCP*, puede causar fallo de conexión, porque en la aplicación sacme debe poseer IP fija.

Los demás parámetros (2..4) afortunadamente no son necesarios modificarlos. En caso de requerir modificaciones en algún parámetro específico, refiérase a la documentación para el Xport en el CD adjunto. El último paso es guardar cambios y salir.

### A 1.2.4 Salir Guardando la configuración

Presione la tecla S y automáticamente el módulo guarda la configuración en la memoria no volátil y se reinicia.

El Módulo esta ahora configurado y listo para que el SACME funcione satisfactoriamente.

## A 1.3 INSTALACION DE LA APLICACION SACME EN TOMCAT

La aplicación SACME esa disponible el CD adjunto, la carpeta contenedora de la aplicación también tiene el mismo nombre sacme, además se proporciona el archivo war (Web-Archives) donde esta contenida toda la estructura de un sitio web en un solo archivo comprimido.

Cuando se lleva acabo la ejecución de Tomcat éste inspecciona y automáticamente expande cualquier archivo WAR que se encuentra bajo el directorio `webapps`.

La otra forma de instalar el sitio web en tomcat es copiar la carpeta sacme en el directorio *webapps.*

.

En la siguiente figura podemos ver la estructura de directorios de la aplicación Web:

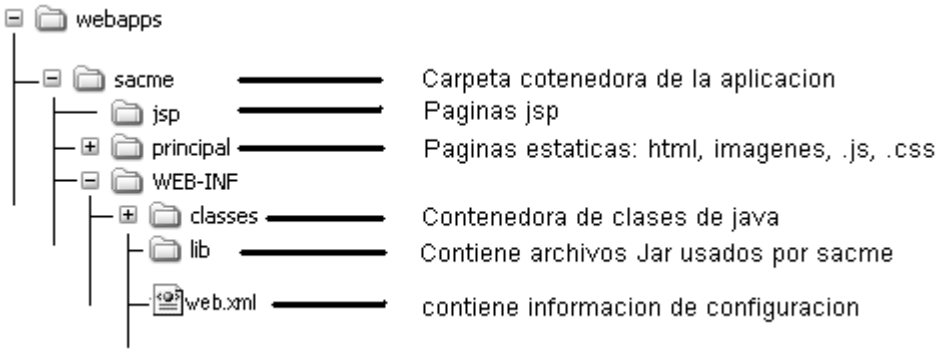


**Figura A 1. 3** Estructura de directorios de la aplicación SACME.

La versión del Tomcat utilizado en este trabajo de graduación es apache-tomcat-6.0.13 también proporcionado en el CD adjunto

Bajo Linux, por ejemplo, si hemos instalado tomcat bajo el directorio /opt/apache-tomcat-6.0.13, deberíamos copiar las la carpeta raíz de la aplicación debajo del directorio:

```
/opt/apache-tomcat-6.0.13/webapps/
```

Si ejecutamos tomcat bajo Windows y los hemos instalado en el directorio C:\Archivos de programa\apache-tomcat-6.0.13, deberíamos copiar las la carpeta raíz de la aplicación debajo del directorio:

```
C:\Archivos de programa\apache-tomcat-6.0.13\webapps\
```

Esto es todo lo necesario, no hay más configuración.

## A 1.4 PUESTA EN MARCHA LA APLICACIÓN SACME

Para poner en marcha la aplicación sacme, se debe arrancar el servidor Tomcat, abrir un navegador Web y escribir una URL con la siguiente forma:

```
http://{direccion}:{puerto}/{nombreAplicacion}
```

Donde:

- dirección, es el nombre o dirección IP de la máquina que está ejecutando Tomcat. Podemos usar localhost si el navegador se está ejecutando sobre la misma máquina que Tomcat.
- puerto, es el puerto en el que escucha Tomcat. Por defecto, es el puerto 8080.
- nombreAplicacion, es el nombre del sitio web que queremos invocar. Debería corresponder al valor contenido en las etiquetas <url-pattern></url-pattern> del fichero descriptor de despliegue configurado en el archivo web.xml.

Por ejemplo, si Tomcat se está ejecutando en la misma máquina que el navegador y está escuchando el puerto por defecto (8080), podemos probar invocar nuestro sitio web abriendo la siguiente URL:

```
http://localhost:8080/sacme
```

### A 1.4.1 Servicio meSACME

Es necesario levantar el servicio controlador del módulo electrónico de la manera siguiente:

1. En el menú Inicio de Windows, hacer click en *Ejecutar* y escriba el comando "cmd" y enter.
2. Cambiar de directorio al directorio donde se encuentra las clases del servicio meSACME que esta en la siguiente ruta.

   ```
   C:\.....\apache-tomcat 6.0.13\webapps\sacme\WEB-INF\classes\tesis\servicios\modbus
   ```

3. Inicializar las variables de entorno. En la línea de comandos ejecutar el archivo.

   ```
   envconfig.bat
   ```

   **nota:** este archivo debe ser modificado de acuerdo a las rutas de instalación de directorios en el equipo en custion.

4. Ejecutar la aplicación de consola del servicio así.

   ```
   java  Sacme_ME
   ```

La información siguiente deberá aparecer en pantalla.

```
C:\Archivos de programa\Apache Software Foundation\Tomcat 6.0\webapps\sacme
\WEB-INF\classes\tesis\servicios\modbus>java Sacme_ME
Controlador Sacme_ME
 Inicializado...
```

**Figura A 1. 4** Información en pantalla cuando meSACME está levantado.

### A 1.4.2 Servicio aeSACME

Para que los eventos se ejecuten a la hora que está programado según la base de datos, el servicio administrador de eventos debe estar lanzado, para ello seguir los siguientes pasos.

1. En el menú Inicio de Windows, hacer click en *Ejecutar* y escriba el comando "cmd" luego Enter.
2. Cambiar de directorio al directorio donde se encuentra las clases del servicio aeSACME que esta en la siguiente ruta.

   ```
   C:\.....\apache-tomcat 6.0.13\webapps\sacme\WEB-
   INF\classes\tesis\servicios\eventos
   ```

3. inicializar las variables de entorno. En la línea de comandos ejecutar el archivo.

   ```
   envconfig.bat
   ```

4. ejecutar la aplicación de consola del servicio así.

   ```
   java  Sacme_AE
   ```

El siguiente menú aparecerá en pantalla.

```
*****************************************************************************
                  Servicio Administrador de Eventos
*****************************************************************************

   CONFIGURAR TIPO DE DISPARO

     1. Cada N Minutos

     2. A determinada Hora del Dia

     3. En este Momento (Solo una ves)


Seleccione por Favor (1 , 2 o 3) _?
```

**Figura A 1. 5** Pantalla de configuración del servicio Administrador de Eventos.

Donde:

1. Cada N Minutos: se le indica al servicio que cada N minutos debe revisar la base de datos y ejecutar todos los eventos que se encuentren dentro del intervalo de N minutos a partir de la hora actual, por ejemplo, si la hora actual fuera 10:00 y N=5 se ejecutaran los eventos programados para las 10:00, 10:01, 10:02, 10:03 10:04 luego revisa de nuevo la base de datos para el intervalo de 10:05 hasta 10:09.
2. A determinada Hora del Día: el servicio revisa la base de datos a la hora que se especifique, la siguiente revisión a la base de datos será el siguiente día a la hora especificada.
3. En este Momento (Solo una vez): aeSACME revisa la base de datos y ejecuta todos los eventos para ese día solamente a la hora indicada.

## A 1.5 LA INTERFAZ DEL SITIO WEB SACME

Una vez que el usuario se ha identificado en la página de inicio, el navegador le muestra al usuario la siguiente pantalla, que para el caso de un administrador seria.

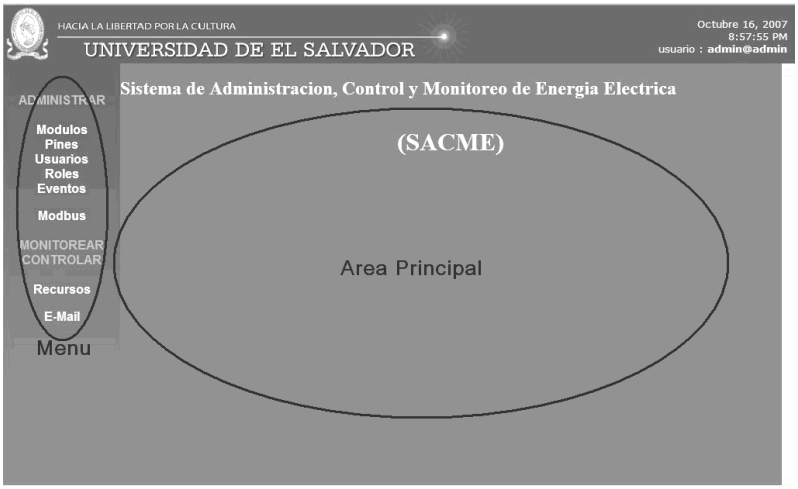


**Figura A 1. 6** Pantalla principal del usuario Administrador.

Las dos secciones importantes son las resaltadas con un círculo en la figura anterior, en la parte izquierda se encuentra el Menú de Opciones y en la parte central es donde se muestra toda la información de la opción seleccionada.

### A 1.5.1 Listar Recursos

Se encuentra como un enlace en el área de menú con la etiqueta “Recursos”, al hacer click en el hipervínculo se muestra una lista de recursos disponible para el usuario actual, si el usuario es el administrador se le proporciona la lista total de registros que hay en la base de

datos para los recursos. La pantalla se deberá actualizar similar a la mostrada en la figura siguiente.


**Figura A 1. 7** Pantalla de listado de Recursos.

Donde:
*Seleccionar*: checkbox para seleccionar el recurso que queremos monitorear o controlar.
*Estado*: Desconocido, ON, OFF o en caso de error nos presenta su respectiva excepción.
*Descripción*: información descriptiva a cerca del recurso respectivo.
*# pin / Modulo*: indica que numero físico de pin y a que módulo electrónico pertenece.

## A 1.5.2 Nuevo Usuario

Esta tarea es exclusiva del administrador, y basta con seguir la secuencia de pantallas, una vez se haga clic en el hipervínculo "Usuarios" se muestra en la página actual la lista de todos los usuarios que están registrados en el sistema, permitiendo poder Modificar y Eliminar a cada usuario y añadir uno nuevo.


**Figura A 1. 8** Pantalla de listado de Usuarios registrados en SACME.

Al hacer clic en el enlace "Añadir un Nuevo Usuario" (marcado con un círculo en el área principal), se presenta un formulario que debe ser llenado con los respectivos datos.

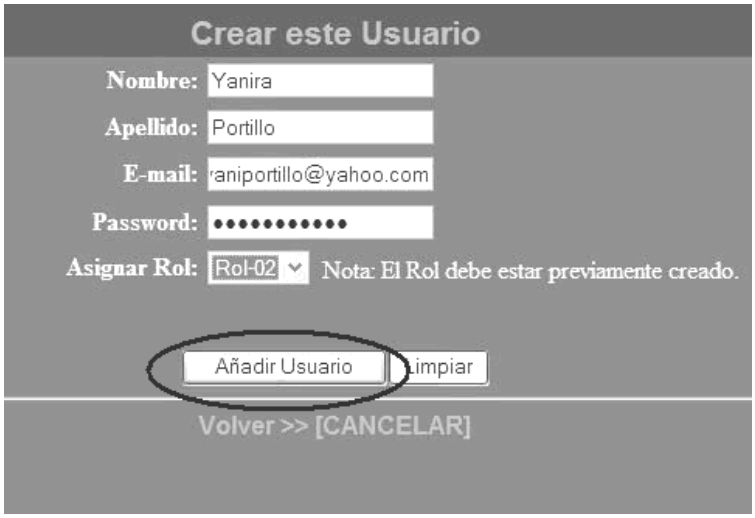


**Figura A 1. 9** Formulario para crear un usuario nuevo.

Una vez llenado el formulario se hace clic en el botón "Añadir Usuario", si es satisfactoria la información queda guardada en la base de datos y desde luego un nuevo usuario esta en la lista de usuarios.


**Figura A 1. 10** El nuevo Usuario aparece en la lista de usuarios registrados en SACME.

### A 1.5.3 Nuevo Evento

Para ingresar un nuevo evento el procedimiento es similar que para el caso de un usuario nuevo por lo tanto no es relevante repetir los pasos anteriores, solo lo que se hace diferente es el formulario de datos del evento, aunque no requiere de mayor explicación se hace una breve descripción de este.

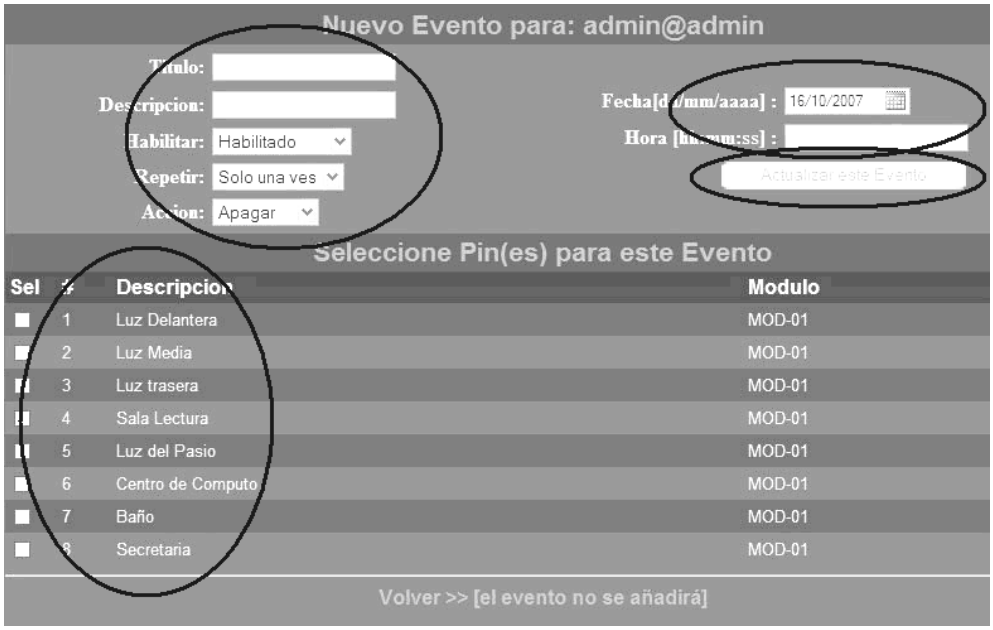


**Figura A 1. 11** Formulario para añadir nuevo evento.

Los círculos resaltan las partes importante del formulario, los datos propios del evento:

*Titulo*: es un nombre único que no puede repetirse en la base de datos e identifica al evento.

*Descripción:* información general del evento.

*Habilitar:* Habilitado o deshabilitado, indica si el evento se debe ejecutar o solo esta preprogramado para no tener que ingresar de nuevo los datos en un futuro.

*Repetir:* Solo una vez, Diario, Semanal; frecuencia en la que se da el evento.

*Acción***:** Apagar, Encender; se ejecuta sobre el o los recursos seleccionados.

El botón "Añadir este Evento", se activa hasta que por lo menos un recurso ha sido seleccionado de la lista de recursos disponible para el usuario correspondiente, de lo contrario no permitirá añadir el evento.

### A 1.5.4 Cliente Modbus

Utilidad agregada para ayudar al administrador a hacer consulta rápida o control rápido de recursos, este agregado requiere que el administrador tenga conocimiento del protocolo Modbus, porque la respuesta esta en el formato que el protocolo exige.


**Figura A 1. 12** Cliente Modbus/TCP desde la web.

*IP Addres:* es la dirección IP del Módulo electrónico al que se quiere investigar algún recurso que le pertenece.

*Port:* es el puerto reservado de Modbus/TCP 502.

*Dir. Inicio*: es la dirección de inicio física en el hardware del módulo electrónico. Por ejemplo si se quiere saber el estado de la salida 3 (recurso 3), es necesario que la dirección de inicio sea 0 y el tercer bit de la respuesta indica el estado de la salida (0=OFF y 1=ON), porque la función 02 regresa el estado de grupos de 8 salidas a la vez. Si se requiere la salida 10 la dirección de inicio sea 8 y el bit segundo indica el estado de la salida 10. Ahora si se trata de encender o apagar la salida, la dirección de inicio es la misma salida, o sea 3 para el primer caso y 10 para el segundo.

Cantidad/Estado: "Cantidad" si se trata de la función 02 para monitorear y puede ser el valor que se muestra en la tabla de direccionamiento en la misma pagina. "Estado" si se trata de la función 05 para controlar, 1 si lo que queremos es encender y 0 si el caso fuera apagar.

### A 1.5.5 Mandar E-mail


**Figura A 1. 13** Formulario para enviar correo electrónico.

Una vez teniendo acceso a Internet este utilidad hecha con JavaMail puede ayudar mantener comunicado al administrador de parte de un usuario y puede ser utilizado para manejo de reporte de errores o informes en nuevas versiones, funciona perfectamente para enviar correo electrónico a cualquier contacto incluyendo móviles si la compañía telefónica lo permite.

# A 2 CODIGO FUENTE

## A 2.1 Código Fuente del servicio controlador del Módulo electrónico meSACME

```
/*****************************************************************************************
*       Programa: Sacme_ME.java                                                          *
*       Autor: Gustavo Gómez & Wilman  Villatoro                                         *
*       Version: 1.0      2007                                                           *
*                                                                                        *
*       Descripción:  Clase principal del Servicio Controlador del Modulo Electrónico SACME-ME   *
*****************************************************************************************/

//***************       CLASE DEL SERVIDOR          ***************

import java.net.*;
import java.io.*;

class Sacme_ME {
        ServerSocket s;
        int clientes;
        /*************************************
        *       Lanza la aplicación del servidor      *
        *************************************/
        void lanzarServidor(){
                String cadena;
                mostrarMensaje("\n Controlador Sacme_ME \n Inicializado...");
                try     {
                        s=new ServerSocket(5020,10); //conexión  port 5020 y máximo de 10 clientes
                        while(true)
                                new NuevoCliente(s.accept(),this,clientes++).start();
                }       catch(IOException ioe)     {
                        mostrarMensaje("Error al inicializarse: "+ioe.toString());
                }
        }
        /*************************************
        *       Muestra un mensaje en la pantalla   *
        *************************************/
         public void mostrarMensaje(String texto) {
                System.out.println(texto);
         }
```

```
        /*****************************
        *       Aplicacion de consola       *
        *                                   *
        *       Sintaxis: java Sacme_ME  *
        *****************************/
        public static void main(String args[])          {
                Sacme_ME Servidor=new Sacme_ME();
                Servidor.lanzarServidor();
        }
}

        /*****************************************
        *       La clase nuevo cliente extiende de Thread.    *
        *****************************************/
class NuevoCliente extends Thread{
        Socket conexion;
        Sacme_ME serv;
        int cliente;                    // contador de clientes conectados
        DataInputStream entrada;  //cadena enviada por el cliente
        DataOutputStream salida;  //cadena de respuesta

        NuevoCliente(Socket c,Sacme_ME s,int numero)      {
        //Entrada de un nuevo cliente
                try       {
                        conexion=c;
                        serv=s;
                        cliente=numero;

                        serv.mostrarMensaje("Nuevo Cliente conectado # "+cliente);
                        InetAddress ia=c.getInetAddress();
                        serv.mostrarMensaje(ia.getHostName());
                        System.out.println(ia.getLocalHost());

                        entrada=new DataInputStream(conexion.getInputStream());
                        salida=new DataOutputStream(conexion.getOutputStream());
                }       catch(IOException e)       {
                        s.mostrarMensaje("Error :"+e.toString());
                }
        }
        //método run, sobreescrito para tratar a los clientes
        public void run()  {
                boolean Salir=false;
                serv.mostrarMensaje("Thread run...");

                //añadir el cliente al servidor
                serv.mostrarMensaje("cliente "+cliente+" se ha conectado \n");
                while(!Salir)       {
            try   {
                String cadena=entrada.readUTF();  //Lee  dato del cliente
                        /* Captura de parametros de la cadena solicitud */
                        int pos=0;
                        String direccion,puerto,funcion,ref,count="";
                        String respuesta="Error: Funcion no valida";
                        pos=cadena.indexOf("#");
                        direccion=cadena.substring(0,pos); // lee direccion
```

```
                    serv.mostrarMensaje("IP: "+direccion);

                    cadena = cadena.substring(pos+1);
                    pos=cadena.indexOf("#");
                    puerto=cadena.substring(0,pos);                 //Lee puerto, normalmente 502
                    serv.mostrarMensaje("PORT: "+puerto);

                    cadena = cadena.substring(pos+1);
                    pos=cadena.indexOf("#");
                    funcion=cadena.substring(0,pos);    //Lee funcion modbus 02 o 05
                    serv.mostrarMensaje("FUNCION: "+funcion);

                    cadena = cadena.substring(pos+1);
                    pos=cadena.indexOf("#");
                    ref=cadena.substring(0,pos);          //Lee referencia o direccion de inicio
                serv.mostrarMensaje("REF: "+ref);

                    count = cadena.substring(pos+1);    //lee cantidad para 02 o estado para 05

                    serv.mostrarMensaje("COUNT: "+count);
                    //se crea una instancia para la peticion modbus y se envia la peticion
                    ModbusFunc peticion=new ModbusFunc();
                      if(funcion.equals("2")){
                            respuesta=peticion.funcion02(direccion, ref, count);
                       } else if(funcion.equals("5")) {
                            respuesta=peticion.funcion05(direccion, ref, count);
                        }

                serv.mostrarMensaje(respuesta);     //iprime en pantalla la respuesta enviada al cliente

                    //manda respuesta al cliente
                    DataOutputStream salida;
                    salida=new DataOutputStream(conexion.getOutputStream());
                    salida.writeUTF(respuesta);

                 Salir=true;

                    }       catch(SocketException se) {
                            Salir=true;
                    }       catch(IOException e)        {
                            Salir=true;
                            serv.mostrarMensaje("\n Error:"+e.toString());
                    }
                }
                serv.mostrarMensaje("El cliente " +cliente+"  Se ha desconectado \n");
                try         {
                    conexion.close();
                }       catch(IOException e)        {
                    serv.mostrarMensaje("\nError :"+e.toString());
                }
        }
}
```

## A 2.2 Cliente Modbus/TCP del meSACME

```
/*****************************************************************************
 *                  Autor      Gustavo Gomez &  Wilman Villatoro                *
 *                  Trabajo de Graduacion 2007.                                 *
 *                  version 1.0                                                 *
 *                  Clase que implementa un clinte Modbus/TCP                   *
 *****************************************************************************/

import java.net.*;
import java.io.*;
import net.wimpi.modbus.*;
import net.wimpi.modbus.msg.*;
import net.wimpi.modbus.io.*;
import net.wimpi.modbus.net.*;
import net.wimpi.modbus.util.*;

public class ModbusFunc {

        /*************************************************
        *          FUNCION 02 READ DISCRETE INPUTS       *
        *************************************************/

public  String funcion02(String ip,String dir_inicio, String cantidad) {
          String args[]={ip,dir_inicio,cantidad};
          /* Instancias importantes */
          TCPMasterConnection con = null; //La conexion
          ModbusTCPTransaction trans = null; //La transaccion
          ReadInputDiscretesRequest req = null; //La peticion
          ReadInputDiscretesResponse res = null; //La respuesta

          try {

                  /* Variables para capturar los parametros */
                  InetAddress addr = null;              //direccion del esclavo
                  int port = Modbus.DEFAULT_PORT; // 502 por default
                  int ref = 0;                    //la refenercia; offset donde empieza la lectura
                  int count = 0;                          //el numero de entradas discretas a leer
                  int repeat = 1;                        //lazo para repetir la transaccion

                  /****     1. Captura de los parámetros          ****/
                  if (args.length < 3) {
                     return "Error: Argumentos insuficientes";
                  } else {
                    try {
                     String astr = args[0];
                     int idx = astr.indexOf(':');
                     if(idx > 0) {
                       port = Integer.parseInt(astr.substring(idx+1));
                       astr = astr.substring(0,idx);
                     }
                     addr = InetAddress.getByName(astr);
                     ref = Integer.decode(args[1]).intValue();
                     count = Integer.decode(args[2]).intValue();
                     if (args.length == 4) {
```

```
                        repeat = Integer.parseInt(args[3]);
                      }
                     } catch (Exception ex) {
                      //ex.printStackTrace();
                      //System.exit(1);
                      return "Error :"+ex.getMessage();
                     }
                    }

                    /**** 2. Abrir la conexion ****/
                    con = new TCPMasterConnection(addr);
                    con.setPort(port);
                    con.connect();

                    /****     3. Preparar la Solicitud      ****/
                    req = new ReadInputDiscretesRequest(ref, count);
                    req.setUnitID(1);// setear la ID del esclavo

                    /****     4. Preparar la transacción   ****/
                    trans = new ModbusTCPTransaction(con);
                    trans.setRequest(req);
                    trans.setReconnecting(false);

                    /****     5. Ejecutar la transaccion "repeat" veces       ****/
                    int k = 0;
                    do {
                      trans.execute();
                      res = (ReadInputDiscretesResponse) trans.getResponse();
                      //System.out.println("Response: " + res.getHexMessage() );
          //System.out.println("Digital Inputs Status=" + res.getDiscretes().toString());

                      k++;
                    } while (k < repeat);

                     /****     6. Serrar la conexión          ****/
                     con.close();
          } catch (Exception ex) {
               return "Error :"+ex.getMessage();
              }
            return res.getDiscretes().toString();
}//metodo funcion02


/*****************************************************
*          FUNCION 05 WRITE SINGLE COIL                    *
*****************************************************/

public  String funcion05(String ip,String dir_inicio, String estado) {
          String args[]={ip,dir_inicio,estado};

          /* Instancias importantes */
          TCPMasterConnection con = null; //La conexion
          ModbusTCPTransaction trans = null; //La transaccion
          WriteCoilRequest req = null; //La peticion
          WriteCoilResponse res = null; //La respuesta
```

```
          try {

          /* Variables para capturar los parametros */
          InetAddress addr = null;                //direccion del esclavo
          int port = Modbus.DEFAULT_PORT; // 502 por default
          int ref = 0;                            //la refenercia; offset donde empieza la lectura
          int count=0;                            //ON (1) OFF (0)
          int repeat = 1;                         //lazo para repetir la transaccion
          boolean status= false;

          /****     1. Captura de los parámetros          ****/
          if (args.length < 3) {
            return "Error: Argumentos Insuficientes";
          } else {
            try {
              String astr = args[0];
              int idx = astr.indexOf(':');
              if(idx > 0) {
                port = Integer.parseInt(astr.substring(idx+1));
                astr = astr.substring(0,idx);
              }
              addr = InetAddress.getByName(astr);
              ref = Integer.decode(args[1]).intValue();
              count=Integer.decode(args[2]).intValue();
                      if(count==1)
                      status = true;

              if (args.length == 4) {
                repeat = Integer.parseInt(args[3]);
              }
            }          catch (Exception ex) {
              return "Error :"+ex.getMessage();
            }
          }

          /****     2. Abrir la conexión          ****/
          con = new TCPMasterConnection(addr);
          con.setPort(port);
          con.connect();

          /****     Preparar la Solicitud          ****/
          req = new WriteCoilRequest(ref, status);
          req.setUnitID(1);// setear la ID del esclavo

          /****     4. Preparar la transacción   ****/
          trans = new ModbusTCPTransaction(con);
          trans.setRequest(req);

/****     5. Ejecutar la transaccion "repeat" veces       ****/
int k = 0;
do {
  trans.execute();
  res = (WriteCoilResponse) trans.getResponse();
  //out.print("Digital Inputs Status=" + res.getDiscretes().toString());
  k++;
} while (k < repeat);
```

```
/****    6. Serrar la conexión          ****/
        con.close();

   } catch (Exception ex) {
     return "Error :"+ex.getMessage();
   }
        String respuesta = "OFF";
        if(res.getCoil())
        respuesta="ON";
   //return res.getHexMessage();
        return respuesta;
  }//metodo funcion05

}//clase ModbusFunc
```

## A 2.3 Código del servicio administrador de Eventos aeSACME

```
/*****************************************************************************
*       Programa: Sacme_AE.java                                                 *
*       Autor: Gustavo Gomez    &       Wilman Villatoro                        *
*       Version: 1.0/2007                                                       *
*                                                                               *
*       Descripción: Programa Servicio Administracion de Eventos  SACME-AE      *
*                                                                               *
*****************************************************************************/

//package tesis.servicios.eventos;
import tesis.datos.*;
import tesis.dominio.*;
import tesis.negocio.*;
import tesis.servicios.eventos.*;

import java.text.SimpleDateFormat;
import java.io.*;
import java.util.*;
import java.net.*;

//**************        CLASE DEL SERVIDOR ***************
public class Sacme_AE{

        /*********************************************
        *       Metodo Principal: Aplicacion de consola
        *       Modo de uso: java Sacme_AE      <ENTER>
        *********************************************/
        public static void main(String Arg[ ]) throws Exception{

                int selsen=1;
                int seldis=3;
                int intervalo=0;
                Thread timin = new Thread();
                String hora;
                int minutos;
```

```
        Calendar tiempo = Calendar.getInstance();

        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("*****************************************************************");
        System.out.println("                Servicio Administrador de Eventos");
        System.out.println("*****************************************************************");
        System.out.println("\n");
        System.out.println("   CONFIGURAR TIPO DE DISPARO\n");
        System.out.println("     1. Cada N Minutos\n");
        System.out.println("     2. A determinada Hora del Dia \n");
        System.out.println("     3. En este Momento (Solo una ves) \n");
        System.out.println("\n");
        System.out.println("Seleccione por Favor (1 , 2 o 3) _?");
        System.out.println("\n");
        seldis = Integer.parseInt(in.readLine( ));
        System.out.println("\n");
        switch(seldis){
                case 1:   //Caso 1: El disparo lo realiza cada intervalo de n minutos
                System.out.println("Hora :"+tiempo.get(tiempo.HOUR_OF_DAY)+":
                        "+tiempo.get(tiempo.MINUTE)+":"+tiempo.get(tiempo.SECOND));
                                System.out.println("Digite el Intervalo en Minutos (ej. 5) \n");
                                minutos = Integer.parseInt(in.readLine());
                                while (true){
                                        tiempo = Calendar.getInstance();
                                System.out.println("Verificando Base de Datos... \n");
                                        System.out.println("HORA ACTUAL:
                                         +tiempo.get(tiempo.HOUR_OF_DAY)+":
                                         "+tiempo.get(tiempo.MINUTE)+":
                                         "+tiempo.get(tiempo.SECOND)+"\n");
        List nextEvent = proximosEventos(minutos);//obtener la lista de eventos a ejecutarse
                                ejecutarEventos(nextEvent);
                                timin.sleep(minutos*60000);          //60000 milisegundos = 1 minuto

                                }
                case 2:   //Disparo a determinada hora del dia
                                System.out.println("Hora :
                        "+tiempo.get(tiempo.HOUR_OF_DAY)+":
                        "+tiempo.get(tiempo.MINUTE)+":"+tiempo.get(tiempo.SECOND));
                                System.out.println("Digite la Hora (ej. 08:00) \n");
                                hora = in.readLine();
                                while (true){
                                        tiempo = Calendar.getInstance();
                                while (Integer.parseInt(hora.substring(0,2)) ==
                tiempo.get(tiempo.HOUR_OF_DAY)
                 && Integer.parseInt(hora.substring(3,5)) == tiempo.get(tiempo.MINUTE)){
                // Aqui lo que se ejecutara a una hora determinada de cada dia
List nextEvent = proximosEventos(5);//obtener la lista de eventos a ejecutarse proximos 5 min
                                        ejecutarEventos(nextEvent);
                                        timin.sleep(60000);
                                        }
                                        timin.sleep(30000);
                                }

                case 3: //Disparo inmediatamente solo una ves
                                System.out.println("Hora :
                                  "+tiempo.get(tiempo.HOUR_OF_DAY)+":
```

```
                                                "+tiempo.get(tiempo.MINUTE)+":
                                                  "+tiempo.get(tiempo.SECOND));
                                                System.out.println("Verificando la Base de Datos... \n");
                                                //Aqui lo que se ejecutara inmediatamente y solo una ves
                                        List nextEvent = proximosEventos(5);//obtener la lista de eventos a ejecutarse
                                                        ejecutarEventos(nextEvent);
                                                break;
                                }

                        }

                        /**********************************************************
                        *       Ejecutar la lista de  eventos correspondientes
                        *
                        **********************************************************/
  public static void ejecutarEventos(List listaOk) { //levantar un thead por cada evento
                                int pos=0;
                                String hora;
                                if (listaOk != null) {
                                        if (listaOk.size() > 0) {
                                          Iterator iterator = listaOk.iterator();
                                          while (iterator.hasNext() ) {
                                                String cadena = (String)iterator.next ();
                                                pos=cadena.indexOf("#");
                                                hora=cadena.substring(0,pos);         // lee HORA
                                        cadena = cadena.substring(pos+1);//cadena: contiene la peticion modbus
                                                new EjecutarEventos(hora,cadena).start(); // ejecutar los eventos
                                                }
                                        }
                                }
                        }

                        /*********************************************
                        *       proximosEventos correspondiente a
                        *       Rango del intervalo en minutos
                        *********************************************/
                        public static List proximosEventos(int minutos){
                                SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yy HH:mm:ss");
                                EventoDAO daoEventos = new EventoDAO();
                                //List listaOk =daoEventos.obtenerTabla();
                                List<String> allList = new ArrayList<String>();
                                List listaOk =daoEventos.listaEventos("Habilitado"); //solo los habilitados
                                        if (listaOk != null) {
                                          if (listaOk.size () > 0) {
                                                Iterator iterator = listaOk.iterator();
                                                while (iterator.hasNext() ) {
                                                        Evento evento = (Evento) iterator.next ();
                                                                //Filtrar Eventos
                                                                boolean isOK =
filtrarEvento(formato.format(evento.getHorario()).toString(), evento.getRepeticion(), minutos);
                                                                if(isOK){
                                                                //obtener  informacion de este evento
                                                                List exeEventos=infoEvento(evento);
                                                                //anexar a la lista total
                                                                allList.addAll(exeEventos);
                                                                }
```

```
                                        }
                                }
                        }   else {
                                System.out.println("No hay ningun evento Pendiente");
                        }
                return allList;
        }


        /**************************************************************************
        *       Obtiene la informacion necesaria para cada evento a ejecutarse                  *
        **************************************************************************/
        public static List infoEvento(Evento evento){
                List<String> newList = new ArrayList<String>();
                //a cada pin obtener el numero de pin y el modulo al que pertenece
                PinesNgc ngcPines = new PinesNgc();
                ModuloNgc ngcModulo = new ModuloNgc();
                String count = "0";                                             //por defecto apagar
                //obtener la hora del evento
                SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yy HH:mm:ss");
                String fechayhora=formato.format(evento.getHorario()).toString();
                String horaDelEvento = fechayhora.substring(9,14);
                Collection misPines=evento.getPinesXevento();
                        if(misPines!=null){
                                Pines pin;
                                Iterator iterador = misPines.iterator();
                                while(iterador.hasNext()){
                                        pin = (Pines)iterador.next();
                                        int numPin=(int)pin.getNumero(); //numero de pin
                                        System.out.println(numPin);

                                        String nombreModulo =
ngcPines.obtenerModuloPropietario((int)pin.getId());
                                        String ipModulo = ngcModulo.obtenerIpModulo(nombreModulo);//IP
del modulo

                                        int ref = ((int)pin.getNumero() -1);
                                        if(evento.getAccion().equals("Encender")){
                                                count = "1";
                                        }else {
                                                count = "0";
                                        }
                                        // Añadimos elementos

        newList.add(horaDelEvento+"#"+ipModulo+"#502#5#"+ref+"#"+count);
                                }
                        }
                return newList;
        }
.
```

```
        /*****************************************************
        *       Filtra los Eventos
        *       fecha: formato dd/MM/yy HH:mm:ss
        *       repeticion:     "Solo una ves", "Diario", "Semanal"
        *       intervalo: rango de minutos en que se da los eventos
        *****************************************************/
        public  static boolean filtrarEvento(String fecha, String repeticion, int intervalo){
                int frecuencia = 32;
                Calendar tiempo = Calendar.getInstance();
                int minDelEvento =  Integer.parseInt(fecha.substring(12,14)) ;
                int minActual =tiempo.get(tiempo.MINUTE);
                int diaDelEvento = Integer.parseInt(fecha.substring(0,2));
                if(repeticion.equals("Solo una ves")){                  //Solo una ves
                        frecuencia = 32;   //forzar a ser mayor de un mes
                } else if(repeticion.equals("Diario")){                 //Diario
                        frecuencia = 1;
                } else if(repeticion.equals("Semanal")){        //Semanal
                        frecuencia = 7;
                }
                        while(diaDelEvento <= tiempo.get(tiempo.DAY_OF_MONTH)){

                                if(diaDelEvento == tiempo.get(tiempo.DAY_OF_MONTH)){ // Coincide dia
del evento y hoy
                                        if(Integer.parseInt(fecha.substring(9,11)) ==
tiempo.get(tiempo.HOUR_OF_DAY)){
                                                if(minDelEvento >= minActual && minDelEvento <
minActual + intervalo){
                                                        return true;
                                                }
                                        }
                                        if(minActual + intervalo >= 60){
                                                int minAux = (minActual+intervalo) - 60;
                                                if(Integer.parseInt(fecha.substring(9,11)) ==
tiempo.get(tiempo.HOUR_OF_DAY)+1){
                                                        if(minDelEvento < minAux ){
                                                                return true;
                                                        }
                                                }
                                        }

                                }
                                diaDelEvento = diaDelEvento + frecuencia;
                        }       return false;
        }

/*****************************************************************************
*        Crea el socket cliente y lee envia la Peticion en una cadena                                *
*       Ip#Puerto#Funcion#Dir_Inicio#(Count/estado) al meSACME                                      *
*****************************************************************************/
   public   static String sacmeCliente(String request)   {
                    String respuesta="";
                try  {
                   /* Se crea el socket cliente */
                   Socket socket = new Socket ("localhost", 5020);//para conectarse al servicio.

                                DataOutputStream salida;
```

```
                                        salida=new DataOutputStream(socket.getOutputStream());
                                        salida.writeUTF(request);

                                        DataInputStream entrada;
                                        entrada=new DataInputStream(socket.getInputStream());
                                        respuesta=entrada.readUTF(); //Lee  del servidor
                    } catch (Exception e)  {
                                        respuesta="          Error al leer del servidor...";
                                        return respuesta;
                       // e.printStackTrace();
                    }
                         return respuesta;
       }

}
```

## A 2.4 Código Script para crear las tablas  y los Registros por defecto.

```
-- MySQL dump 10.11
--
--
-- Table structure for table `eventos`
--

DROP TABLE IF EXISTS `eventos`;
CREATE TABLE `eventos` (
  `id_evento` int(11) unsigned NOT NULL auto_increment,
  `descripcion` varchar(50) NOT NULL,
  `horario` datetime default NULL,
  `estado` varchar(20) NOT NULL,
  `repeticion` varchar(20) NOT NULL,
  `accion` varchar(20) NOT NULL,
  `id_user_FK` int(10) unsigned default NULL,
  `titulo` varchar(10) NOT NULL,
  PRIMARY KEY  (`id_evento`),
  KEY `Eventos1N` (`id_user_FK`),
  CONSTRAINT `Eventos1N` FOREIGN KEY (`id_user_FK`) REFERENCES `usuarios` (`id_usuario`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;



--
-- Table structure for table `modulos`
--

DROP TABLE IF EXISTS `modulos`;
CREATE TABLE `modulos` (
  `id_modulo` int(10) unsigned NOT NULL auto_increment,
  `descripcion` varchar(50) default NULL,
  `ip` varchar(50) NOT NULL,
  `nombre` varchar(50) NOT NULL,
  PRIMARY KEY  (`id_modulo`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
--
-- Dumping data for table `modulos`
--

LOCK TABLES `modulos` WRITE;
/*!40000 ALTER TABLE `modulos` DISABLE KEYS */;
INSERT INTO `modulos` VALUES (1,'Modulo Principal','192.168.1.50','MOD-01');
/*!40000 ALTER TABLE `modulos` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `pines`
--

DROP TABLE IF EXISTS `pines`;
CREATE TABLE `pines` (
  `id_pines` int(11) unsigned NOT NULL auto_increment,
  `numero` int(11) unsigned NOT NULL,
  `descripcion` varchar(50) NOT NULL,
  `id_modulo_FK` int(10) unsigned default NULL,
  PRIMARY KEY  (`id_pines`),
  KEY `Modulos` (`id_modulo_FK`),
  CONSTRAINT `Modulos` FOREIGN KEY (`id_modulo_FK`) REFERENCES `modulos` (`id_modulo`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `pines`
--
LOCK TABLES `pines` WRITE;
/*!40000 ALTER TABLE `pines` DISABLE KEYS */;
INSERT INTO `pines` VALUES (1,1,'Luz Delantera',1),(2,2,'Luz Media',1),(3,3,'Luz trasera',1),(4,4,'Sala
Lectura',1),(5,5,'Luz del Pasio',1),(6,6,'Centro de Computo',1),(7,7,'Baño',1),(8,8,'Secretaria',1);
/*!40000 ALTER TABLE `pines` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `pines_por_eventos`
--
DROP TABLE IF EXISTS `pines_por_eventos`;
CREATE TABLE `pines_por_eventos` (
  `NMID` int(10) unsigned NOT NULL auto_increment,
  `id_evento_FK` int(11) unsigned NOT NULL default '0',
  `id_pines_FK` int(11) unsigned NOT NULL default '0',
  PRIMARY KEY  (`NMID`),
  KEY `Eventos` (`id_evento_FK`),
  KEY `Pines` (`id_pines_FK`),
  CONSTRAINT `Eventos` FOREIGN KEY (`id_evento_FK`) REFERENCES `eventos` (`id_evento`) ON
DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `Pines` FOREIGN KEY (`id_pines_FK`) REFERENCES `pines` (`id_pines`) ON DELETE NO
ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
--
-- Table structure for table `pines_por_rol`
--
DROP TABLE IF EXISTS `pines_por_rol`;
CREATE TABLE `pines_por_rol` (
  `NMID` int(10) unsigned NOT NULL auto_increment,
  `id_rol_FK` int(11) unsigned NOT NULL default '0',
  `id_pines_FK` int(11) unsigned NOT NULL default '0',
  PRIMARY KEY  (`NMID`),
  KEY `Roles` (`id_rol_FK`),
  KEY `Pinesxrol` (`id_pines_FK`),
  CONSTRAINT `Pinesxrol` FOREIGN KEY (`id_pines_FK`) REFERENCES `pines` (`id_pines`) ON DELETE
CASCADE ON UPDATE CASCADE,
  CONSTRAINT `Roles` FOREIGN KEY (`id_rol_FK`) REFERENCES `roles` (`id_rol`) ON DELETE CASCADE
ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `pines_por_rol`
--

LOCK TABLES `pines_por_rol` WRITE;
/*!40000 ALTER TABLE `pines_por_rol` DISABLE KEYS */;
INSERT INTO `pines_por_rol` VALUES (8,3,1),(9,3,2),(10,2,2),(11,2,3);
/*!40000 ALTER TABLE `pines_por_rol` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `roles`
--
DROP TABLE IF EXISTS `roles`;
CREATE TABLE `roles` (
  `id_rol` int(11) unsigned NOT NULL auto_increment,
  `nombre` varchar(50) default NULL,
  `descripcion` varchar(50) default NULL,
  PRIMARY KEY  (`id_rol`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `roles`
--
LOCK TABLES `roles` WRITE;
/*!40000 ALTER TABLE `roles` DISABLE KEYS */;
INSERT INTO `roles` VALUES (2,'Rol-01','Rol Invitado'),(3,'Rol-02','Privado');
/*!40000 ALTER TABLE `roles` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `usuarios`
--
DROP TABLE IF EXISTS `usuarios`;
CREATE TABLE `usuarios` (
  `id_usuario` int(11) unsigned NOT NULL auto_increment,
  `nombre` varchar(50) NOT NULL,
  `apellido` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
```

```
  `email` varchar(50) NOT NULL,
  `cargo` varchar(20) NOT NULL,
  `id_rol_FK` int(10) unsigned default NULL,
  PRIMARY KEY  (`id_usuario`),
  KEY `Rol` (`id_rol_FK`),
  CONSTRAINT `Rol` FOREIGN KEY (`id_rol_FK`) REFERENCES `roles` (`id_rol`) ON DELETE CASCADE
ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `usuarios`
--
LOCK TABLES `usuarios` WRITE;
/*!40000 ALTER TABLE `usuarios` DISABLE KEYS */;
INSERT INTO `usuarios` VALUES (2,'Invitado','-
','DAQ4otdwBReJy6/dR/4lqdf3RYc=','invitado@invitado','',2),(5,'SACME','Administrador','3Hbp8MAAbo+RngxR
XGbbujmC94U=','admin@admin','',2);
/*!40000 ALTER TABLE `usuarios` ENABLE KEYS */;
UNLOCK TABLES;

-- Dump completed on 2007-08-29 21:01:03 DROP TABLE IF EXISTS `sacme`.`eventos`;
CREATE TABLE  `sacme`.`eventos` (
  `id_evento` int(11) unsigned NOT NULL auto_increment,
  `descripcion` varchar(50) NOT NULL,
  `horario` datetime default NULL,
  `estado` varchar(20) NOT NULL,
  `repeticion` varchar(20) NOT NULL,
  `accion` varchar(20) NOT NULL,
  `id_rol_FK` int(10) unsigned NOT NULL default '1',
  `titulo` varchar(10) NOT NULL default '',
  PRIMARY KEY  (`id_evento`),
  KEY `Ref_02` (`id_rol_FK`),
  CONSTRAINT `Ref_02` FOREIGN KEY (`id_rol_FK`) REFERENCES `roles` (`id_rol`) ON DELETE
CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `sacme`.`modulos`;
CREATE TABLE  `sacme`.`modulos` (
  `id_modulo` int(10) unsigned NOT NULL auto_increment,
  `descripcion` varchar(50) default NULL,
  `ip` varchar(50) NOT NULL,
  `nombre` varchar(50) NOT NULL default '',
  PRIMARY KEY  (`id_modulo`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `sacme`.`pines`;
CREATE TABLE  `sacme`.`pines` (
  `id_pines` int(11) unsigned NOT NULL auto_increment,
  `numero` int(11) unsigned NOT NULL,
  `descripcion` varchar(50) NOT NULL,
  `id_modulo_FK` int(11) unsigned NOT NULL default '0',
  PRIMARY KEY  (`id_pines`),
  KEY `Ref_04` (`id_modulo_FK`),
  CONSTRAINT `Ref_04` FOREIGN KEY (`id_modulo_FK`) REFERENCES `modulos` (`id_modulo`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
DROP TABLE IF EXISTS `sacme`.`pines_por_eventos`;
CREATE TABLE  `sacme`.`pines_por_eventos` (
  `NMID` int(11) NOT NULL,
  `id_evento_FK` int(11) unsigned NOT NULL default '0',
  `id_pines_FK` int(11) unsigned NOT NULL default '0',
  PRIMARY KEY  (`NMID`),
  KEY `Ref_07` (`id_evento_FK`),
  KEY `Ref_08` (`id_pines_FK`),
  CONSTRAINT `Ref_07` FOREIGN KEY (`id_evento_FK`) REFERENCES `eventos` (`id_evento`) ON DELETE
CASCADE ON UPDATE CASCADE,
  CONSTRAINT `Ref_08` FOREIGN KEY (`id_pines_FK`) REFERENCES `pines` (`id_pines`) ON DELETE
CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `sacme`.`pines_por_rol`;
CREATE TABLE  `sacme`.`pines_por_rol` (
  `NMID` int(11) NOT NULL,
  `id_rol_FK` int(11) unsigned NOT NULL default '0',
  `id_pines_FK` int(11) unsigned NOT NULL default '0',
  PRIMARY KEY  (`NMID`),
  KEY `Ref_05` (`id_rol_FK`),
  KEY `Ref_06` (`id_pines_FK`),
  CONSTRAINT `Ref_05` FOREIGN KEY (`id_rol_FK`) REFERENCES `roles` (`id_rol`) ON DELETE
CASCADE ON UPDATE CASCADE,
  CONSTRAINT `Ref_06` FOREIGN KEY (`id_pines_FK`) REFERENCES `pines` (`id_pines`) ON DELETE
CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `sacme`.`roles`;
CREATE TABLE  `sacme`.`roles` (
  `id_rol` int(11) unsigned NOT NULL auto_increment,
  `nombre` varchar(50) character set latin1 collate latin1_bin default NULL,
  `descripcion` varchar(50) character set latin1 collate latin1_bin default NULL,
  PRIMARY KEY  (`id_rol`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `sacme`.`usuarios`;
CREATE TABLE  `sacme`.`usuarios` (
  `id_usuario` int(11) unsigned NOT NULL auto_increment,
  `nombre` varchar(50) NOT NULL,
  `apellido` varchar(50) NOT NULL,
  `password` varchar(50) character set latin1 collate latin1_bin NOT NULL,
  `email` varchar(50) NOT NULL,
  `cargo` varchar(20) NOT NULL,
  `id_rol_FK` int(10) unsigned NOT NULL default '1',
  PRIMARY KEY  (`id_usuario`),
  KEY `Ref_01` (`id_rol_FK`),
  CONSTRAINT `Ref_01` FOREIGN KEY (`id_rol_FK`) REFERENCES `roles` (`id_rol`) ON DELETE
CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

## A 2.5 Firmware del Microcontrolador en el Módulo Electrónico

Este código también esta disponible en el cd adjunto a este documento.

```
;-------------------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
;Programa:  funcionprincipal.asm
;Autor:     Wilman R. V. & Gustavo G. P.
;Fecha:     17-02-2007
;Version:   1.0
;
;-------------------------------------------------------------------------------
                        List        p=16F877A           ;Tipo de procesador
                        #INCLUDE<P16F877a.INC>


#DEFINE BANCO0  BCF STATUS,RP0
#DEFINE BANCO1  BSF STATUS,RP0

;-------------------------------------------------------------------------------
;Definicion de variables globales
;-------------------------------------------------------------------------------

DIRECCION           EQU     H'20'
PUERTO              EQU     H'21'
BLOQUE              EQU     H'22'
POSICION            EQU     H'23'
AUX1                EQU     H'24'
AUX2                EQU     H'25'
AUX3                EQU     H'26'
AUX4                EQU     H'27'
AUX5                EQU     H'28'
ESTADO              EQU     H'29'
BYTE_C              EQU     H'2A'
N_BYTES             EQU     H'2B'
CRC_LSB             EQU     H'2C'
CRC_MSB             EQU     H'2D'
COUNTER1            EQU     H'2E'
AUX6                EQU     H'2F'
EXCEPT              EQU     H'30'
F_CODE              EQU     H'31'
ST_ADDRESS_LO       EQU     H'32'
ST_ADDRESS_HI       EQU     H'33'
Q_INPUTS_LO         EQU     H'34'
Q_INPUTS_HI         EQU     H'35'
F_ERROR_CODE        EQU     H'36'
EXCEPTION_C         EQU     H'37'

BYTE_1              EQU     H'38'
BYTE_2              EQU     H'39'
BYTE_3              EQU     H'3A'
BYTE_4              EQU     H'3B'
SLAVE_ADDRESS       EQU     H'3C'
SALIDA              EQU     H'50'
AUX7                EQU     H'51'
COUNT1              EQU     H'52'
COUNT2              EQU     H'53'
COUNT_L             EQU     H'54'
COUNT_H             EQU     H'55'
K                   EQU     H'56'
AUX8                EQU     H'57'
AUX9                EQU     H'58'
ALTO                EQU     H'59'
BAJO                EQU     H'5A'
AUX10       EQU     H'5B'
```

```
;-------------------------------------------------------------------------------
;Definicion de CONSTANTES
;-------------------------------------------------------------------------------

;Definen los tiempo para el reloj de 1.5 caracteres serial
TIMER1.5_H          EQU     H'F9'
TIMER1.5_L          EQU     H'42'


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

                    ORG     H'0'
                    GOTO    MAIN

;-------------------------------------------------------------------------------
;vector de interrupciones
;-------------------------------------------------------------------------------

                    ORG     H'04'
                    GOTO    MAIN

;=============================================================================
;=============================================================================
;-------------------------------------------------------------------------------
; Bloque principal del programa (MAIN)
;-------------------------------------------------------------------------------
;=============================================================================
;=============================================================================

MAIN

;-------------------------------------------------------------------------------
;------AQUI SE RELIZAN TODAS LAS CONFIGURACIONES AL MODULO
;-------------------------------------------------------------------------------

;CONFIGURACION DE PUERTOS
;-----------------------

                    BANCO1
                    MOVLW   B'10001000'
                    MOVWF   TRISB
                    MOVWF   TRISD
                    MOVWF   TRISC
                    BANCO0

;CONFIGURACION DE LA USART
;-------------------------

                    BANCO1                  ;Cambiar a Banco 1
                    BCF     TRISC,D'6'      ;RC6\TX como salida
                    BSF     TRISC,D'7'      ;RC7\RX como entrada
                    MOVLW   D'25'           ;BAUD RATE 9600
                    MOVWF   SPBRG
                    MOVLW   B'01000101'     ;7:x 6:9bits 5:disable 4:asin 3:0
                                            ;2:high 1:empty 0:0
                    MOVWF   TXSTA
                    BANCO0                  ;Cambiar a Banco 0
                    BSF     RCSTA,SPEN      ;Habilitar los pines del USART

                    BANCO1                  ;Cambiar a Banco 1
                    BSF     TXSTA,TXEN      ;Habilitar la transmision

                    BANCO0                  ;Cambiar a Banco 0
                    BSF     RCSTA,CREN      ;Habilitar la recepcion


;CONFIGURACION DE TIMER1
;-----------------------

                    CLRF    T1CON           ;1:1 PREESCALER, INTERNAL CLOCK
```

```
;CONFIGURACION DE TIMER0
;-----------------------

                BANCO1
                CLRF    OPTION_REG
                BANCO0
                CLRF    TMR0
                BANCO1
                MOVLW   B'00001000'
                MOVWF   OPTION_REG
                CLRWDT
                MOVLW   B'00001101'
                MOVWF   OPTION_REG
                BANCO0


;-------FIN DE LAS CONFIGURACIONES


;================================================================================
;--------------------------------------------------------------------------------
;Capa de enlace de datos del protocolo MODBUS
;--------------------------------------------------------------------------------
;Aqui se recibe la trama modbus y se verifica el numero del esclavo y que la trama llegue
;libre de errores, de lo contrario se descarta la trama y se continua a la espera de una
;nueva trama.
;================================================================================================

RECEPCION

                ;recibir el primer byte por el puerto serie
                CLRWDT
                BTFSS   PIR1,RCIF       ;HAY UN DATO EN EL REGISTRO RCREG?
                GOTO    RECEPCION
                CALL    RECEIVE_BYTE
                MOVWF   SLAVE_ADDRESS

                ;iniciar el contador de 1.5t
                CALL    TIMER1_1.5

                ;revisar el byte recibido si corresponde al ID del esclavo (por default "1")
                MOVLW   H'01'
                XORWF   SLAVE_ADDRESS,W
                BTFSS   STATUS,Z
                GOTO    FRAME_ERROR     ;LA PETICION NO ES PARA ESTE ESCLAVO

                ;aqui se genera un bucle para recibir los bytes restantes y guardarlos de forma consecutiva
                MOVLW   H'3C'           ;INICIO DE LA TABLA DE BYTES RECIBIDOS
                                        ; 3CH ES LA DIRECCION DE
                MOVWF   FSR             ;SLAVE_ADDRESS
                INCF    FSR,F

                MOVLW   H'01'
                MOVWF   N_BYTES         ;CONTADOR DE BYTES RECIBIDOS


ESPERAR         BTFSC   PIR1,RCIF       ;REVISAR SI HA LLEGADO UN NUEVO BYTE
                GOTO    NUEVO_BYTE      ;
                BTFSS   PIR1,TMR1IF     ;REVISAR SI SE HA DESBORDADO EL TIMER
                GOTO    ESPERAR         ;
                GOTO    FIN_TRAMA       ;


NUEVO_BYTE
                BCF     T1CON,TMR1ON    ;DETENER EL TIMER
                CALL    RECEIVE_BYTE
                MOVWF   INDF

                CALL    TIMER1_1.5
```

```
                INCF    N_BYTES,F
                INCF    FSR,F
                GOTO    ESPERAR



FIN_TRAMA
                BCF     T1CON,TMR1ON    ;DETENER EL TIMER
                MOVLW   H'8'            ;SE VERIFICA SI SE RECIBIO MAS DE 8 BYTES,
                                        ;DE CONTRARIO HAY UN
                SUBWF   N_BYTES,W       ;ERROR DE TRAMA.
                BTFSS   STATUS,C
                GOTO    FRAME_ERROR


                ;inicia la verificacion del CRC de los datos recibidos
                MOVLW   H'3C'           ;INDICA EL INICIO DEL BLOQUE DE BYTES
                                        ;RECIBIDOS
                MOVWF   FSR
                DECF    N_BYTES,F
                DECF    N_BYTES,F


                ;Limpia variables para calculo del CRC
                MOVLW   H'FF'
                MOVWF   CRC_LSB
                MOVWF   CRC_MSB



CALCULAR_CRC
                CLRWDT
                MOVF    INDF,W          ;SE CALCULA EN CRC PARA LOS BYTES RECIBIDOS
                CALL    GENERATE_CRC    ;
                INCF    FSR,F           ;
                DECFSZ  N_BYTES,F       ;
                GOTO    CALCULAR_CRC    ;

                MOVF    INDF,W          ;
                XORWF   CRC_LSB,W       ;
                BTFSS   STATUS,Z        ;
                GOTO    FRAME_ERROR     ;SE VERIFICA EL CRC RECIBIDO CON EL CALCULADO
                INCF    FSR,F           ;SI NO SON IGUALES HAY UN ERROR DE TRAMA
                MOVF    INDF,W          ;
                XORWF   CRC_MSB,W       ;
                BTFSC   STATUS,Z
                GOTO    CAPA_APLICAC


FRAME_ERROR
                BCF     T1CON,TMR1ON    ;DETENER EL TIMER

                GOTO    RECEPCION

                ;aqui se calcula el CRC de la trama a enviar y se envia la trama via serie
ENVIAR_RESP
                CLRWDT
                ;Limpia variables para calculo del CRC
                MOVLW   H'FF'
                MOVWF   CRC_LSB
                MOVWF   CRC_MSB

                MOVF    N_BYTES,W       ;
                MOVWF   AUX5            ;GUARDAR UNA COPIA DE N_BYTES PARA USARLO EN EL ENVIO
                INCF    AUX5,F          ;
                INCF    AUX5,F

                MOVLW   H'3C'           ;INDICA EL INICIO DEL BLOQUE DE BYTES RECIBIDOS
                MOVWF   FSR
```

```
CALC_CRC
                CLRWDT
                MOVF    INDF,W          ;SE CALCULA EN CRC PARA LOS BYTES RECIBIDOS
                CALL    GENERATE_CRC    ;
                INCF    FSR,F           ;
                DECFSZ  N_BYTES,F       ;
                GOTO    CALC_CRC        ;

                MOVF    CRC_LSB,W       ;
                MOVWF   INDF            ;ANEXAR EL CRC CALCULADO A LA TRAMA QUE SE ENVIA
                INCF    FSR,F           ;
                MOVF    CRC_MSB,W       ;
                MOVWF   INDF


                MOVLW   H'3C'           ;INDICA EL INICIO DEL BLOQUE DE BYTES RECIBIDOS
                MOVWF   FSR

ENVIAR          MOVF    INDF,W          ;
                CALL    SEND_BYTE       ;ENVIA BYTE POR BYTE LA TRAMA DE RESPUESTA
                INCF    FSR,F           ;
                DECFSZ  AUX5,F          ;
                GOTO    ENVIAR

FIN_ENVIAR
                GOTO    RECEPCION

;=============================================================================
;-----------------------------------------------------------------------------
;Capa de aplicacion del protocolo MODBUS
;-----------------------------------------------------------------------------
;en esta capa se ejecuta las funciones del modulo, y se revisa si los datos como codigo de
;funcion, direcciones y cantidad de salidas o entrdas estan soportadas por el modulo
;ademas se prepara las tramas que seran enviadas como respuesta.
;=============================================================================

CAPA_APLICAC

;-----------------------------------------------------------------------------
;Validacion de codigos de funciones,datos y direcciones
;-----------------------------------------------------------------------------
                CLRWDT
                MOVLW   H'3C'           ;DIRECCION DE SLAVE_ADDRESS PRIMER BYTE
                MOVWF   FSR
                INCF    FSR,F

                MOVF    INDF,W          ;OBTENER EL CODIGO DE LA FUNCION
                MOVWF   F_CODE


                ;verificar el codigo de la funcion, de lo contrario generar la excepcion corresp.
                SUBLW   H'02'
                BTFSC   STATUS,Z        ;VERIFICAR SI ES FUNCION 2
                GOTO    FUNCTION_02     ;DE LO CONTRARIO PASAR A LA SIGUIENTE FUNCION

                MOVF    F_CODE,W
                SUBLW   H'05'
                BTFSC   STATUS,Z        ;VERIFICAR SI ES FUNCION 5
                GOTO    FUNCTION_05     ;DE LO CONTRARIO PASAR A LA SIGUIENTE FUNCION

NO_FUNCION
                MOVF    F_CODE,W        ;NINGUNA DE LAS FUNCIONES SOPORTADAS, SE GENERA
                ADDLW   H'80'           ;EL CODIGO DE ERROR CONRRESPONDIENTE
                MOVWF   F_ERROR_CODE    ;

                MOVLW   H'01'
                MOVWF   EXCEPTION_C

                GOTO    MAKE_ERR_RESP
```

```
FUNCTION_02

                    CALL    VERIFY              ;LLAMAR LA RUTINA DE VERIFICACION DE DATOS

                    ;es hora de procesar la funcion y obtener el estado de las entradas solicitado
                    MOVLW   H'0'
                    XORWF   EXCEPT,W
                    BTFSC   STATUS,Z
                    GOTO    FUNCION02

                    MOVLW   H'02'
                    XORWF   EXCEPT,W
                    BTFSC   STATUS,Z
                    GOTO    EXCEP_02
                    GOTO    EXCEP_03


CREAR_TRAMA_02
                    CLRWDT
                    MOVLW   H'3C'               ;INDICA EL INICIO DEL BLOQUE DE BYTES RECIBIDOS (SlAVE_ADDRESS)
                    MOVWF   FSR

                    INCF    FSR,F               ;
                    MOVLW   H'02'               ;AGREGAR CODIGO DE FUNCION A LA TRAMA
                    MOVWF   INDF

                    INCF    FSR,F               ;
                    MOVF    BYTE_C,W            ;AGREGAR CONTADOR DE BYTES A LA TRAMA
                    MOVWF   INDF

                    INCF    FSR,F               ;
                    MOVF    BYTE_1,W            ;
                    MOVWF   INDF                ;AGREGAR LOS BYTES DE DATOS A LA TRAMA
                    INCF    FSR,F               ;
                    MOVF    BYTE_2,W            ;
                    MOVWF   INDF                ;
                    INCF    FSR,F               ;
                    MOVF    BYTE_3,W            ;
                    MOVWF   INDF                ;
                    INCF    FSR,F               ;
                    MOVF    BYTE_4,W            ;
                    MOVWF   INDF

                    MOVLW   H'03'
                    ADDWF   BYTE_C,W
                    MOVWF   N_BYTES

                    GOTO    ENVIAR_RESP         ;IR A LA CAPA DE ENLACE PARA ENVIAR LA TRAMA


FUNCTION_05

                    CLRWDT
                    INCF    FSR,F               ;OBTENER LA DIRECCION INICIAL PARTE ALTA
                    MOVF    INDF,W              ;
                    MOVWF   ST_ADDRESS_HI       ;

                    INCF    FSR,F               ;OBTENER LA DIRECCION INICIAL PARTE BAJA
                    MOVF    INDF,W              ;
                    MOVWF   ST_ADDRESS_LO       ;

                    INCF    FSR,F               ;OBTENER EL VALOR DE LA SALIDA PARTE ALTA
                    MOVF    INDF,W              ;
                    MOVWF   Q_INPUTS_HI         ;

                    INCF    FSR,F               ;OBTENER EL VALOR DE LA SALIDA PARTE BAJA
                    MOVF    INDF,W              ;
                    MOVWF   Q_INPUTS_LO         ;
```

```
                    ;verificar que el valor de salida sea 0000 ó ff00
                    MOVLW  H'0'                 ;
                    XORWF  Q_INPUTS_LO,W        ;LA PARTE BAJA DEBE SER SIEMPRE CERO
                    BTFSS  STATUS,Z             ;
                    GOTO   EXCEP_03


                    MOVLW  H'0'                 ;
                    XORWF  Q_INPUTS_HI,W        ;LA PARTE ALTA PUEDE SER CERO
                    BTFSC  STATUS,Z             ;
                    GOTO   Q_OK


                    MOVLW  H'FF'                ;
                    XORWF  Q_INPUTS_HI,W        ;LA PARTE ALTA PUEDE SER FF
                    BTFSC  STATUS,Z             ;
                    GOTO   Q_OK
                    GOTO   EXCEP_03

Q_OK


                    ;verificar que la direccion de inicio sea correcta I < 160

                    MOVLW  H'0'                 ;
                    XORWF  ST_ADDRESS_HI,W;LA PARTE ALTA DE LA DIRECCION DE INICIO DEBE SER CERO
                    BTFSS  STATUS,Z             ;
                    GOTO   EXCEP_02

                    MOVLW  D'160'               ;
                    SUBWF  ST_ADDRESS_LO,W          ;LA DIRECCION DE LA SALIDA DEBE SER MENOR QUE 160
                    BTFSC  STATUS,C             ;
                    GOTO   EXCEP_02


                    ;listo para ir a poner a uno o a cero la salida especificada

                    GOTO   FUNCION05

CREAR_TRAMA_05
                    CLRWDT
                    MOVLW  H'06'
                    MOVWF  N_BYTES

                    GOTO   ENVIAR_RESP      ;IR A LA CAPA DE ENLACE PARA ENVIAR LA TRAMA


;--------------------------------------EXCEPCIONES---------------------------------------
EXCEP_02

                    MOVF   F_CODE,W
                    ADDLW  H'80'
                    MOVWF  F_ERROR_CODE

                    MOVLW  H'02'
                    MOVWF  EXCEPTION_C

                    GOTO   MAKE_ERR_RESP


EXCEP_03

                    MOVF   F_CODE,W
                    ADDLW  H'80'
                    MOVWF  F_ERROR_CODE

                    MOVLW  H'03'
                    MOVWF  EXCEPTION_C

                    GOTO   MAKE_ERR_RESP
```

```
EXCEP_04

                    MOVF    F_CODE,W
                    ADDLW   H'80'
                    MOVWF   F_ERROR_CODE

                    MOVLW   H'04'
                    MOVWF   EXCEPTION_C

                    GOTO    MAKE_ERR_RESP

                    ;se crea la trama de respuesta cuando se genera una excepcion

MAKE_ERR_RESP
                    CLRWDT
                    MOVLW   H'3C'               ;INDICA EL INICIO DEL BLOQUE DE BYTES RECIBIDOS (SlAVE_ADDRESS)
                    MOVWF   FSR

                    INCF    FSR,F
                    MOVF    F_ERROR_CODE,W
                    MOVWF   INDF

                    INCF    FSR,F
                    MOVF    EXCEPTION_C,W
                    MOVWF   INDF

                    MOVLW   H'03'
                    MOVWF   N_BYTES
                    GOTO    ENVIAR_RESP     ;IR A LA CAPA DE ENLACE A ENVIAR TRAMA DE RESPUESTA



;-------------------------------------FUNCION 02-----------------------------------------
FUNCION02

                    MOVF    ST_ADDRESS_LO,W         ;COPIAR LA DIRECCION AL REGISTRO DE TRABAJO
                    MOVWF   DIRECCION       ;

                    CALL    DIR_RESOL       ;RESOLVER LA DIRECCION
                    CALL    ORDENAR_BITS    ;

                    BSF     POSICION,6      ;PONER EL SEXTO BIT A 1, (CUANDO SE DIRECCIONA ENTRADAS)


                    MOVLW   H'1'
                    MOVWF   BYTE_C

                    MOVLW   D'8'                ;DIVIDE EL NUMERO DE ENTRADAS EN BLOQUES DE 8
DIVIDE              SUBWF   Q_INPUTS_LO,F       ;PARA SABER EN CUANTOS BYTES CABE LA RESPUESTA
                    BTFSC   STATUS,Z            ;(UNA ENTRADA POR BIT)
                    GOTO    CONTINUAR

                    INCF    BYTE_C,F
                    GOTO    DIVIDE

CONTINUAR



;accesar a las entradas de un bus de forma consecutiva

LEERBUS
                    CLRWDT
                    MOVLW   H'0'
                    MOVWF   AUX4

                    BTFSC   BLOQUE,0
                    GOTO    BLK1
```

```
                ;lectura de las entradas en el bloque 0
BLK0
                MOVLW  D'8'             ;INICIAR CONTADOR A 8 VUELTAS
                MOVWF  AUX3

                CALL   SERIAL_DATA
                BSF    INDF,2           ;ACTIVAR EL DEMULTIPLEXOR
                CALL   DELAY_125
                CLRF   ESTADO

R_MOD           BTFSS  INDF,3           ;LEER EL ESTADO ACTUAL DE LA ENTRADA
                BCF    ESTADO,7         ;ESPECIFICADA
                BTFSC  INDF,3           ;
                BSF    ESTADO,7

                                        ;CADA ESTADO SE ALMACENA EN UN BIT DEL REGISTRO ESTADO,
                DECFSZ AUX3,F           ;CUANDO SE HA LLENADO EL REGISTRO SALTA A REINICIAR
                GOTO   SEGUIR           ;PARA GUARDAR DICHO REGISTRO Y REINICIARLO PARA
                GOTO   REINICIAR        ;LEER OTROS OCHO ESTADOS
SEGUIR
                RRF    ESTADO,F         ;UBICAR CADA LECTURA COMO EL MSB
                INCF   POSICION,F       ;
                CALL   SERIAL_DATA      ;
                GOTO   R_MOD

REINICIAR

                MOVLW  H'38'            ;DIRECCION DE BYTE_1
                ADDWF  AUX4,W  ;SUMA EL VALOR DE AUX4(CONTADOR DE BYTES) A LA DIRECCION INICIAL
                MOVWF  FSR              ;Y ASI POSICIONAR A FSR EN EL REGISTRO SUPERIOR CONSECUTIVO

                MOVF   ESTADO,W
                MOVWF  INDF

                INCF   AUX4,F
                MOVF   AUX4,W
                XORWF  BYTE_C,W
                BTFSS  STATUS,Z
                GOTO   DISTINTO
                GOTO   IGUAL


DISTINTO
                MOVLW  B'11110000'      ;LIMPIA LOS TRES LSB BITS PARA COMENZAR UNA NUEVA CUENTA DE
OCHO
                ANDWF  POSICION,F       ;ENTRADAS

                MOVLW  B'00010000'      ;INICIALIZA EL PUNTERO DE ENTRADAS AL SIGUIENTE BLOQUE
                ADDWF  POSICION,F       ;O MODULO DE EXPANSION
                GOTO   BLK0

IGUAL
                MOVF   PUERTO,W
                MOVWF  FSR
                BCF    INDF,2           ;DESACTIVAR DEMULTIPLEXOR
                GOTO   CREAR_RESP



                ;lectura de las entradas en el bloque 1
BLK1

                MOVLW  D'8'             ;INICIAR CONTADOR A 8 VUELTAS
                MOVWF  AUX3

                CALL   SERIAL_DATA
                BSF    INDF,6           ;ACTIVAR EL DEMULTIPLEXOR
                CALL   DELAY_125
                CLRF   ESTADO
```

```
R_MOD2          BTFSS   INDF,7          ;LEER EL ESTADO ACTUAL DE LA ENTRADA
                BCF     ESTADO,7        ;ESPECIFICADA
                BTFSC   INDF,7          ;
                BSF     ESTADO,7


                DECFSZ  AUX3,F
                GOTO    SEGUIR2
                GOTO    REINICIAR2
SEGUIR2
                RRF     ESTADO,F        ;UBICAR CADA LECTURA COMO EL MSB
                INCF    POSICION,F      ;
                CALL    SERIAL_DATA     ;
                GOTO    R_MOD2

REINICIAR2

                MOVLW   H'38'           ;DIRECCION DE BYTE_1
                ADDWF   AUX4,W
                MOVWF   FSR

                MOVF    ESTADO,W
                MOVWF   INDF

                INCF    AUX4,F
                MOVF    AUX4,W
                XORWF   BYTE_C,W
                BTFSS   STATUS,Z
                GOTO    DISTINTO2
                GOTO    IGUAL2


DISTINTO2
                MOVLW   B'11110000'
                ANDWF   POSICION,F

                MOVLW   B'00010000'
                ADDWF   POSICION,F
                GOTO    BLK1


IGUAL2
                MOVF    PUERTO,W
                MOVWF   FSR
                BCF     INDF,6          ;DESACTIVAR DEMULTIPLEXOR


CREAR_RESP

                GOTO    CREAR_TRAMA_02


;-------------------------------------FUNCION 05----------------------------------------
FUNCION05
                CLRWDT
                MOVF    ST_ADDRESS_LO,W         ;COPIAR LA DIRECCION AL REGISTRO DE TRABAJO
                MOVWF   DIRECCION       ;

                CALL    DIR_RESOL       ;RESOLVER LA DIRECCION
                CALL    ORDENAR_BITS    ;

                MOVLW   H'3'            ;
                MOVWF   AUX6            ;CONTADOR DE VUELTAS

                CLRF    ESTADO

                BTFSC   BLOQUE,0
                GOTO    BL1
```

```
BL0

                ;obtener el estado actual de la salida
                BSF     POSICION,6      ;PONER EL SEXTO BIT A 1, (CUANDO SE DIRECCIONA ENTRADAS)
                CALL    SERIAL_DATA


                BSF     INDF,2          ;ACTIVAR EL DEMULTIPLEXOR
                CALL    DELAY_125

                MOVLW   D'20'
                MOVWF   AUX9

                CLRF    ALTO
                CLRF    BAJO

CONTEO

                BTFSS   INDF,3
                INCF    BAJO,F
                BTFSC   INDF,3
                INCF    ALTO,F

                CALL    DELAY_125

                DECFSZ  AUX9,F
                GOTO    CONTEO


                BCF     INDF,2          ;APAGAR EL DEMULTIPLEXOR


                MOVF    ALTO,W
                SUBWF   BAJO,W
                BTFSC   STATUS,C
                BCF     ESTADO,0
                BTFSS   STATUS,C
                BSF     ESTADO,0


                BTFSS   Q_INPUTS_HI,0
                GOTO    LOW_E
                BTFSC   Q_INPUTS_HI,0
                GOTO    HI_E


LOW_E
                BTFSC   ESTADO,0
                GOTO    CAMBIAR_E
                BTFSS   ESTADO,0
                GOTO    HECHO
                GOTO    CAMBIAR_E

HI_E
                BTFSS   ESTADO,0
                GOTO    CAMBIAR_E
                BTFSC   ESTADO,0
                GOTO    HECHO
                GOTO    CAMBIAR_E


CAMBIAR_E
                BTFSS   AUX6,1
                GOTO    ERROR_04
                BTFSS   AUX6,0
                BSF     POSICION,3


                BCF     POSICION,6
```

```
                CALL    SERIAL_DATA
                CALL    SERIAL_DATA

                BSF     INDF,2          ;ACTIVAR MULTIPLEXOR

                CALL    DELAY_5MS
                CALL    DELAY_5MS
                CALL    DELAY_5MS
                CALL    DELAY_5MS


                BCF     INDF,2          ;APAGAR MULTIPLEXOR

                BCF     POSICION,3

                MOVLW   D'5'
                MOVWF   AUX8

DELAY_50
                CALL    DELAY_5MS

                DECFSZ  AUX8,F
                GOTO    DELAY_50


                DECFSZ  AUX6,F
                GOTO    BL0


BL1

                ;obtener el estado actual de la salida
                BSF     POSICION,6      ;PONER EL SEXTO BIT A 1, (CUANDO SE DIRECCIONA ENTRADAS)
                CALL    SERIAL_DATA


                BSF     INDF,6          ;ACTIVAR EL DEMULTIPLEXOR
                CALL    DELAY_125


                MOVLW   D'20'
                MOVWF   AUX9

                CLRF    ALTO
                CLRF    BAJO

CONTEO1

                BTFSS   INDF,7
                INCF    BAJO,F
                BTFSC   INDF,7
                INCF    ALTO,F

                CALL    DELAY_125

                DECFSZ  AUX9,F
                GOTO    CONTEO1


                BCF     INDF,6          ;APAGAR EL DEMULTIPLEXOR


                MOVF    ALTO,W
                SUBWF   BAJO,W
                BTFSC   STATUS,C
                BCF     ESTADO,0
                BTFSS   STATUS,C
                BSF     ESTADO,0
```

```
                BTFSS   Q_INPUTS_HI,0
                GOTO    LOW_E1
                BTFSC   Q_INPUTS_HI,0
                GOTO    HI_E1


LOW_E1
                BTFSC   ESTADO,0
                GOTO    CAMBIAR_E1
                BTFSS   ESTADO,0
                GOTO    HECHO
                GOTO    CAMBIAR_E1

HI_E1
                BTFSS   ESTADO,0
                GOTO    CAMBIAR_E1
                BTFSC   ESTADO,0
                GOTO    HECHO
                GOTO    CAMBIAR_E1


CAMBIAR_E1
                BTFSS   AUX6,1
                GOTO    ERROR_04
                BTFSS   AUX6,0
                BSF     POSICION,3


                BCF     POSICION,6
                CALL    SERIAL_DATA
                CALL    SERIAL_DATA

                BSF     INDF,6          ;ACTIVAR MULTIPLEXOR

                CALL    DELAY_5MS
                CALL    DELAY_5MS
                CALL    DELAY_5MS
                CALL    DELAY_5MS


                BCF     INDF,6          ;APAGAR MULTIPLEXOR

                BCF     POSICION,3

                MOVLW   D'5'
                MOVWF   AUX8

DELAY_501
                CALL    DELAY_5MS

                DECFSZ  AUX8,F
                GOTO    DELAY_501


                DECFSZ  AUX6,F
                GOTO    BL1

ERROR_04

                CLRWDT
                GOTO    EXCEP_04

HECHO

                CLRWDT
                GOTO    CREAR_TRAMA_05
```

```
#INCLUDE<funciones_basicas.inc>
                        END
```

## funciones_basicas.inc

```
;===============================================================================================
;-----------------------------------------------------------------------------
;DEFINICION DE FUNCIONES A IMPLEMENTAR POR EL MODULO CENTRAL MODELO: SMARTSWITCH256
;-----------------------------------------------------------------------------
;===============================================================================================


;-----------------------------------------------------------------------------
;FUNCION DE RESOLUCION DE DIRECCIONES: DIR_RESOL
;
;PARAMETROS:    RECIBE UN NUMERO QUE CORRESPONDE A UNA SALIDA O ENTRADA DE LAS 160 DISPONIBLES
;               EN EL SISTEMA, DICHO VALOR ESTA EN EL RANGO DE 0 - 159 YA SEA PARA ENTRADAS
;               O SALIDAS, Y SE LEE EN UN REGISTRO LLAMADO DIRECCION.
;
;RETORNA:       DEVUELVE TRES REGISTROS CON LOS VALORES CORRESPONDIENTES A EL PUERTO, EL BLOQUE
;               DENTRO DE DICHO PUERTO, Y LA POSICION QUE OCUPA EN DICHO BLOQUE.
;
;
;               DIRECCION, PUERTO, BLOQUE, POSICION

;-----------------------------------------------------------------------------
DIR_RESOL

                CLRWDT
;------LA SALIDA/ENTRADA SE ENCUENTRA ENTRE 0 Y 31

BUS1            MOVF    DIRECCION,W
                SUBLW   D'31'
                BTFSS   STATUS,C        ;VERIFICAR SI DIRECCION ES MENOR QUE 32
                GOTO    BUS2            ;DE LO CONTRARIO PASAR AL SIGUIENTE BLOQUE

                MOVLW   H'6'            ;PUERTO B
                MOVWF   PUERTO
                BCF     BLOQUE,0        ;BLOQUE '0'
                MOVF    DIRECCION,W     ;POSICION ENTRE 0-31
                MOVWF   POSICION
                RETURN


;------LA SALIDA/ENTRADA SE ENCUENTRA ENTRE 32 Y 63

BUS2            MOVF    DIRECCION,W
                SUBLW   D'63'
                BTFSS   STATUS,C
                GOTO    BUS3

                MOVLW   H'6'            ;PUERTO B
                MOVWF   PUERTO
                BSF     BLOQUE,0        ;BLOQUE '1'
                MOVLW   D'32'
                SUBWF   DIRECCION,W     ;POSICION ENTRE 0-31
                MOVWF   POSICION
                RETURN
```

```
;------LA SALIDA/ENTRADA SE ENCUENTRA ENTRE 64 Y 95

BUS3            MOVF    DIRECCION,W
                SUBLW   D'95'
                BTFSS   STATUS,C
                GOTO    BUS4

                MOVLW   H'8'            ;PUERTO D
                MOVWF   PUERTO
                BCF     BLOQUE,0        ;BLOQUE '0'
                MOVLW   D'64'
                SUBWF   DIRECCION,W     ;POSICION ENTRE 0-31
                MOVWF   POSICION
                RETURN


;------LA SALIDA/ENTRADA SE ENCUENTRA ENTRE 96 Y 127

BUS4            MOVF    DIRECCION,W
                SUBLW   D'127'
                BTFSS   STATUS,C
                GOTO    BUS5

                MOVLW   H'8'            ;PUERTO D
                MOVWF   PUERTO
                BSF     BLOQUE,0        ;BLOQUE '1'
                MOVLW   D'96'
                SUBWF   DIRECCION,W     ;POSICION ENTRE 0-31
                MOVWF   POSICION
                RETURN


;------LA SALIDA/ENTRADA SE ENCUENTRA ENTRE 128 Y 159

BUS5            MOVLW   H'7'            ;PUERTO C
                MOVWF   PUERTO
                BCF     BLOQUE,0        ;BLOQUE '0'
                MOVLW   D'128'
                SUBWF   DIRECCION,W     ;POSICION ENTRE 0-31
                MOVWF   POSICION
                RETURN



;-------------------------------------------------------------------------------------
;FUNCION PARA SERIALIZAR Y ENVIAR DATOS AL MODULO SECUNDARIO: SERIAL_DATA
;
;parametros:        Recibe la POSICION, PUERTO Y BLOQUE a la que enviara un byte de forma serial.
;retorna:   No retorna ningun valor.
;
;               Esta funcion envia un byte de forma serial a un bus especificado por los registros
;               PUERTO, POSICION Y BLOQUE, luego retorna.
;-------------------------------------------------------------------------------------

SERIAL_DATA

                CLRWDT
                MOVLW   H'8'            ;
                MOVWF   AUX1            ;CONTADOR DE ROTACIONES DE 8 BITS

                MOVF    POSICION,W
                MOVWF   AUX2


PUERTOB
                MOVF    PUERTO,W
                SUBLW   H'6'
                BTFSS   STATUS,Z        ;VERIFICAR SI ES PUERTO B
                GOTO    PUERTOC                 ;DE LO CONTRARIO PASAR AL SIGUIENTE PUERTO
```

```
                MOVLW   H'6'            ;INICIALIZAR EL FSR EN EL PUERTO B
                MOVWF   FSR             ;
                GOTO    BLOQUES         ;

PUERTOC
                MOVF    PUERTO,W
                SUBLW   H'7'
                BTFSS   STATUS,Z        ;VERIFICAR SI ES PUERTO C
                GOTO    PUERTOD                 ;DE LO CONTRARIO PASAR AL SIGUIENTE

                MOVLW   H'7'            ;INICIALIZAR EL FSR EN EL PUERTO C
                MOVWF   FSR             ;
                GOTO    BLOQUES         ;

PUERTOD
                MOVF    PUERTO,W
                SUBLW   H'8'
                BTFSS   STATUS,Z        ;VERIFICAR SI ES PUERTO D
                RETURN                  ;SI NINGUNO DE LOS PUERTOS ES ELEGIDO REGRESAR

                MOVLW   H'8'            ;INICIALIZAR EL FSR EN EL PUERTO D
                MOVWF   FSR             ;


BLOQUES
                BTFSS   BLOQUE,0        ;Verificar el bloque en el que se encuentra la direccion
                GOTO    BLOQUE0         ;
                BTFSC   BLOQUE,0        ;
                GOTO    BLOQUE1         ;


BLOQUE0
                RLF     AUX2,F          ;ENVIAR "MSB" PRIMERO

                BTFSS   STATUS,C        ;ES UN UNO?
                BCF     INDF,0          ;NO, ES UN CERO
                BTFSC   STATUS,C        ;ES UN CERO?
                BSF     INDF,0          ;NO, ES UN UNO

                BSF     INDF,1          ;ENVIAR PULSO DE RELOJ
                NOP
                NOP
                NOP
                BCF     INDF,1          ;

                DECFSZ  AUX1,F
                GOTO    BLOQUE0

                RETURN


BLOQUE1
                RLF     AUX2,F          ;ENVIAR "MSB" PRIMERO

                BTFSS   STATUS,C        ;ES UN UNO?
                BCF     INDF,4          ;NO, ES UN CERO
                BTFSC   STATUS,C        ;ES UN CERO?
                BSF     INDF,4          ;NO, ES UN UNO

                BSF     INDF,5          ;ENVIAR PULSO DE RELOJ
                NOP
                NOP
                NOP
                BCF     INDF,5          ;

                DECFSZ  AUX1,F
                GOTO    BLOQUE1

                RETURN
```

```
;-------------------------------------------------------------------------------
;FUNCION PARA ORDENAR EL BYTE QUE SERA ENVIADO: ORDENAR_BITS
;
;parametros:            Recibe el registro POSICION.
;retorna:    el registro POSICION modificado
;
;                       Esta funcion corre los bits 3 y 4 una posicion a la izquierda para permitir
;                       la correcta multiplexion en los modulos secundarios
;
;                       antes:   00011111
;                       despues: 00110111
;
;-------------------------------------------------------------------------------



ORDENAR_BITS

                        BTFSS   POSICION,4          ;ES UN UNO?
                        BCF     POSICION,5          ;NO, ES UN CERO
                        BTFSC   POSICION,4          ;ES UN CERO?
                        BSF     POSICION,5          ;NO, ES UN UNO

                        BTFSS   POSICION,3          ;ES UN UNO?
                        BCF     POSICION,4          ;NO, ES UN CERO
                        BTFSC   POSICION,3          ;ES UN CERO?
                        BSF     POSICION,4          ;NO, ES UN UNO

                        BCF     POSICION,3

                        RETURN


;-------------------------------------------------------------------------------
;NOMBRE:                        Timer1_1.5
                                        *           *
;DESCRIPCIÓN:    Crea un retardo de 2.5 caracteres serial
;                                       a 9600 baudios
;-------------------------------------------------------------------------------

TIMER1_1.5
                        CLRWDT
                        BCF     PIR1,TMR1IF         ;BANDERA QUE INDICA DESBORDAMIENTO DEL TIMER
                        MOVLW   TIMER1.5_L          ;Se carga el registro del TMR1 para 1.5 caracteres
                        MOVWF   TMR1L
                        MOVLW   TIMER1.5_H
                        MOVWF   TMR1H
                        BSF     T1CON,TMR1ON        ;ACTIVAR TIMER1
                        RETURN


;---------------------------------------------------------------
; USARTTX: Subrutina de envio del contenido del registro W por medio
;               de USART
;---------------------------------------------------------------

SEND_BYTE
USARTTX_UNTIL

                        BTFSS   PIR1,TXIF                   ;ESTA VACIO EL REGISTRO TXREG?
                        GOTO    USARTTX_UNTIL
                        MOVWF   TXREG

                        RETURN
```

```
;-----------------------------------------------------------------
; USARTRX: Subrutina de recibo de datos por medio de USART
;                 El byte recibido lo deja en el registro W
;-----------------------------------------------------------------

RECEIVE_BYTE

                        MOVF    RCREG,W

                        RETURN


;-----------------------------------------------------------------
;NOMBRE:                        GENERATE_CRC
;DESCRIPCIÓN:    Calcula el CRC          para el medio serial
;PARAMETROS:     CRC_LSB, CRC_MSB, COUNTER1
;-----------------------------------------------------------------

GENERATE_CRC
                        ;Se aplica Xor con el primer byte de dato y el byte
                        ;menos significativo del CRC, el resultado se almacena
                        ;el CRC_LSB

                        XORWF   CRC_LSB,F

                        ;Se inicializa el condator de lazos
                        MOVLW   H'00'
                        MOVWF   COUNTER1

START_SHIFT

                        MOVF    COUNTER1,W
                        SUBLW   H'08'
                        BTFSC   STATUS,Z
                        GOTO    EXIT_SHIFT_BYTE

                        ;se incrementa el contador de desplazamientos hacia la derecha
                        INCF    COUNTER1,F

                        ;se establece el bit c de status a cero (para el desplazamiento)
                        BCF     STATUS,C

                        ;desplazamiento del byte mas significativo del crc
                        RRF     CRC_MSB,F

                        ;el bit menos significativo de crc_msb ha quedado almacenado
                        ;en el bit c del registro status

                        ;desplazamiento del byte menos significativo del crc
                        RRF     CRC_LSB,F

                        ;ahora el bit menos significativo de crc_lsb ha quedado en
                        ;el bit c del registro status, este bit nos servira para
                        ;el siguiente proceso

                        BTFSS   STATUS,C
                        GOTO    START_SHIFT

                        ;se hara un xor con la palabra 0xa001
                        MOVLW   H'01'
                        XORWF   CRC_LSB,F

                        MOVLW   H'A0'
                        XORWF   CRC_MSB,F

                        GOTO    START_SHIFT

EXIT_SHIFT_BYTE
                        CLRWDT
                        RETURN
```

```
;-------------------------------------------------------------
;NOMBRE:                 VERIFY
;DESCRIPCIÓN:    VERIFICA LOS DATOS RECIBIDOS DE CANTIDAD DE ENTRADAS/SALIDAS Y DIRECCIONES
;PARAMETROS:     NINGUNO
;-------------------------------------------------------------


VERIFY
                MOVLW  H'3C'            ;DIRECCION DE SLAVE_ADDRESS PRIMER BYTE
                MOVWF  FSR
                INCF   FSR,F

                CLRF   EXCEPT

                INCF   FSR,F            ;OBTENER LA DIRECCION INICIAL PARTE ALTA
                MOVF   INDF,W           ;
                MOVWF  ST_ADDRESS_HI    ;

                INCF   FSR,F            ;OBTENER LA DIRECCION INICIAL PARTE BAJA
                MOVF   INDF,W           ;
                MOVWF  ST_ADDRESS_LO    ;

                INCF   FSR,F            ;OBTENER LA CANTIDAD DE ENTRADAS PARTE ALTA
                MOVF   INDF,W           ;
                MOVWF  Q_INPUTS_HI      ;

                INCF   FSR,F            ;OBTENER LA CANTIDAD DE ENTRADAS PARTE BAJA
                MOVF   INDF,W           ;
                MOVWF  Q_INPUTS_LO      ;

                ;verificar los limites de la cantidad de entradas a leer 8 <= Q <= 32
                MOVLW  H'0'             ;
                XORWF  Q_INPUTS_HI,W    ;LA PARTE ALTA DE LA CANTIDAD DE ENTRADAS DEBE SER CERO
                BTFSS  STATUS,Z         ;
                GOTO   EXCEPT_03


                MOVLW  D'8'
                XORWF  Q_INPUTS_LO,W
                BTFSC  STATUS,Z
                GOTO   Q_GOOD

                MOVLW  D'16'
                XORWF  Q_INPUTS_LO,W
                BTFSC  STATUS,Z
                GOTO   Q_GOOD

                MOVLW  D'24'
                XORWF  Q_INPUTS_LO,W
                BTFSC  STATUS,Z
                GOTO   Q_GOOD

                MOVLW  D'32'
                XORWF  Q_INPUTS_LO,W
                BTFSC  STATUS,Z
                GOTO   Q_GOOD

                GOTO   EXCEPT_03

Q_GOOD

                ;verificar que la direccion de inicio sea correcta I+Q <= 32

                MOVLW  H'0'             ;
                XORWF  ST_ADDRESS_HI,W;LA PARTE ALTA DE LA DIRECCION DE INICIO DEBE SER CERO
                BTFSS  STATUS,Z         ;
                GOTO   EXCEPT_02
```

```
        MOVLW   D'0'
                MOVWF   AUX10
DIV8            MOVF    AUX10,W
        XORWF   ST_ADDRESS_LO,W
        BTFSC   STATUS,Z
        GOTO    MULTIPLO

        MOVLW   D'8'
        ADDWF   AUX10,F
                MOVLW   D'160'
                XORWF   AUX10,W
                BTFSC   STATUS,Z
                GOTO    EXCEPT_02
                GOTO    DIV8


MULTIPLO
                MOVLW   D'0'
                MOVWF   AUX10

DIV32           MOVF    AUX10,W
                SUBWF   ST_ADDRESS_LO,W
                BTFSS   STATUS,C
                GOTO    EXCEPT_02
                ADDWF   Q_INPUTS_LO,W   ;
                SUBLW   D'32'
                BTFSC   STATUS,C
                GOTO    D_GOOD

                MOVLW   D'32'
                ADDWF   AUX10,F
                MOVLW   D'160'
                XORWF   AUX10,W
                BTFSC   STATUS,Z
                GOTO    EXCEPT_02
                GOTO    DIV32



EXCEPT_02
                MOVLW   H'02'
                MOVWF   EXCEPT
                RETURN

EXCEPT_03
                MOVLW   H'03'
                MOVWF   EXCEPT
                RETURN

D_GOOD          CLRWDT
                RETURN


;-------------------------------------------------------------------------------
; * FUNCION: Retardo de 5 ms                                                   *
;-------------------------------------------------------------------------------

DELAY_5MS
                MOVLW   D'41'
                MOVWF   COUNT2

DELAY
                CALL    DELAY_125
                DECFSZ  COUNT2,F
                GOTO    DELAY
                CLRWDT
                RETURN
```

```
;-------------------------------------------------------------------------------
; * FUNCION: Retardo de 125 us                                                  *
;-------------------------------------------------------------------------------

DELAY_125
                    MOVLW  D'42'
                    MOVWF  COUNT1
REPEAT
                    DECFSZ  COUNT1,F
                    GOTO    REPEAT
                    RETURN


;-------------------------------------------------------------------------------
; * FUNCION: Retardo de K x 100ms @ 4MHz                                        *
; * ENTRADA: K en W, rango de 1 a 256                                          *
; * SALIDA : Flags y W alterados                                               *
;-------------------------------------------------------------------------------

DELAY_K100MS
                    MOVLW  H'10'
                    MOVWF  K

;TAREA1: HACER UN RETARDO DE 100ms

                    MOVLW  D'130'
                    MOVWF  COUNT_H
                    CLRF   COUNT_L

DK_LOOP

                    DECFSZ  COUNT_L,F
                    GOTO    DK_LOOP
                    DECFSZ  COUNT_H,F
                    GOTO    DK_LOOP

;TAREA2: DECREMENTAR K

                    DECFSZ  K,F
                    GOTO    DK_LOOP
                    CLRWDT
                    RETURN
```