UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELECTRICA

"DISEÑO Y CONSTRUCCION DE UN SISTEMA INTELIGENTE PROGRAMABLE VIA ETHERNET PARA LA OPTIMIZACION DE ENERGIA ELECTRICA DESDE UNA APLICACIÓN WEB"

PRESENTADO POR:

ERNESTO ARQUIMIDES CASTELLON TORRES

CARLOS ROBERTO ROMERO MIRANDA

PARA OPTAR AL TITULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, SEPTIEMBRE DE 2009

UNIVERSIDAD DE EL SALVADOR

RECTOR :

MSC. RUFINO ANTONIO QUEZADA SÁNCHEZ

SECRETARIO GENERAL :

LIC. DOUGLAS VLADIMIR ALFARO CHÁVEZ

FACULTAD DE INGENIERÍA Y ARQUITECTURA

DECANO :

ING. MARIO ROBERTO NIETO LOVO

SECRETARIO :

ING. OSCAR EDUARDO MARROQUÍN HERNÁNDEZ

ESCUELA DE INGENIERÍA ELÉCTRICA

DIRECTOR :

ING. JOSÉ WILBER CALDERÓN URRUTIA

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO ELECTRICISTA

Título :

“DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA INTELIGENTE PROGRAMABLE VÍA ETHERNET PARA LA OPTIMIZACIÓN DE ENERGÍA ELÉCTRICA DESDE UNA APLICACIÓN WEB”.

Presentado por :

ERNESTO ARQUIMIDES CASTELLÓN TORRES

CARLOS ROBERTO ROMERO MIRANDA

Trabajo de Graduación Aprobado por:

Docente Director :

Ing. Hugo Miguel Colato Rodríguez

San Salvador, Septiembre de 2009

Trabajo de Graduación Aprobado por:

Docente director:

Ing. Hugo Miguel Colato Rodríguez

AGRADECIMIENTOS

Agradezco a Dios Padre y a su hijo Jesucristo por darme la vida, el entendimiento y la fuerza para lograr esta meta muy significativa en mi vida. Le doy gloria y honra a El por culminar mi carrera universitaria y por estar conmigo todos los días de mi vida.

Agradezco y dedico este trabajo a mi madre Rosa Miriam Torres por darme su soporte emocional, moral y económico. Y por estar ahí siempre que necesite su apoyo. Agradezco a mi padre Juan Antonio Castellón por darme la vida y también agradezco a mi segundo padre José Mauro Orellana por servir de inspiración para elegir esta carrera por su cuido y correcciones en mi vida.

A mis hermanos Elio Enrique Castellón Torres y Rubén Eduardo Orellana Torres, por sus muestras de cariño y apoyo en el transcurso de esta ardua jornada. Gracias a toda mi familia por estar siempre ahí dispuesta a ayudar.

A mi novia y futura esposa, por sus oraciones, cariño y amor. Por su comprensión y apoyo. Por escucharme en momentos difíciles, por esperarme a culminar mi carrera y por muchas cosas más. Muchas gracias mi baby.

A mi compañero y colega Carlos Roberto Romero Miranda por compartir alegrías, tristezas y sufrimiento en el transcurso del desarrollo de este trabajo el cual se culmino y nos permitió fortalecer nuestra amistad.

A nuestro asesor Ing. Hugo Miguel Colato, por su dedicación, esfuerzo y tiempo. Por haber creído en este proyecto.

A todos mis Catedráticos e instructores que sirvieron de motivación.

A todos mis compañeros y colegas de la carrera de ingeniería eléctrica, con los cuales compartimos noches de desvelos estudiando.

A mis amigos Ovidio Reyes, Ricardo Ernesto Castillo, Roberto Romero, Joel Rivera, José Mario Valdez, a meme y al niño por haber contribuido de una y otra manera en la realización de este proyecto

A mis pastores y consejeros René y Ligia Ramírez por estar pendientes de mi vida, por sus oraciones y cuidos incondicionales. Estoy infinitamente agradecidos y solo les puedo decir que Dios los Bendiga.

ERNESTO ARQUIMIDES CASTELLON TORRES

AGRADECIMIENTOS

Agradezco y dedico los frutos de este logro a ti Dios Padre y Señor mío, que desde mi concepción y sin merecerlo me has dado tu inmenso amor, decidiste dejarme al lado de los seres que más amo cuando apenas era un niño porque tenias ya un camino lleno de bendiciones preparado para mí; gracias a ti mi Señor Jesús, por estar a mi lado y enviarme ángeles cuando el esfuerzo dedicado a esta tesis me dejaba cansado y sin fuerzas.

Tu, mami, Marta Catalina Miranda, eres el ángel más bello que Dios me dio, mi pedacito de cielo, a ti dedico y agradezco este logro porque sin tu apoyo, ejemplo y guía no hubiera podido culminar mi carrera, has sido mi mayor inspiración y ejemplo de humildad y tenacidad. Sin tus cuidos madre mía esto no sería posible. Te amo.

A ti, mamita, María Ángela Miranda, agradezco infinitamente tu apoyo incondicional a mi madre, hermano y a mí, agradezco tus cuidos y tu fe en mí. Gracias mamita también tuyo es este esfuerzo porque siempre que lo he necesitado ha estado tu amor presente.

A mi padre, Jorge Alberto Romero, que con su dirección desde pequeño me enseño a usar el sentido común, forjo mi carácter e inspiró en mi los valores morales que me ayudaron en el transcurso de mi carrera y que serán mi estandarte en lo profesional, gracias por enfrentar los difíciles retos que la vida puso a tu paso con mi llegada al mundo, por tu amor y por ayudarme a crecer.

Gracias, mi hermanito. Jorge Eduardo Romero, tu eres mi otro ángel, gracias por tus cuidos cuando el desvelo y cansancio mermaban mis fuerzas, tuya era la mano que me cubría del frio; te agradezco hermano todo tu apoyo y aporte en este trabajo. Espero te sirva de inspiración para que recuerdes que aunque difícil, todo se puede lograr con la ayuda de Dios.

A ti Ojos, Elena Benavides, dedico esta tesis porque tu amor, paciencia y cariño fueron el impulso que recargaba mis energías. Tu inteligencia y la grandeza de tu corazón son las cualidades que más admiro de ti. Gracias amor porque siempre me distes ánimos para culminar esta meta. Espero en Dios amor que tanto para José Eduardo como Elena Sofía, nuestros futuros hijos, seas la inspiración como lo eres para mí, te amo.

Ernesto Arquimides Castellón Torres, mi amigo, gracias por no dejar de creer en este proyecto, por poner tu hombro junto al mío e impulsar la carreta cuando mi rueda se estancaba, por no desfallecer cuando la frustración nos abrumaba; por tu esfuerzo, desvelo y aporte de ideas para culminar satisfactoriamente este trabajo, te estaré siempre agradecido. El mayor de los éxitos en la profesión que ahora comienzas.

Ingeniero Hugo Colato, igualmente me siento agradecido porque su paciencia y apoyo hicieron posible que lográramos el cometido. Creyó en nosotros y agradezco en gran manera su aporte a mi formación profesional.

A mis preciosas tías, familia y amigos, gracias.

CARLOS ROBERTO ROMERO MIRANDA

PREFACIO

La tecnología al servicio del hombre.

El presente trabajo de graduación es una recopilación de novedosas tecnologías puestas al servicio del hombre para la optimización de energía eléctrica.

En el futuro se vislumbra el concepto de “Smart Grid” o redes inteligentes que permitirán el control y optimización energética a distancia; aunque el Sistema Inteligente desarrollado en este trabajo dista mucho de ser una red inteligente; pero, es un paso importante en el entendimiento de la importancia que la optimización de energía tiene no solo para El Salvador sino para el mundo entero.

Los recursos de El Salvador deben preservarse y optimizarse. Es por ello que este trabajo de graduación juega un papel importante en el desarrollo de sistemas inteligentes capaces de optimizar los recursos energéticos de nuestro país.

El sistema inteligente es el resultado del diseño ingenieril de esta tesis, provee características de optimización a través de la administración, el control y el monitoreo de cargas eléctricas por medio de la configuración de eventos que controlan el comportamiento de esas cargas.

El sistema consta de una aplicación Web desde donde el usuario debidamente autenticado tiene acceso a tres controladores inteligentes, para controlar Aires Acondicionado y luces, Accesos electrónicos o chapas y Administrar un reloj de tiempo real; proveyendo al lugar donde sea aplicado de una administración programable del consumo energético de las cargas controladas por el sistema.

RESUMEN DEL TRABAJO

El presente trabajo de graduación denominado "DISEÑO Y CONSTRUCCION DE UN SISTEMA INTELIGENTE PROGRAMABLE VIA ETHERNET PARA LA OPTIMIZACION DE ENERGIA ELECTRICA DESDE UNA APLICACIÓN WEB", es la actualización y reingeniería tecnológica de diseño del prototipo modulo electrónico programable vía Ethernet desde la WEB, denominado SACME[1] v1.0, que administra el consumo de energía eléctrica, controlando y monitoreando el estado de cargas y actuadores ON/OFF a través de una aplicación WEB; El resultado final, es el sistema inteligente denominado SIPROE[2] v1.0 el cual reutiliza las funcionalidades del modulo electrónico.

Las modificaciones que presenta el Sistema desarrollado son a nivel de hardware, firmware y software. Entre las modificaciones a nivel de hardware están el manejo de entradas y salidas análogas, comunicación intermodular sin interferencia de ruido por medio del protocolo CAN[3] para expandir la red a distancias de hasta 100 m y el control autónomo de eventos a través de un reloj calendario de tiempo real.

Las modificaciones realizadas al firmware incluyen la reestructuración del código de implementación del Protocolo MODBUS adaptándolo a las características del hardware, la implementación de las funciones 03 y 16 del protocolo MODBUS lectura y escritura de registros correspondientemente. La implementación de la comunicación a través de MODBUS sobre CAN, el control de salidas y entradas por expansores I2C y la administración de hasta 16 eventos de forma autónoma por cada controlador.

Entre las modificaciones realizadas al software están el rediseño de la interfaz grafica más amigable para configuración de eventos, peticiones MODBUS funciones 02, 03, 05 Y 16, y la reestructuración del repositorio de información para guardar datos de los nuevos controladores diseñados.

Todo esto para que en conjunto se pueda optimizar energía eléctrica por medio del manejo inteligente de los sistemas de iluminación y aire acondicionado. Como también proporcionar un mecanismo de acceso y seguridad para cualquier instalación.

En el capítulo I, se desarrollan las bases teóricas necesarias para comprender la filosofía de funcionamiento de cada una de las partes que componen a SIPROE, se desarrollan los conceptos para la implementación del protocolo MODBUS, TCP/IP y serial RTU, CAN e I2C. Asimismo se describen las tecnologías utilizadas para el desarrollo de la aplicación WEB basada en el modelo JAVA2EE, servidor WEB y base de datos.

En capítulo II, se detalla cada parte o componente del diseño del hardware utilizado para implementar el prototipo SIPROE V1.0, el cual cuenta con un controlador maestro que recibe las peticiones MODBUS-TCP/IP y las procesa si son peticiones para el modulo

[1] SACME : **S**istema de **A**dministración, **C**ontrol y **M**onitor de **Energía** Eléctrica
[2] SIPROE: **S**istema **I**nteligente **pr**ogramable **O**ptimizador de **E**nergía Eléctrica
[3] CAN : **C**ontrol **Á**rea **N**etwork

electrónico SACME o las retransmite por el bus CAN si las peticiones son para los controlares de iluminación, aire acondicionado, acceso y reloj de tiempo real.

En el capítulo III, Se describe cada una de las partes del desarrollo software que ofrece una Interface hombre maquina (HMI) amigable para la comunicación con los diferentes controladores y la administración de dispositivos según jerarquía de roles. Además de describir la interacción del usuario a través del aplicativo web tanto con la base de datos como el aplicativo intermedio el cual provee la comunicación con el hardware del sistema.

TABLA DE CONTENIDO

Capitulo **Página**

LISTA DE TABLAS

CAPITULO I.

CAPITULO II

CAPITULO III

ANEXOS

LISTA DE FIGURAS

CAPITULO I.

CAPITULO II

CAPITULO III

ANEXOS

OBJETIVOS

Objetivo General

Diseñar e implementar un sistema inteligente prototipo para la optimización en el uso de la energía eléctrica. Controlando, administrando y monitoreando los sistemas de iluminación y aire acondicionado. Como también proporcionar una opción de control de acceso y seguridad para áreas restringidas desde una aplicación WEB.

Objetivos Específicos

- Reutilizar, ampliar y aplicar reingeniería de diseño a las funcionalidades del modulo electrónico SACME V1.0

- Diseñar y construir controladores con funcionalidades de manejar sistemas de iluminación, aire acondicionado y control de acceso.

- Optimizar la comunicación entre el controlador maestro de SIPROE V1.0 y los controladores inteligentes, utilizando el protocolo CAN para lograr mayor rendimiento en la transferencia de información, alcance entre los mismos y reducir la interferencia electromagnética.

- Dotar a SIPROE V1.0 de un grado de autonomía para que una vez configurado no dependa de la aplicación WEB para funcionar correctamente.

- Rediseñar el firmware para que implemente las funciones 03 y 16 del MODBUS y el protocolo CAN.

- Rediseñar el hardware para que procese y expanda sus funcionalidades a entradas/salidas análogas.

- Adaptar el hardware para que se pueda implementar el protocolo CAN y un reloj calendario de tiempo real en el bus CAN.

MARCO TEORICO

Introducción.

El módulo electrónico SACME V1.0 posee las características necesarias para administrar el funcionamiento de hasta ciento sesenta dispositivos eléctricos ON-OFF. Cuenta con las funciones necesarias para ser utilizado por otros sistemas en modalidad de esclavo, para ello tiene embebido un servidor MODBUS, el cual puede ser gestionado por cualquier cliente MODBUS. SACME dispone de una aplicación Web para manejar al módulo electrónico y dotar al usuario de una interface 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 de SACME.

En este capítulo se definen las bases teóricas de la reingeniería de diseño implementada al modulo SACME, a partir de las cuales se ha desarrollado SIPROE v1.0, desarrollándose en cada una de las secciones la teoría sobre los protocolos de comunicación modbus, CAN e I2C, la teoría de configuración e implementación de cada uno de los componentes del hardware y las características del software que comunica el Sistema SIPROE con los usuarios.

1.1 Protocolo MODBUS

Básicamente MODBUS es un protocolo de la capa de aplicación en el esquema del modelo OSI para la comunicación entre dispositivos y principalmente para el intercambio de datos en el campo de la automatización. MODBUS es un protocolo cliente-servidor parecido a http, basado en transacciones, el cual consiste de una petición hecha por el cliente y una respuesta generada por el servidor. En el campo de aplicación es uno de los esquemas que imperan en el comportamiento de la comunicación de bajo nivel en una red que utiliza un cable de señal compartida: maestro-esclavo. Para evitar la confusión, la relación directa que describe Maestro-Esclavo en términos de Cliente-Servidor es:

- El **Maestro** es un **Cliente**.
- El **Esclavo** es un **Servidor**.

La comunicación está basada en un simple paquete o trama, llamado Unidad de Datos del Protocolo (PDU) que se observa en la Figura 1.1 . La especificación del protocolo define tres tipos de PDU como se describen en la Tabla 1.1

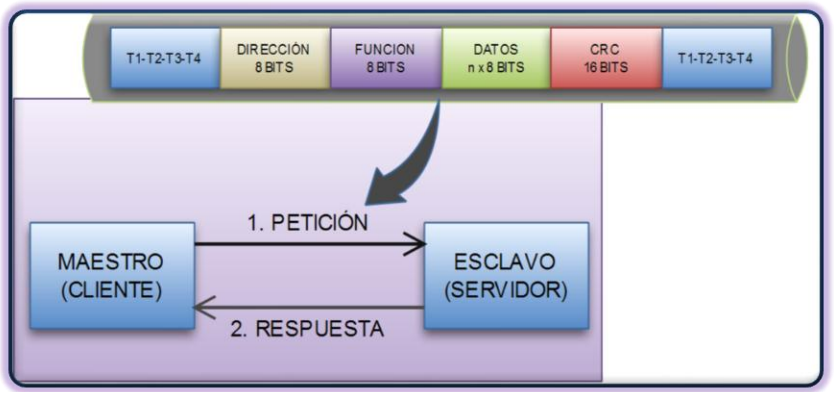


Figura 1.1 Especificación de la comunicación MODBUS

Tabla 1.1 Tipos de Unidad de Datos PDU

Tipo de PDU	Descripción
PDU de Petición	Consiste de (1) un código especificando una función (Código de Función, 1 byte) y (2) datos de la función específica (el numero de bytes varia)
PDU de Respuesta	Consiste de (1) el código de función correspondiente a la petición (Código de Función, 1 byte) y (2) Datos de Respuesta (el numero de bytes varia)
PDU de Respuesta Excepción	Consiste de (1) el código de Función correspondiente a la petición + 0x80 (Código de Error, 1 byte) y (2) un código especificando la excepción (Código Excepción, 1 byte)

1.1.1 Funciones de petición MODBUS

En la Figura 1.2 se describe el procesamiento de peticiones Modbus. La especificación MODBUS define un número de funciones a las que se le asigna un código de función único, estos códigos están en el rango de 1-127 y 129-255 para los códigos de excepciones. La actual especificación define categorías de códigos de función como lo muestra la Tabla 1.2.

La documentación para cada función consiste de:

- Una descripción de la función, sus parámetros y valores de retorno, incluyendo posibles excepciones.
- El Código de Función asignado
- El PDU de petición
- El PDU de Respuesta
- El PDU de Respuesta Excepción

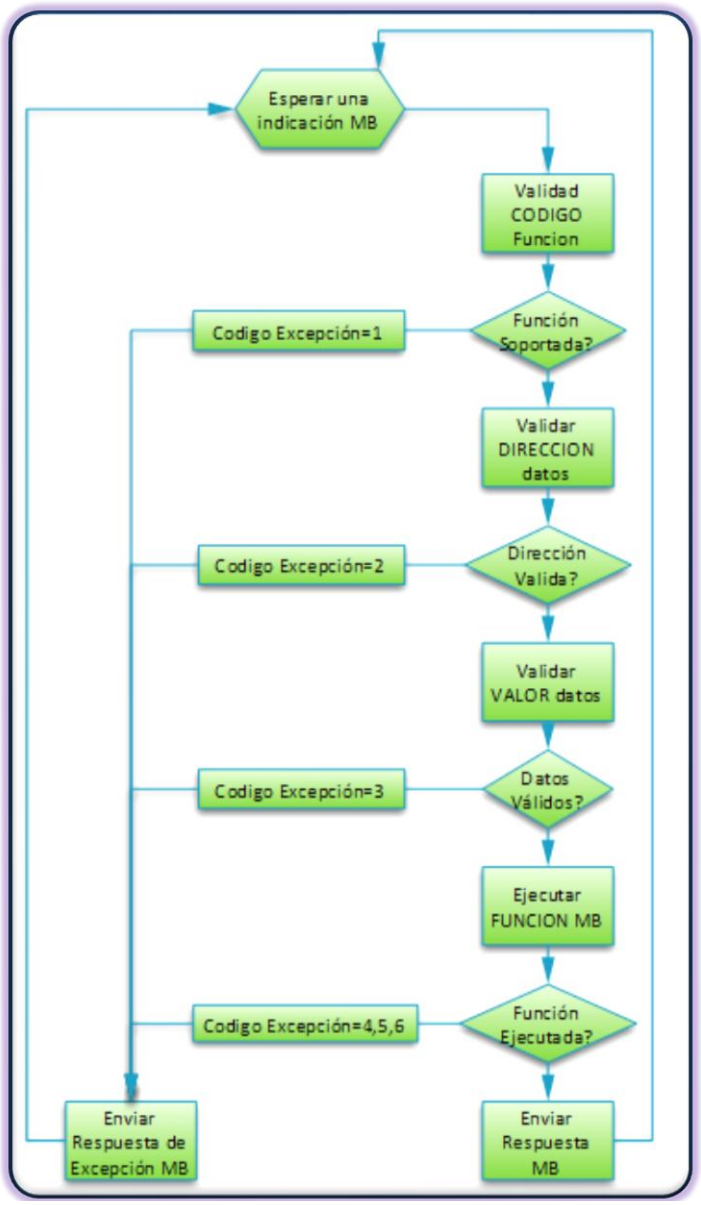


Figura 1.2 Procesamiento de Peticiones Modbus

Tabla 1.2 Categorías de códigos de Función.

Categoría	Descripción
Público	Son códigos únicos y específicos de funciones bien definidas que están públicamente documentadas.
Definido por Usuario	Están disponibles para funciones definidas por el usuario, por lo que su código podría no ser único. La especificación define los rangos de 65-72 y 100-110 para funciones definidas por el usuario.
Reservado	Estas son usadas por algunas compañías para legalidad de productos y no están disponibles para uso público.

1.1.2 Excepciones

Bajo ciertas circunstancias, la respuesta de un esclavo será una excepción. La primera identificación de una respuesta de excepción es el código de error (código de función +128), el cual es seguidamente especificado por el código de excepción. La Tabla 1.3. Muestra las excepciones generadas por MODBUS.

1.1.3 Modelo de Datos MODBUS

Las funciones públicas básicas han sido desarrolladas para intercambiar datos típicos en el campo de la automatización. La Tabla 1.4 contiene los tipos de datos básicos MODBUS definidos por la especificación.

1.1.4 Modos de transmisión MODBUS

Básicamente MODBUS ha sido implementado y usado sobre todo tipo de medio físico (cable, fibra, radio) y varios tipos de comunicación de bajo nivel; SIPROE utiliza dos modos básicos de transmisión:

- Serial : Maestro/Esclavo Asíncrono
- IP : Maestro /Esclavo

El protocolo modbus serial opera basado en el estándar de comunicación serial, dentro de los que se encuentran el RS232, Usado en cortas distancias en comunicaciones punto a punto; RS485, Usado en comunicación multipunto, múltiples dispositivos conectados al mismo cable de señal.

Tabla 1.3 Códigos de excepción de funciones MODBUS.

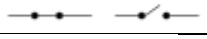
Código de excepción	Nombre MODBUS	Comentarios
01	Código de Función ilegal	El código función es desconocido por el servidor
02	Dirección de datos ilegal	Dependiendo de la petición
03	Valor de dato ilegal	Dependiendo de la petición
04	Fallo servidor	El servidor fallo durante la excepción
05	Reconocimiento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

SIPROE utiliza el medio serial para la comunicación entre el gateway MODBUS TCP/RTU y el microcontrolador. Las transmisiones modbus se realizan en dos modos los cuales difieren en codificación, trama y checksum. En el modo ASCII, las tramas son codificadas dentro de dos caracteres ASCII por byte, en forma hexadecimal. el checksum de error es representado por una redundancia longitudinal de chequeo (LRC, 1byte) y el mensaje comienza con un carácter especial (':', 0x3A) y termina con un retorno de línea y avance de carro ("CRLF", 0x0D0A), este tipo de tramas ocupa el doble de ancho de banda que RTU, debido a que dos bytes para un caracter. En el modo RTU, las tramas son transmitidas en binario para lograr una más alta densidad. el checksum de error es representado por un chequeo de redundancia cíclica (CRC16, 2byte), cuya función polinomial se muestra en la Figura 1.3.

$$\boldsymbol{Polinomio\ CRC16\ Modbus = 1 + x^2 + x^{15} + x^{16} = 0xA001}$$

Figura 1.3 Polinomio para el cálculo del CRC

Tabla 1.4 Tipos de datos MODBUS.

NOMBRE	TIPO	ACCESO	VISUAL	FUNCIÓN IMPLEMENTADA EN SIPROE
Entrada Discreta	1 bit	Solo lectura		Función 2
Salida Discreta	1 bit	Lectura-Escritura		Función 5
Registros de Entrada	Palabra 16-bit	Solo Lectura		Función 3
Registros de Retención	Palabra 16-bit	Lectura-Escritura		Función 16

En el modo IP, MODBUS utiliza un stack TCP/IP para la comunicación por el puerto registrado 502, extiende el PDU con un encabezado IP, llamado MBAP, compuesto de 7 bytes y de los siguientes campos descritos en la Tabla 1.5.

Tabla 1.5 Campos del encabezado MBPA del empaquetado TCP

CAMPO	LONGITUD	DESCRIPCION
Identificador de transacción	2 bytes	El Identificador de una petición MODBUS o respuesta de petición.
Identificador de protocolo	2 bytes	El Identificador de Protocolo, 0 para MODBUS por defecto.
longitud	2 bytes	un conteo de los byte siguientes
Identificador de unidad	1 byte	Identificador de Unidad utilizado para identificar una unidad remota en una red no TCP/IP

Como se muestra en la Figura 1.4, el empaquetado resultante es denominado Unidad de Datos de Aplicación (ADU).

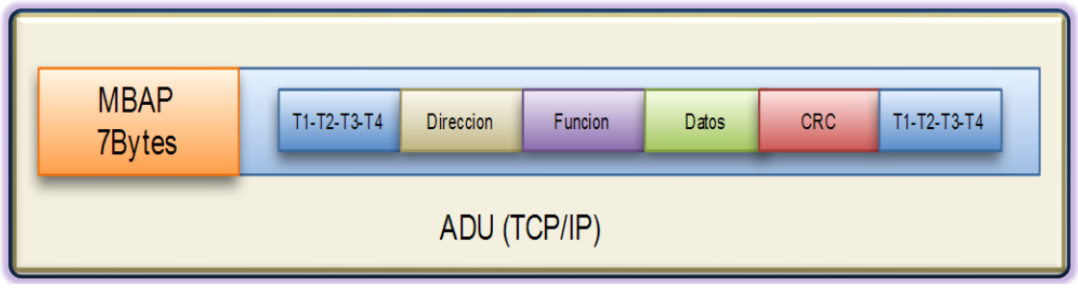


Figura 1.4 Empaquetado TCP/IP de la trama MODBUS

Esta implementación TCP/IP de MODBUS utiliza el medio Ethernet para la comunicación entre el controlador maestro y el software basado en Java EE u otra aplicación que comparta el medio físico e implemente MODBUS.

1.2 Transmisiones MODBUS TCP/IP basadas en el API JAMOD

JAMOD es un API basado en Java que permite la implementación de comunicaciones MODBUS, está diseñado para soportar ambos modos de transmisión, usando una implementación basada en la API javax.comm; por lo tanto, esta API se ha utilizado para la implementación de MODBUS desde la capa de aplicación.

La aplicación WEB de SIPROE desarrolla una implementación en JAMOD de las funciones 2, 3, 5 y 16, en la Tabla 1.6 se describen los paquetes que conforman la implementación JAMOD y en la Tabla 1.7 se describen las clases con las que se desarrollan las transacciones de petición y respuesta desde la aplicación WEB.
La incorporación del API JAMOD al desarrollo de la interface gráfica de usuario se logra de manera sencilla incorporando a las librerías del desarrollo Java, el archivo compilador de toda la API denominado **Jamod.jar.**

Tabla 1.6 Paquetes que conforman JAMOD

PAQUETE JAMOD	DESCRIPCIÓN
net.wimpi.modbus	Clase principal que provee una Implementación orientada a objetos de MODBUS en Java.
net.wimpi.modbus.cmd	Clase con Herramientas demostrativas desde la línea de comandos
net.wimpi.modbus.facade	Proporciona implementaciones de patrón facade
net.wimpi.modbus.io	Proporciona clases e interfaces relacionadas con transporte y E/S
net.wimpi.modbus.msg	Proporciona interfaces y clases que encapsulan mensajes MODBUS en un objeto.
net.wimpi.modbus.net	Proporciona partes de esta implementación MODBUS relacionadas con la red.
net.wimpi.modbus.procimg	Cuenta con interfaces y clases que proporcionan un acople entre un esclavo y un maestro.
net.wimpi.modbus.util	clases de utilidades y ayuda para el Implementación MODBUS.

Tabla 1.7 Clases utilizadas por SIPROE para transferencia de información.

Clases Utilizadas por SIPROE	Descripción
ExceptionResponse	Clase que implementa una Respuesta Modbus que representa una excepción.
ModbusRequest	Clase abstracta que implementa una Petición Modbus.
ModbusResponse	Clase abstracta que implementa una Petición Modbus.
ReadInputDiscretesRequest	Clase que implementa una Petición de Lectura de entradas discretas F2.
ReadInputDiscretesResponse	Clase que implementa una Respuesta de Lectura de entradas discretas.
ReadMultipleRegistersRequest	Clase que implementa una Petición de Lectura de múltiples Registros F3.
ReadMultipleRegistersResponse	Clase que implementa una Respuesta de Lectura de múltiples Registros.
WriteCoilRequest	Clase que implementa Petición de Escritura de bobina F5.
WriteCoilResponse	Clase que implementa Respuesta de Escritura de bobina.
WriteMultipleRegistersRequest	Clase que implementa Petición de Escritura de múltiples registros F16.
WriteMultipleRegistersResponse	Clase que implementa Respuesta de Escritura de múltiples registros.

1.3 Transmisiones MODBUS Serial RTU basadas en protocolo CAN

El protocolo MODBUS sobre CAN desarrollado, está basado en el modelo OSI utilizando la capa de aplicación y la capa física, re direcciona las peticiones MODBUS de los clientes, las cuales se procesaron en la capa de aplicación donde se obtiene la dirección del controlador a quien va dirigida la petición, empaquetando la petición en los datos de la trama CAN para luego ser enviadas sobre el bus, debido a que la longitud de datos permitida por el protocolo CAN es de 8 bytes, las peticiones son enviadas en tramas de 8 bytes. Cuando las peticiones MODBUS son recibidas por el ID del controlador a quien iban dirigidas, se procesan y se obtiene una respuesta MODBUS/CAN. La Figura 1.5 muestra el empaquetamiento CAN de la trama MODBUS

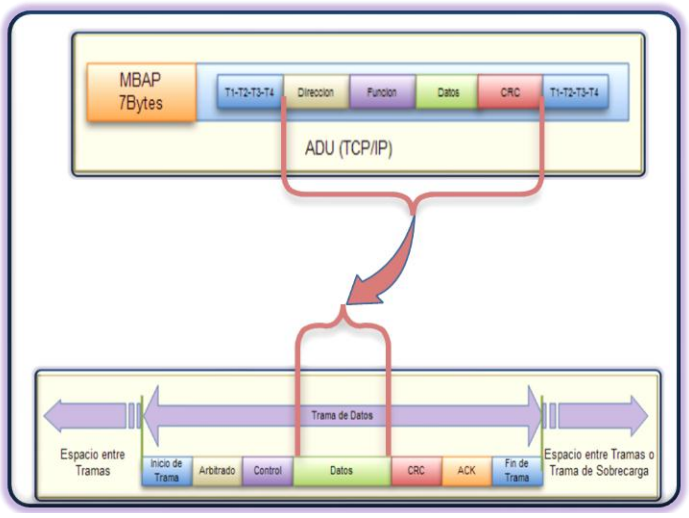


Figura 1.5 Empaquetamiento CAN de la trama MODBUS

1.4 Protocolo CAN

El Controlador de Área de Red (CAN) es un protocolo de comunicación serial que eficientemente soporta control distribuido de tiempo real con un muy alto grado de integridad en los datos. El protocolo CAN es un protocolo CSMA/CD. El CSMA[4] significa que cada **Controlador Inteligente** en la red debe monitorear el bus esperando un tiempo de inactividad antes de tratar de enviar un mensaje (Detección de Portadora), una vez ocurre cada controlador tiene igual oportunidad de transmitir un mensaje (Múltiple Acceso). El CD[5] significa que si dos controladores en la red comienzan a transmitir al mismo tiempo, detectaran una "colisión" y tomaran la acción apropiada para evitarla y mantener la integridad de los datos. En el protocolo CAN, se utiliza un método de arbitraje no destructivo a nivel de bit, lo que significa que los mensajes permanecen intactos después que el arbitraje es completado incluso si se detectan colisiones. El arbitraje toma lugar sin corromper o retardar el mensaje de mayor prioridad. Este protocolo es utilizado para transportar las tramas del protocolo MODBUS a través de los controladores Inteligentes, los cuales son descritos a detalle en el desarrollo del capítulo II.

1.4.1 Estructura de capas de un controlador CAN

Para lograr un diseño transparente y flexible de implementar, el CAN esta subdividido en tres capas o layers:

[4] Carrier Sense Multiple Access : múltiple acceso con detección de portadora
[5] Collision Detect : Detección de Colisión

- La Capa Objeto
- La Capa de Transferencia
- La Capa Física

Las Capas Objeto y Transferencia constituyen todos los servicios y funciones de la capa de enlace de datos definida en el modelo OSI. El alcance de la capa Objeto incluye: determinar cuáles mensajes serán transmitidos, decidir cuales mensajes recibidos por la capa de transferencia serán usados y proporcionar una interface al hardware relacionada a la "capa de aplicación". La capa de transferencia le concierne principalmente la transferencia del protocolo, es decir el control de tramas, efectuar el arbitraje, el chequeo de errores, etc. La Capa Física cubre la transferencia de bits entre controladores en lo que respecta a las propiedades eléctricas, en una red de controladores la capa física debe ser idéntica para todos.

La Capa Objeto también maneja el filtrado de mensajes, así como el manejo de estados y mensajes. La "capa de Transferencia" representa el kernel del protocolo CAN, envía los mensajes recibidos a la capa objeto y acepta mensajes a ser transmitidos por esa capa, es responsable de la sincronización y temporización de bits, trama del mensaje, arbitraje, reconocimiento, detección y señalización de error. La "capa Física" define como las señales son transmitidas.

1.4.2 Mensajes

La información de la trama MODBUS sobre el bus es enviada en mensajes con un formato ajustado de diferentes pero limitados tamaños. Cuando el bus esta libre cualquier nodo (**Controladores Inteligente** desde la perspectiva SIPROE) conectado puede iniciar la transmisión de un nuevo mensaje. El contenido de un mensaje es descrito por un identificador. El identificador no indica el destino del mensaje pero describe el significado de los datos así, todos los controladores en la red son capaces de decidir, por filtrado de mensaje, si el dato actuará sobre el controlador o no. Como consecuencia del concepto de filtrado de mensaje, cualquier numero de controladores puede recibir y actuar simultáneamente sobre el mismo mensaje esto se denomina Multicast. El identificador define una prioridad estática de mensaje durante el acceso al bus. Al estar libre el bus, el controlador que tenga el mensaje con la más alta prioridad obtendrá el acceso por tanto se considera un sistema Multi MAESTRO

1.4.3 Descripción de la Trama del Mensaje

El CAN define cuatro tipos diferentes de mensajes (o Tramas). El primer tipo y más importante es la **trama de Datos** utilizada cuando un controlador transmite información a otro o a todos los controladores en el sistema. Ver Figura 1.6

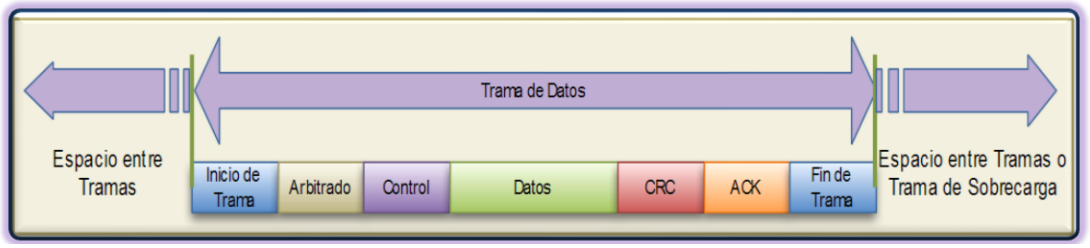


Figura 1.6 Trama de Datos del protocolo CAN

El segundo tipo es la Trama Remota la cual básicamente es una de datos, sin el campo de datos y con el bit RTR[6] en 1. Ver Figura 1.7**.**

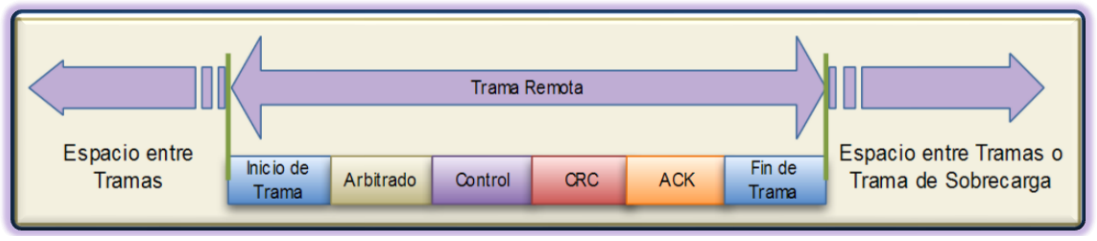


Figura 1.7 Trama Remota del protocolo CAN

Las otras dos tramas son para manejo de errores. Una es llamada Trama de Error y la otra Trama de Sobrecarga. Son generadas por controladores que detectan cualquiera de los errores definidos por CAN. Errores de Sobrecarga son generados por controladores que requieren más tiempo para procesar los mensajes recibidos.

Las Tramas de Datos consisten de campos que proporcionan información adicional acerca del mensaje. Embebidos en la trama de datos se encuentran los campos de Arbitraje, Control, Datos, CRC, reconocimiento de dos bit y el final de trama. Ver Figura 1.8.

El "Campo de Arbitrado" es utilizado para priorizar mensajes en el bus. Ya que el CAN define un lógico 0 como el estado dominante, un valor bajo en este campo determina un mensaje con prioridad más alta. Este campo consiste de 12-bits (11 bit de identificador y un bit RTR) o 32-bits (29 bits de identificador, 1-bit para definir el mensaje como una trama de datos extendida, un bit SRR y un bit RTR) para una trama estándar o extendida respectivamente. La petición de transmisión remota es usada por un controlador cuando este requiere el envío de información desde otro controlador.

[6] Remote Transmit Request: Petición de transmisión Remota si el bit asociado es recesivo el mensaje es una trama remota, si es dominante la trama es de datos

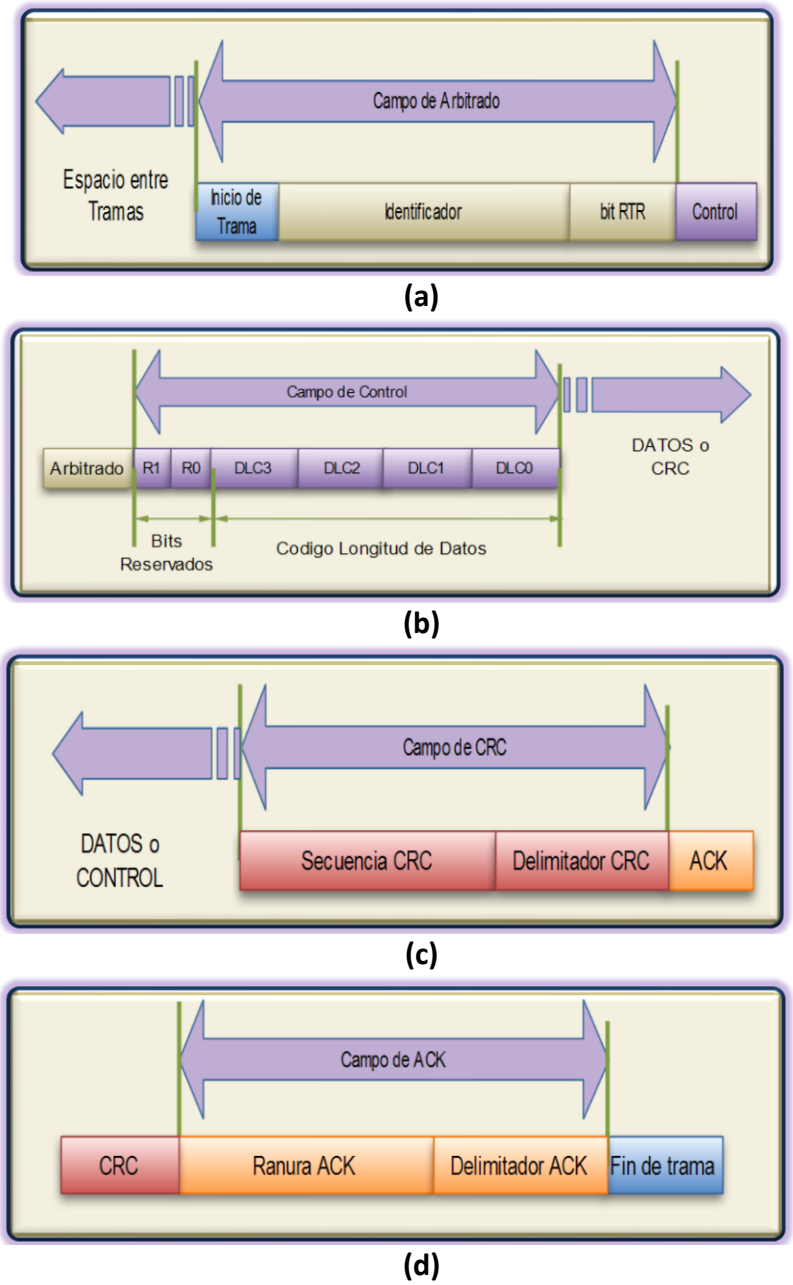


Figura 1.8 Campos trama de datos CAN. (a)Arbitrado, (b) Control, (c) CRC y (d) Reconocimiento.

El "Campo de Control" consiste de 6 bits. El MSB[7] es el bit IDE (trama extendida) el cual es dominante para tramas de datos estándar. En tramas extendidas este bit es R1 y está reservado al igual que el siguiente bit R0. Los cuatro LSBs son los bits de código de longitud de datos (Data Length Code, DLC). Estos determinan cuantos bytes

[7] Most Significant Bit: bit más significante

de datos son incluidos en el mensaje. La trama Remota no tiene campo de datos a pesar del valor de los bits DLC. La Tabla 1.8 muestra la codificación de estos bits.

Tabla 1.8 Codificación de longitud de datos.

CODIGO DE LONGITUD DE DATOS				CONTEO DE BYTES DATOS
DLC3	DLC2	DLC1	DLC0	
d	d	d	d	0
d	d	d	r	1
d	d	r	d	2
d	d	r	r	3
d	r	d	d	4
d	r	d	r	5
d	r	r	d	6
d	r	r	r	7
r	d	d	d	8
d= dominante r=recesivo				

El “campo de datos” consiste del número de bytes de datos descritos en el código de longitud de datos del campo de control. El “campo CRC” consiste de un campo de quince bits y un delimitador CRC, es usado por los controladores receptores para determinar si han ocurrido errores de transmisión.

El “Campo de Reconocimiento” es utilizado para indicar si el mensaje fue recibido correctamente. Si el controlador ha recibido correctamente el mensaje coloca un bit dominante en el tiempo del bit ACK sea que se procese o descarte el dato.

Debido a la flexibilidad del protocolo CAN, Los controladores inteligentes pueden ser agregados a la red CAN sin requerir algún cambio en el software o hardware de algún controlador o en la capa de aplicación, exceptuando el identificador Modbus-CAN que agrega a cada controlador la característica “Plug and Play”. El enlace de la comunicación serial CAN es un bus para el cual un numero de controladores que puede ser conectados es, teóricamente, ilimitado. Prácticamente, el número total de controladores está limitado por tiempos de retardo y/o las cargas eléctricas en el bus.

CAN especifica dos estados lógicos: Recesivo y Dominante. La ISO-11898 define un voltaje diferencial para representar los estados recesivo y dominante, como se muestra en la Figura 1.9. En el estado Recesivo (lógico ‘1’ en la entrada TXD del MCP2551) ver Figura 1.10. El voltaje diferencial en CANH y CANL es menor que el

umbral mínimo (<0.5V entrada receptora o <1.5V salida transmisora). En el estado dominante (lógico '0' en la entrada TXD del MCP2551), el voltaje diferencial en CANH y CANL es más grande que el umbral mínimo. ISO-11898-2 no especifica la conexión eléctrica. Sin embargo, la especificación requiere resistores terminales de 120Ω en cada final del bus. Ver Figura 1.11.

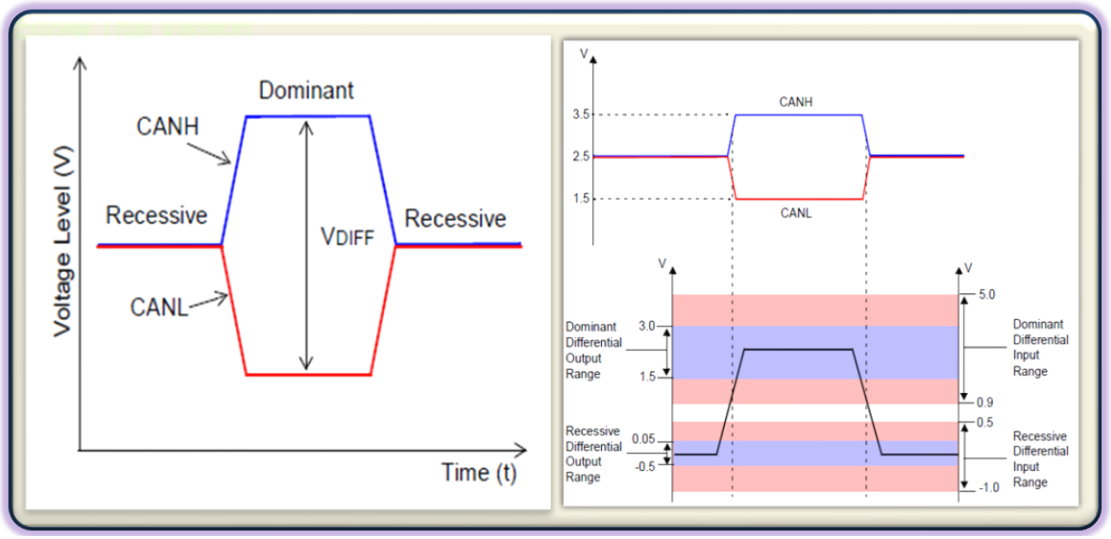


Figura 1.9 Niveles de voltaje del bus CAN para estados Recesivo y Dominante

ISO11898 especifica que un transceptor debe ser capaz de manejar un bus de 40m a 1Mb/s. Una longitud de bus más larga puede conseguirse disminuyendo la tasa de datos como puede observarse en la Tabla 1.10. La limitación más grande en la longitud es el retardo en propagación del transceptor. La Terminación del Bus con resistores minimiza la reflexión de la señal en el bus lo que permite lograr las longitudes máximas propuestas en la Tabla 1.10. Existen diferentes métodos de terminación del bus empleados para incrementar la longitud, disminuir la reflexión de la señal y aumentar el desempeño EMC[8].

[8] **EMC**: Electromagnetic Compatibility. Es una rama de la Ingeniería eléctrica que estudia la generación, propagación y recepción inintencionada de energía electromagnética, así como los efectos no deseados de esa energía. El objetivo de la EMC es el funcionamiento correcto en el mismo entorno de diferentes equipos que producen fenómenos electromagnéticos cuando operan.

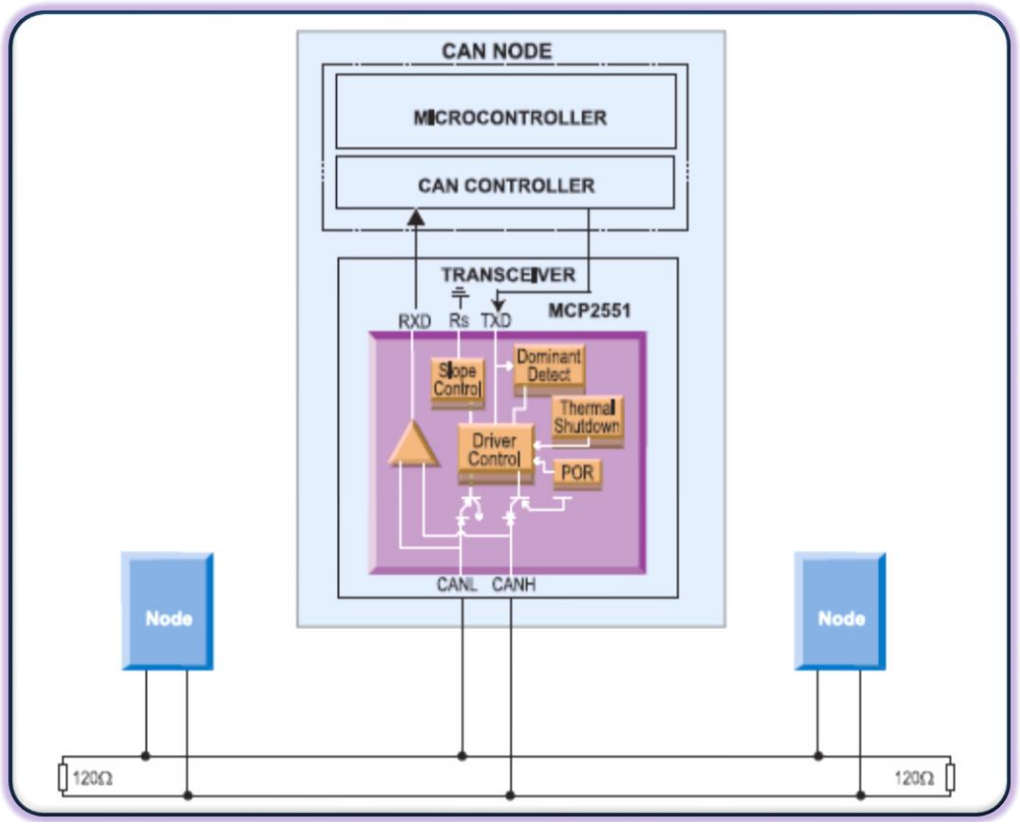


Figura 1.10 Conexión del transceptor MCP2551 al bus CAN.

Tabla 1.9 Tipos de terminaciones del bus CAN

TIPO DE TERMINACION	DESCRIPCION
Terminación estándar	Utiliza resistores de 120Ω en las terminaciones del bus.
Terminación partida	El resistor de 120Ω en cada extremo del bus es partida en 2 resistores de 60Ω, con un capacitor de bypass entre los dos resistores y tierra. Los dos resistores deben tener el mismo valor tanto como sea posible.
Terminación partida polarizada	Método utilizado para mantener el voltaje de modo común a un valor constante, incrementando así el rendimiento EMC. Este circuito es el mismo que la terminación partida con un divisor de voltaje adicional para alcanzar un voltaje de VDD/2 entre los dos resistores de 60Ω. Ver figura 1.11.

Tabla 1.10 Longitudes máximas del bus CAN según la tasa de transferencia.

Longitud del Bus (m)	Tasa de transferencia (Mbps)
40	1
100	0.5
200	0.25
500	0.10
1000	0.05

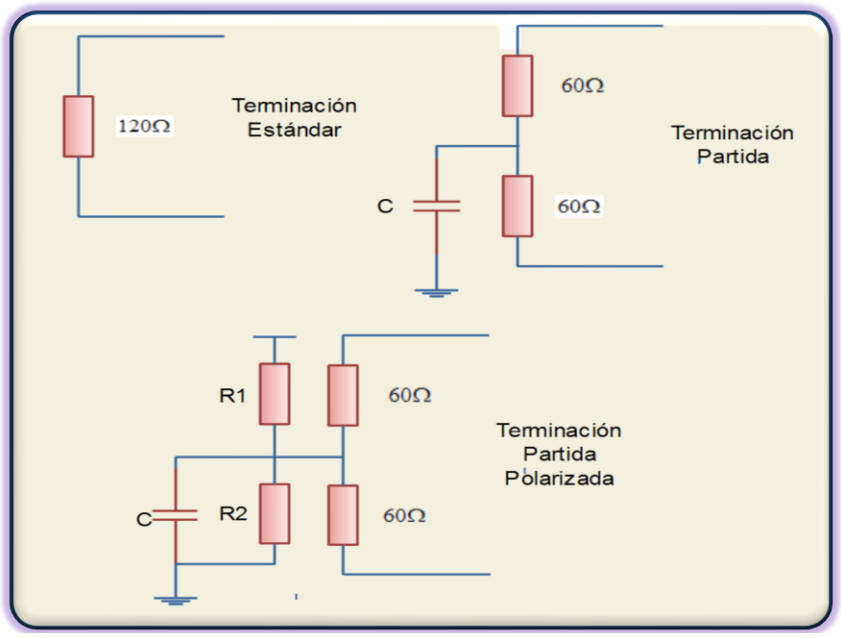


Figura 1.11 Métodos de terminación del bus CAN

Las características físicas mencionadas anteriormente proveen al protocolo CAN de ventajas sobre otros protocolos de comunicación como las descritas en la Tabla 1.11.

Tabla 1.11 Características generales de Sistemas de comunicación CAN

CARACTERISTICA	DESCRIPCION
Recepción Multicast	Con sincronización de tiempo múltiples controladores pueden simultáneamente recibir la trama.
Multi maestro	Cuando el bus esta libre cualquier unidad puede comenzar a transmitir un mensaje. La unidad con el mensaje de más alta prioridad gana el acceso al bus.
Prioridad de mensajes	Dependiendo de la importancia de los mensajes las prioridades son dadas a diferentes mensajes.
Arbitrado	Cada vez que el bus esta libre, cualquier unidad podrá comenzar a transmitir un mensaje. Si 2 o más unidades comienzan la transmisión simultáneamente, la unidad que adquiera el acceso dependerá de los bits de arbitraje usando el identificador. El mecanismo de arbitraje garantiza que ni el tiempo ni la integridad de la información se pierdan.
Enrutamiento de Mensaje	Un identificador nombra el contenido del mensaje. El identificador no indica el destino del mensaje, sino que describe el significado de los datos, de forma que todos los controladores en la red son capaces de decidir por filtrado de mensajes si tomaran acción sobre los datos.
Petición Remota de Datos	Enviando una trama remota un controlador requiriendo datos podrá solicitar a otro controlador enviar la trama correspondiente de datos. El mismo identificador nombra la trama de datos y la correspondiente trama remota.
Retransmisión	Reenvío automático de mensajes corruptos cuando el bus vuelve estar inactiva.
Flexibilidad en la Configuración	Teóricamente cualquier cantidad de controladores pueden ser agregados o removidos sin realizar modificaciones en el software o hardware.
Tasa de transferencia de bits	La velocidad de transferencia del mensaje CAN puede variar en diferentes sistemas, pero para un sistema específico la velocidad es uniforme y ajustada.
Reconocimiento:	Todos los receptores chequean la consistencia del mensaje recibido y reconocen al transmisor con un bit dominante en el campo ACK de la trama cuando la recepción de la trama es correcta.
Modos Sleep/Wakeup	Para reducir el consumo de potencia los dispositivos CAN puede configurarse en modo “dormido” sin ninguna actividad interna, desconectando los manejadores del bus.

1.4.4 Protocolo CAN en los PIC18Fxx8 de MICROCHIP.

La Tabla 1.12 describe las características principales del módulo CAN embebido en los microcontroladores (μC) PIC18Fxx8. La implementación del módulo es un sistema

CAN completo. Las reglas de compatibilidad para el manejo de tramas estándares y extendidas son las siguientes:

- Los controladores activos CAN 2.0B enviarán y aceptarán tramas estándares y extendidas
- Los controladores pasivos CAN 2.0B enviarán y recibirán tramas estándar y descartarán tramas extendidas sin generar errores.
- Los controladores CAN 1.0 generarán errores cuando reciban tramas extendidas.

Tabla 1.12 Características del modulo CAN embebido en los µC PIC18Fxx8

CARACTERISTICA	DESCRIPCION
Variantes del protocolo	La implementación del protocolo CAN1, 2, CAN 2.0A y CAN 2.0B
Tipos de tramas CAN	La trama de datos Estándar y Extendida
Longitud de datos	0-8 bytes de longitud de datos
Transferencia de datos	La tasa de transferencia programable hasta 1 Mbps
Petición remota de datos	Apoyo para Tramas Remotas
Capacidad de almacenamiento	Receptor de Doble-buffer con dos búferes priorizados de almacenamiento de mensajes recibidos
Filtrado de mensajes	Seis filtros completos de aceptación (identificador Estándar/Extendido), 2 asociados con el búfer receptor de prioridad alta y 4 asociados con el búfer receptor de prioridad baja
Enmascarado de mensajes	Dos máscaras del filtro de aceptación completa, cada una asociada con los búferes receptores de prioridad alta y baja
Capacidad de transmisión de datos	Tres búferes de transmisión con la aplicación de priorización especificada y la capacidad de abortar
Reconocimiento de tramas	La funcionalidad Programable de Activarse con el filtro pasa bajo integrado
Autocomprobación	Modo Programable de Bucle con soporte de la operación de autocomprobación
Sincronización	Conexión Programable al módulo temporizador para marcar tiempo y hacer sincronización en la red
Ahorro energético	Modo Sleep a BAJA POTENCIA

El módulo del bus CAN en los µC PIC18Fxx8 consiste de un motor del protocolo, de guardado y control de mensajes. El motor de protocolo maneja todas las funciones para recibir y transmitir mensajes, los cuales son transmitidos al cargar primero los registros apropiados de datos. El estado y los errores pueden también ser verificados

leyendo los registros apropiados. Cualquier mensaje detectado en el bus CAN es verificado por errores y luego emparejado contra los filtros para ver si debe ser recibido y almacenado en uno de los dos registros receptores. El módulo soporta los tipos de trama siguientes:

- La trama Estándar de Datos
- La trama Extendida de Datos
- La trama Remota
- La trama de Error
- La trama sobrecargada de Recepción
- El Espacio entre tramas

Los µC PIC18Fxx8 emplean los pines RB3/CANRX y RB2/CANTX/INT2 para la interface con el bus CAN. Para configurar CANRX y CANTX como interface CAN se requiere que:

- El bit TRISB<3> debe ser 1; configurado como entrada
- El bit TRISB<2> debe ser 0; configurado como salida.

Los µC PIC18Fxx8 tiene tres búferes de transmisión y dos de recepción, dos máscaras de aceptación (una para cada búfer de recepción) y un total de seis filtros de aceptación. La Figura 1.12 es un diagrama de bloques de estos búferes y su conexión al motor del protocolo. Hay muchos registros de control y datos asociados con el módulo CAN en la especificación de MICROCHIP. Por conveniencia, sus descripciones se agrupan en las secciones siguientes:

- Registros de Control y Estado
- Registros del Búfer transmisor (de Datos y Control)
- Registros del Búfer receptor (de Datos y Control)
- Registros de Control de Velocidad en baudios
- Registro de Control de E/S
- Registros de Control y Estado de Interrupción

Referirse a la bibliografía para mayor información sobre la utilización de cada sección de registro o a la hoja de especificaciones técnicas proporcionadas en cd adjunto con este documento.

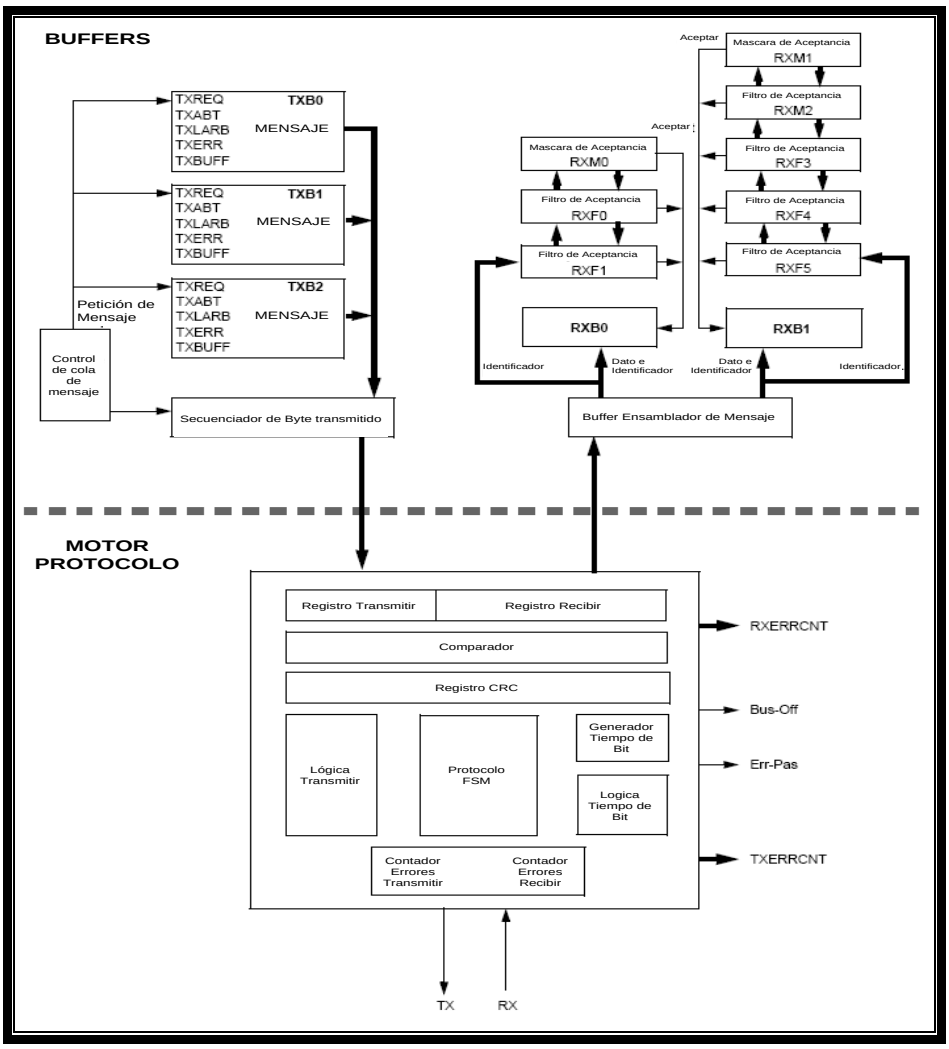


Figura 1.12 Diagrama de bloques del protocolo CAN.

1.4.5 Transceptor CAN

El transceptor es un dispositivo transmisor y receptor de tramas CAN; proporciona una interface entre el µC y el bus físico. Para una mayor compatibilidad se ha utilizado el transceptor MCP2551 del mismo fabricante del microcontrolador, MICROCHIP. Es completamente compatible con el estándar ISO-11898[9]. La velocidad de operación alcanza 1Mbps.

Cada Controlador Inteligente en SIPROE, debe tener un transceptor para convertir señales digitales generadas por el µC a señales adecuadas para transmisión sobre el bus cableado, de forma diferencial. También el transceptor cumple la función de

[9] Revisar bibliografía

búfer protegiendo al microcontrolador de picos de alto voltaje que pueden producirse en el bus.

El MCP2551 puede manejar una carga mínima de 45Ω, permitiendo un máximo de 112 Controladores de Funcionalidad, dando una resistencia de entrada diferencial mínima de 20kΩ y un valor nominal de resistor de terminación de 120Ω. En la Figura 1.13 se presenta el diagrama de bloques interno del MCP2551.

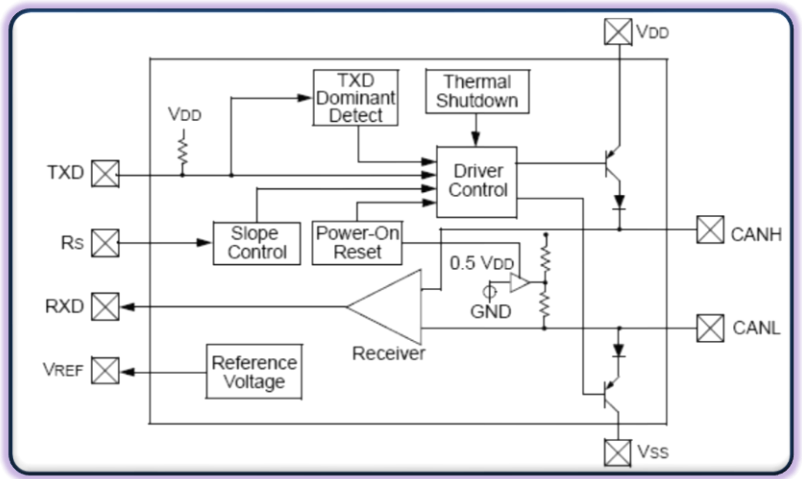


Figura 1.13 Diagrama de bloques del transceptor CAN.

1.5 Protocolo I2C

La comunicación interna de cada controlador inteligente se lleva a cabo por medio del protocolo I2C debido a la disponibilidad y versatilidad de los módulos expansores de entradas o salidas digitales y análogas que componen el controlador los cuales reciben las señales desde los sensores y envían señales hacia los actuadores.
Philips Semiconductors inventó el bus de dos conductores I2C para comunicación entre circuitos integrados en 1980, originalmente especificado para 100 kbits/s. El bus serial I2C ha sido extendido para soportar velocidades de hasta 3.4 Mbits/s. Combinado con una función de desplazamiento del nivel de voltaje, en modo de alta velocidad (HS-mode) ofrece una solución ideal para los sistemas de tecnología mezclada, donde las altas velocidades y la variedad de voltajes (5 V, 3 V o menor) son comúnmente usados. El modo Hs es compatible con todos los sistemas existentes del bus I2C, incluyendo el estándar original (S-mode) y el modo Rápido (F-mode), actualización introducida en 1992, proveyendo 400 kbits/s en transferencia. I2C es usado en una gran variedad de micro controladores y aplicaciones de telecomunicaciones como en control, diagnóstico y administración de potencia. Su simplicidad ha sido retenida independiente de las mejoras a la especificación original.

Tabla 1.13 Terminología básica del Bus I2C.

Términos	Descripción
Transmisor	El dispositivo que envía datos al Bus
Receptor	El dispositivo que recibe datos desde el Bus
Maestro	El dispositivo que inicia una transferencia, genera las señales del reloj y termina un envío de datos
Esclavo	El dispositivo direccionado por un maestro
Multi maestro	Más de un maestro puede controlar el bus al mismo tiempo sin corrupción de los mensajes
Arbitraje	Procedimiento que asegura que si uno o más maestros simultáneamente deciden controlar el Bus solo uno es permitido a controlarlo y el mensaje saliente no es deteriorado
Sincronización	Procedimiento para sincronizar las señales del reloj de dos o más dispositivos

I2C está orientado a las aplicaciones de 8-bit controladas por los µC, en la Tabla 1.14 se describen básicamente los criterios para la utilización del I2C en SIPROE.

En el bus I2C, dos hilos físicos uno de datos (SDA) y otro de reloj (SCL) transportan la información entre los diversos dispositivos conectados al bus. Cada dispositivo es reconocido por una única dirección (si es un microcontrolador, LCD, memoria o teclado) y puede operar cualquiera como transmisor o emisor de datos, dependiendo de la función del dispositivo.

Los dispositivos con funcionalidades de maestros son los µC PIC18Fxx8 que componen cada Controlador Inteligente. Los dispositivos conectados al bus son de colector abierto o drenaje abierto (“en paralelo”), así los estados de salida de las líneas de reloj (SCL) y dato (SDA) desempeñan la función de “cable en AND” del bus. La única limitación en la conexión de dispositivos al bus depende de la capacitancia máxima que no puede superar los 400 pF. Los modos de transferencia de datos en el bus son:

- Modo Estándar aproximadamente a 100 kbps.
- Modo Rápido aproximadamente a 400 kbps.
- Modo Alta velocidad más de 3,4 Mbps.

Tabla 1.14 Criterios de selección del bus I2C

CRITERIO	DESCRIPCION
Maestro-Esclavos	El sistema en donde es utilizado, consiste de al menos un microcontrolador y varios sistemas periféricos como dispositivos E/S
Costo	La conexión entre los varios dispositivos dentro del sistema debe ser el mínimo
Baja tasa de transferencia	El sistema que utiliza este Bus no requiere una alta tasa de transferencia de datos
Selectividad	La total eficacia del sistema depende de la correcta selección de la naturaleza de los dispositivos y de la interconexión de la estructura del bus

Tanto la línea de datos (SDA) como la Señal de Reloj (SCL) son bidireccionales conectadas a una fuente de tensión positiva vía suministro común o resistencias de carga (pull up). Debido a la variedad de diferentes tecnologías usadas en los dispositivos conectados al Bus I2C los niveles lógicos de "0" (Bajo) y "1" (Alto) no están fijados y dependen de la tensión de alimentación del circuito. Un pulso de reloj se genera por cada bit de datos transferidos.

Los bits de datos transferidos en la línea SDA deben ser estables cuando la línea SCL está a nivel "1". El estado de la línea SDA en "1" o "0" solo puede cambiar cuando en la línea SCL la señal es "0". Para operar un esclavo sobre el Bus I2C solo son necesarios seis simples pasos, suficientes para enviar o recibir información para la mayoría de dispositivos.

Tabla 1.15 Pasos para establecer comunicación entre dispositivos I2C

PASO	DESCRIPCION
1	Un bit de Inicio
2	7-bit o 10-bit de direccionamiento
3	Un R/W bit que define si el esclavo es transmisor o receptor
4	Direccionamiento de registro
5	Mensaje dividido en bytes
6	Un bit de Stop

Dentro del proceso de transferencia de datos en el Bus I2C hay dos situaciones básicas que son producidas por el maestro, el Inicio y el Paro de transmisión de toda

transferencia de datos. La Figura 1.14 muestra las condiciones de los pines SDA y SCL para que se reconozca cada situación.

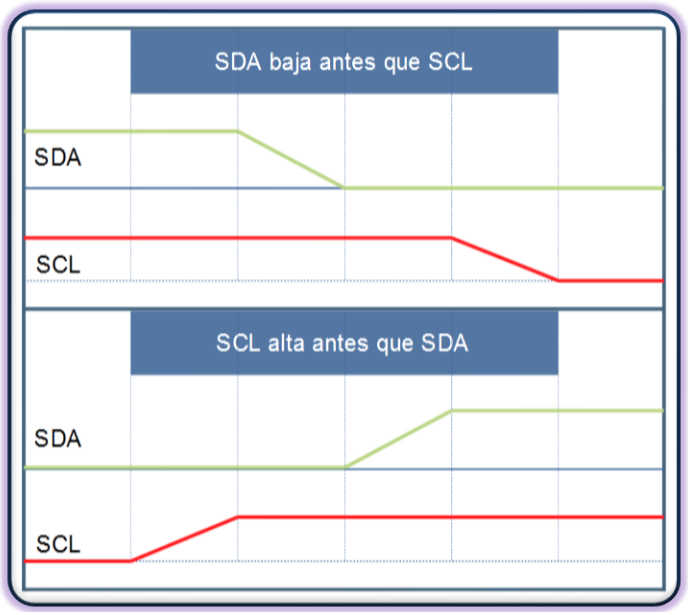


Figura 1.14 Condición de inicio y paro I2C.

1.5.1 Transferencia de Datos

El número de bytes a ser enviados por la línea SDA no tiene restricción. Cada byte debe ir seguido por un bit de reconocimiento, el byte de datos se transfiere empezando por el bit de más peso (7) precedido por el bit de reconocimiento (ACK). Si un dispositivo esclavo no puede recibir o transmitir un byte de datos completo hasta que haya acabado alguno de los trabajos que realiza, puede mantener la línea SCL a “0” lo que fuerza al Maestro a permanecer en un estado de espera. Los datos continúan transfiriéndose cuando el dispositivo esclavo está listo para otro byte de datos y desbloquea la línea de reloj SCL.
El bit de reconocimiento es obligatorio en la transferencia de datos. El pulso de reloj correspondiente al bit de reconocimiento (ACK) es generado por el receptor, todos los Maestros generan su propia señal de reloj sobre la línea SCL al transferir datos sobre el Bus I2C. Los bits de datos son solo validos durante los periodos “1” del reloj; la sincronización del reloj se realiza mediante una conexión AND de todos los dispositivos del Bus a la línea SCL. Esto significa que una transición de un Maestro de “1” a “0” en la línea SCL hace que la línea pase a “0”, esto mantiene la línea SCL en ese estado. Sin embargo la transición de “0” a “1” no cambia el estado de la línea SCL si otro reloj está todavía en su periodo de “0”. Por lo tanto la línea SCL permanecerá a “0” tanto como el periodo más largo de cualquier dispositivo cuyo nivel sea “0”. Los

dispositivos que tienen un periodo más corto de reloj “0” entran en un periodo de espera. Cuando todos los dispositivos conectados al Bus han terminado con su periodo “0”, la línea del reloj se desbloquea y pasa a nivel “1”. Por lo que hay que diferenciar entre los estados de reloj de los dispositivos y los estados de la línea SCL, y todos los dispositivos empiezan a nivel “1”. El primer dispositivo que completa su nivel “1” pone nuevamente la línea SCL a “0”.

Después de la condición de Inicio un código de dirección de un esclavo es enviada, esta dirección tiene 7 bits seguidos por un octavo bit que corresponde al bit de dirección de datos R/W (0-indica transmisión/1-indica solicitud de datos). Una transferencia de datos siempre acaba con una condición de Stop generada por el maestro. Sin embargo, si un maestro todavía desea comunicarse, puede generar repetidamente condiciones de Inicio y direccionar a otro esclavo sin generar primero la condición de Paro.

El procedimiento de dirección para el Bus I2C es tal que el primer byte después de la condición de Inicio usualmente determina que esclavo ha sido seleccionado por el Maestro. La excepción se da en la “llamada general” (byte $0000\ 0000_2$) con la que se direcciona a todos los dispositivos, cuando esta dirección es usada, todos los dispositivos en teoría deben responder con un reconocimiento (ACK), sin embargo algunos dispositivos pueden estar condicionados a ignorar esta dirección. El segundo byte de la “llamada general” define entonces la acción a tomar.

Hay dos formatos de dirección. El más simple es el formato de 7 bits con un bit R/W que permite direccionar hasta 128 dispositivos, que en la práctica se reduce a 112 debido a que las restantes direcciones son de uso reservado. El más complejo es el de 10-bit con un bit R/W. Para el formato de 10-bit, dos bytes deben ser transmitidos con los primeros cinco bits que especifiquen una dirección de 10-bit. El direccionamiento I2C empleado para todos los dispositivos utilizados es de 7 bits y su estructura se puede observar con más detalle en la Figura 1.15.

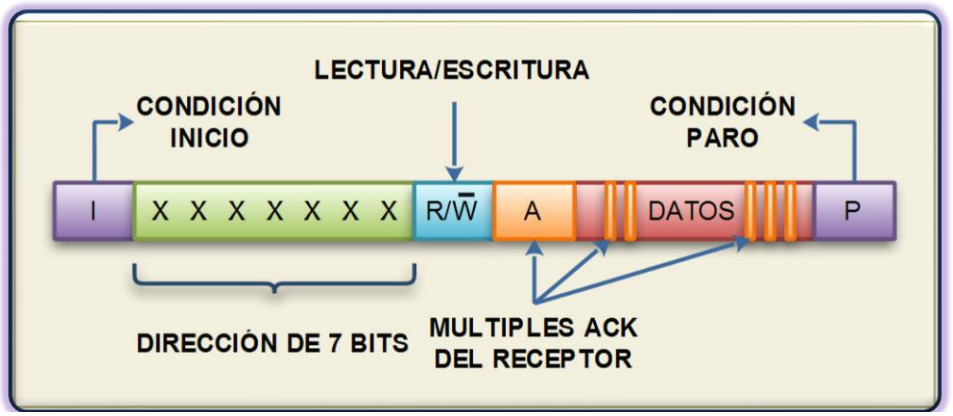


Figura 1.15 Dirección I2C de 7 bits.

1.5.2 Características I2C del Reloj Calendario de Tiempo Real RTC

El controlador inteligente RTC cuenta con un reloj de tiempo real, el DS1337 de MAXIM, cuya función es proporcionar tanto el calendario como la hora en tiempo real al Sistema SIPROE, proporcionándole características de autonomía al no depender de la aplicación WEB para procesar eventos previamente programados por los usuarios en cada controlador Inteligente. El tiempo proporcionado por este controlador inteligente está disponible y es consultado por cada controlador a través del bus CAN. En la se muestra el diagrama de bloques del DS1337.

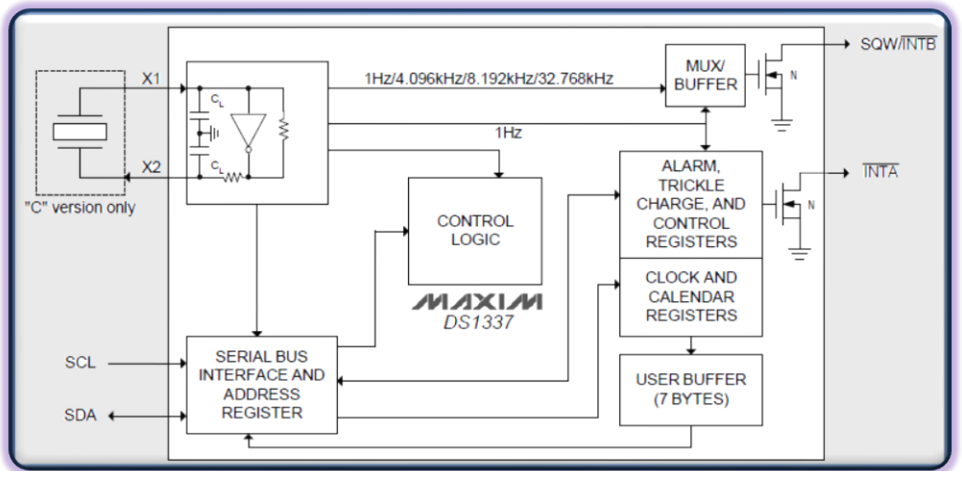


Figura 1.16 Diagrama en bloques del DS1337.

1.5.2.1 Características y programación del DS1337

El DS1337 es un reloj/calendario con una interfaz de comunicación serial I2C que proporciona un indicador de cambio de siglo, año, mes, fecha, día de la semana, hora, minuto y segundo contados ininterrumpidamente a partir de una fecha previamente programada. También tiene la capacidad de ser configurado como alarma, temporizador o generador de señales de onda cuadrada a diferentes frecuencias según la aplicación en la que se utilice. Esta característica es aprovechada para generar interrupciones en el µC cada minuto para enviar a través del bus CAN una trama broadcast a todos los Controladores conteniendo la fecha, una señal cuadrada de 1Hz alimenta un diodo led para indicación de su funcionamiento.

Para su programación, el DS1337 cuenta con dieciséis registros de configuración con direcciones desde 0x00 hasta 0x0F. Como se observa en la Tabla 1.16 los registros sirven tanto para configurar la hora y fecha inicial (0x00 a 0x06) como también para configurar eventos por medio de dos alarmas programables. La Tabla 1.17 muestra un resumen de los bytes de configuración del DS1337.

1.5.3 Características I2C de los Expansores de Entradas-Salidas Digitales

Como interface de expansión de Entradas-Salidas (E/S) digitales se ha incorporado a los Controladores inteligentes el circuito integrado MCP23016 el cual cuenta con dos grupos de E/S de propósito general cada uno de 8 bits y el circuito integrado MCP23008 que cuenta con un grupo E/S de propósito general de 8 bit.

Ambos circuitos integrados consisten de múltiples registros de configuración de entradas, salidas, selección de polaridad y configuración del pin de interrupción. El maestro del bus I^2C puede habilitar las E/S como entradas o salidas escribiendo los bits de configuración del registro de dirección E/S. Los datos de cada entrada o salida son mantenidos en el registro de retención correspondiente de E/S. Todos los registros pueden ser leídos por el maestro del sistema. La más importante característica es la salida de interrupción que es activada cuando algún estado de entrada difiere de su correspondiente estado en el registro de interrupción del puerto. Esto es utilizado para indicar al microcontrolador que un estado de entrada ha cambiado y poder realizar una acción producto de ese cambio, el registro de captura de interrupción captura el valor del puerto en el momento de generarse la interrupción y es utilizado para los cálculos de la lógica combinacional dentro del µC. Tres pines del dispositivo (A0 - A2) determinan la dirección física I2C y permite hasta ocho dispositivos para compartir el mismo bus I2C.

Tabla 1.16 Registros BCD de configuración del RTC DS1337

DIRECCIÓN	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCION	RANGO
00H	0	10 Segundos			Segundos				Segundo	00–59
01H	0	10 Minutos			Minutos				Minuto	00–59
02H	0	12/24	AM/PM	10 Horas	Horas				Hora	1–12+ AM/PM
			10 Horas							00–23
03H	0	0	0	0	0	Días			Día	1–7
04H	0	0	10 Fecha		Fecha				Fecha	01–31
05H	Centenario	0	0	10 Meses	Meses				Mes	01–12+ Centenario
06H	10 Años				Años				Año	00–99
07H	A1M1	10 Segundos			Segundos				Alarma 1 Segundo	00–59
08H	A1M2	10 Minutos			Minutos				Alarma 1 Minuto	00–59
09H	A1M3	12/24	AM/PM	10 Horas	Horas				Alarma 1 Hora	1–12+ AM/PM
			10 Horas							00–23
0AH	A1M4	DY/DT	10 Fecha		Día				Alarma 1 Día	1–7
					Fecha				Alarma 1 Fecha	01–31
0BH	A2M2	10 minutos			Minutos				Alarma 2 Minuto	00–59
0CH	A2M3	12/24	AM/PM	10 Horas	Horas				Alarma 2 Hora	1–12+ AM/PM
			10 Horas							00–23
0DH	A1M4	DY/DT	10 Fecha		Día				Alarma 2 Día	1–7
					Fecha				Alarma 2 Fecha	01–31
0EH	EOSC	0	0	RS2	RS1	INTCN	A2IE	A1IE	Control	---
0FH	OSF	0	0	0	0	0	A2F	A1F	Estado	---

Tabla 1.17 Registros de configuración de alarmas del RTC DS1337.

ALARM 1 BITS DE MASCARA					
DY/DT	**(BIT 7)**				
	A1M4	**A1M3**	**A1M2**	**A1M1**	**ALARMA**
X	1	1	1	1	Alarma una vez por segundo
X	1	1	1	0	Alarma cuando concuerdan los segundos
X	1	1	0	0	Alarma cuando minutos y segundos concuerdan
X	1	0	0	0	Alarma cuando hora, minutos y segundos concuerdan
0	0	0	0	0	Alarma cuando fecha, hora, minutos y segundos concuerdan
1	0	0	0	0	Alarma cuando día, hora, minutos y segundos concuerdan
ALARM 2 BITS DE MASCARA					
	A2M4	**A2M3**	**A2M2**		
X	1	1	1	-	Alarma una vez por minuto
X	1	1	0	-	Alarma cuando minutos concuerdan
X	1	0	0	-	Alarma cuando hora y minutos concuerdan
0	0	0	0	-	Alarma cuando fecha, hora y minutos concuerdan
1	0	0	0	-	Alarma cuando día, hora y minutos concuerdan

El diagrama de bloques y la distribución de pines del IC MCP23016 se muestran en la Figura 1.17 y Tabla 1.18 respectivamente. A continuación se describen los bloques funcionales más importantes de MCP23016.

El MCP23016 utiliza un circuito RC externo para determinar la velocidad interna de reloj. Se debe conectar un arreglo R y C como se muestra en la Figura 1.18. Un reloj de 1 MHz es necesario para que el dispositivo funcione apropiadamente y puede ser medido en el pin TP. Los valores recomendados de REXT y CEXT son mostrados en la Tabla 1.19.

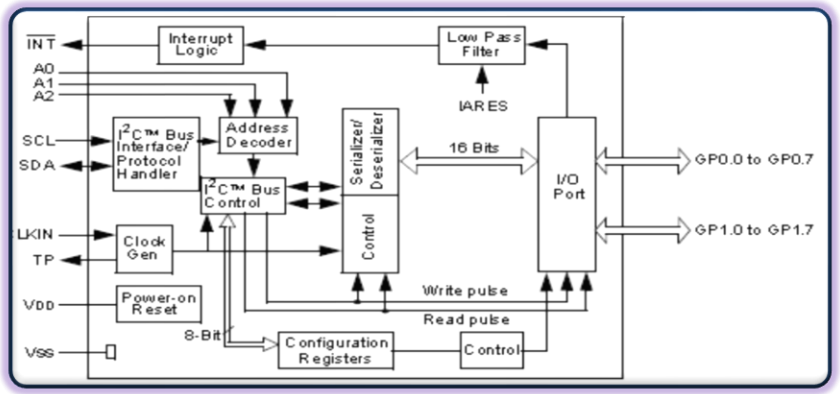


Figura 1.17 Diagrama de Bloques del MCP23016.

Tabla 1.18 Descripción de pines del MCP23016.

NOMBRE PIN	NO. PIN	TIPO	DESCRIPCIÓN	NOMBRE PIN	NO. PIN	TIPO	DESCRIPCIÓN
CLK	9	E	Entrada de Reloj	GP0.3	24	E/S	D3 E/S de GP0
TP	10	S	Prueba (flotante)	GP0.4	25	E/S	D4 E/S de GP0
GP1.0	2	E/S	D0 E/S de GP1	GP0.5	26	E/S	D5 E/S de GP0
GP1.1	3	E/S	D1 E/S de GP1	GP0.6	27	E/S	D6 E/S de GP0
GP1.2	4	E/S	D2 E/S de GP1	GP0.7	28	E/S	D7 E/S de GP0
GP1.3	5	E/S	D3 E/S de GP1	SCL	14	E	Entrada reloj serial
GP1.4	7	E/S	D4 E/S de GP1	SDA	15	E/S	Datos Serial E/S
GP1.5	11	E/S	D5 E/S de GP1	INT	6	S	Salida Interrupción
GP1.6	12	E/S	D6 E/S de GP1	A0	16	E	Entrada Dirección 1
GP1.7	13	E/S	D7 E/S de GP1	A1	17	E	Entrada Dirección 2
GP0.0	21	E/S	D0 E/S de GP0	A2	18	E	Entrada Dirección 3
GP0.1	22	E/S	D1 E/S de GP0	VSS	1, 8, 19	P	Referencia tierra
GP0.2	23	E/S	D2 E/S de GP0	VDD	20	P	Fuente Positiva

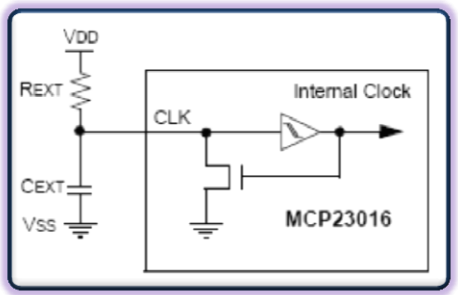


Figura 1.18 Configuración externa del reloj.

Tabla 1.19 Valores recomendados de REXT y CEXT.

R_{EXT}	C_{EXT}
3.9 kΩ	33 pF

El MCP23016 soporta los comandos mostrados en la Tabla 1.20 y el MCP23008 los comandos de la Tabla 1.21.

Tabla 1.20 Comandos MCP23016.

BYTE COMANDO	RESULTADO
00h	Acceso a GP0
01h	Acceso a GP1
02h	Acceso a OLAT0
03h	Acceso a OLAT1
04h	Acceso a IPOL0
05h	Acceso a IPOL1
06h	Acceso a IODIR0
07h	Acceso a IODIR1
08h	Acceso a INTCAP0 (solo lectura)
09h	Acceso a INTCAP1 (solo lectura)
0Ah	Acceso a IOCON0
0Bh	Acceso a IOCON1

Tabla 1.21 Comandos MCP23008.

BYTE COMANDO	RESULTADO
00h	Acceso a IODIR
01h	Acceso a IPOL
02h	Acceso a GPINTEN
03h	Acceso a DEFVAL
04h	Acceso a INTCON
05h	Acceso a IOCON
06h	Acceso a GPPU
07h	Acceso a INTF
08h	Acceso a INTCAP (solo lectura)
09h	Acceso a GPIO
0Ah	Acceso a OLAT

El diagrama de bloques y la distribución de pines del IC MCP23008 se muestran en la Figura 1.19 y la Tabla 1.22 respectivamente.

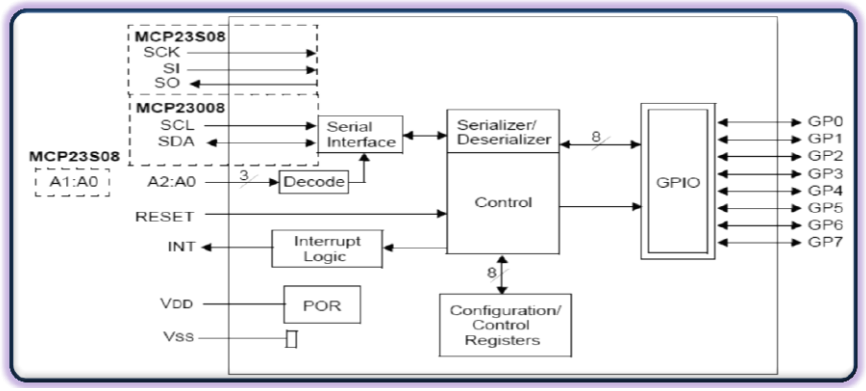


Figura 1.19 Diagrama de Bloques del MCP23008.

Tabla 1.22 Descripción de pines del MCP23008.

Nombre Pin	No. Pin	Tipo	Descripción
SCL	1	E	Entrada de Reloj serial
SDA	2	E/S	Datos E/S serial
A2	3	E	Entrada 3 dirección física
A1	4	E	Entrada 2 dirección física
A0	5	E	Entrada 1 dirección física
$\overline{\text{RESET}}$	6	E	Entrada de Reinicio Externa
NC	7	--	No Conectado
INT	8	S	Salida de Interrupción
V_{SS}	9	P	Tierra
GP0	10	E/S	Pin bidireccional E/S
GP1	11	E/S	Pin bidireccional E/S
GP2	12	E/S	Pin bidireccional E/S
GP3	13	E/S	Pin bidireccional E/S
GP4	14	E/S	Pin bidireccional E/S
GP5	15	E/S	Pin bidireccional E/S
GP6	16	E/S	Pin bidireccional E/S
GP7	17	E/S	Pin bidireccional E/S
V_{DD}	18	P	Potencia

1.5.4 Características I2C de los Expansores de Entradas-Salidas Análogas

El prototipo SIPROE cuenta con dos interfaces analógicas. Una interface de 8 entradas basada en el convertidor análogo a digital del fabricante MAXIM MAX127 y una interface de 8 salidas basada en el convertidor digital a análogo MAX521.

El IC expansor de entradas análogas MAX127 cuenta con 8 entradas analógicas y una interface I2C a través de la cual se comunica con los µC PIC18F258 embebidos en los controladores inteligentes. La Figura 1.20 y la Tabla 1.23, muestran respectivamente el diagrama de bloques y la distribución de pines del MAX127.

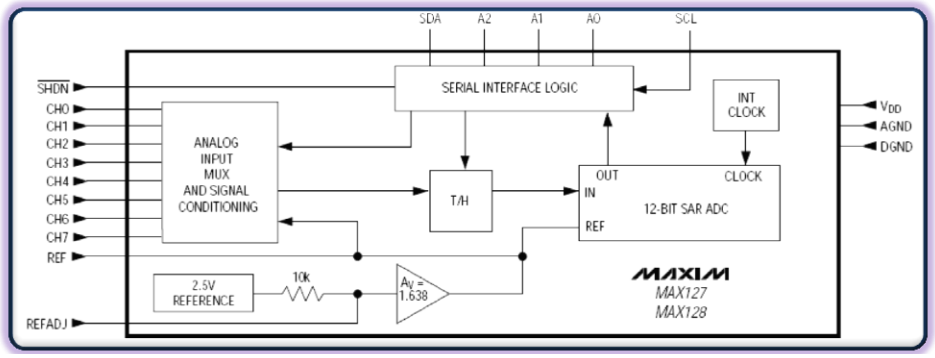


Figura 1.20 Diagrama de Bloques funcional del MAX127.

Tabla 1.23 Descripción de pines del MAX127.

PIN	NOMBRE	FUNCION
1,2	VDD	Fuente +5v
3,9,22,24	N.C.	No conectado
4	DGND	Tierra Digital
5	SCL	Entrada de Reloj Serial
6,8,10	A0,A2,A1	Entradas de dirección
7	SDA	Datos E/S seriales
11	$\overline{\text{SHDN}}$	Entrada de Apagado
12	AGND	Tierra Análoga
13-20	CH0-CH7	Canales análogos de entrada
21	REFADJ	Salida Referencia de voltaje /ajuste externo
23	REF	Referencia Búfer salida/ Entrada Referencia ADC

El MAX127 es un sistema para adquisición de datos (DAS) con una resolución de 12 bits en cada una de las 8 entradas analógicas. Tres pines A0, A1 y A2 son utilizados para programar la dirección de hardware, permitiendo el uso de hasta ocho MAX127

conectados al bus I2C sin hardware adicional. La dirección, el control y los datos de las conversiones son transferidos de forma serie por el bus I2C. Las características del dispositivo incluyen multiplexación de entradas análogas, rangos de voltaje de entrada programable por software de hasta ±10 V, tasa de muestreo de 8ksps (miles de muestras por segundo), voltaje de operación de 5 V, dos modos de operación a baja potencia, referencia de voltaje de 4.096 V y tiempo de conversión típico de 7.7µs; provee adicionalmente una interface con capacidad de manejar directamente sensores con valores nominales de hasta ±12V@20mA en un sistema de +5V como SIPROE. La Tabla 1.24 resume las características eléctricas del MAX127.

Tabla 1.24 Resumen de Características eléctricas del MAX 127.

CARACTERISTICA ELECTRICA	DESCRIPCION	VALORES MAXIMOS NOMINALES ABSOLUTOS
V_{DD} a AGND	Voltaje alimentación a tierra análoga	-0.3V a 6.0V
AGND a DGND	Voltaje entre tierra análoga y digital	-0.3V a +0.3V
CNL0-CNL7 a AGND	Voltaje de entrada del canal	±16.5 V
REF a AGND	Voltaje entre Referencia y tierra análoga	-0.3 a (VDD + 0.3V)
REFADJ a AGND	Voltaje entre Referencia ajustable y tierra análoga	-0.3 a (VDD + 0.3V)
A0, A1, A2 a DGND	Voltaje entre pines de dirección y tierra digital	-0.3 a (VDD + 0.3V)
$\overline{SHDN}$, SCL, SDA a DGND	Voltaje entre los pines de I2C y control y tierra digital	-0.3V a + 6.0V
Imáx	Máxima corriente en cualquier pin	-0.3 a (VDD + 0.3V)
CPD @ +70°C	Disipación continua de Potencia	762mW
OTR	Rangos de temperatura de Operación	0°C A +70°C
THD	Distorsión armónica total	-87dB
T/H AT	Tiempo de adquisición Track/Hold	3µs
REFOUT V	Salida de referencia de voltaje	4.096±0.02V
OSCC	Corriente de cortocircuito de salida	30mA

En la Figura 1.21 se muestra el circuito de operación típico del MAX127. Cada uno es accedido enviando una dirección válida al dispositivo. La dirección consiste de una parte fija (0101) y una parte programable. La parte programable es configurada con los pines A2, A1 y A0. El último bit del byte de dirección es el bit de Lectura/Escritura que determina la dirección de los datos a transferirse. La Figura 1.22 muestra la estructura del byte de dirección.

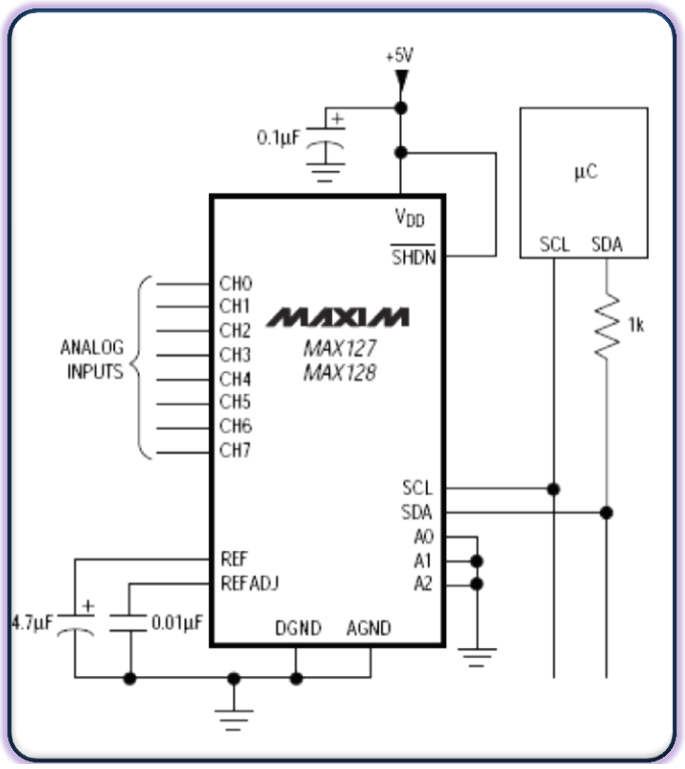


Figura 1.21 Circuito de operación típico MAX127.

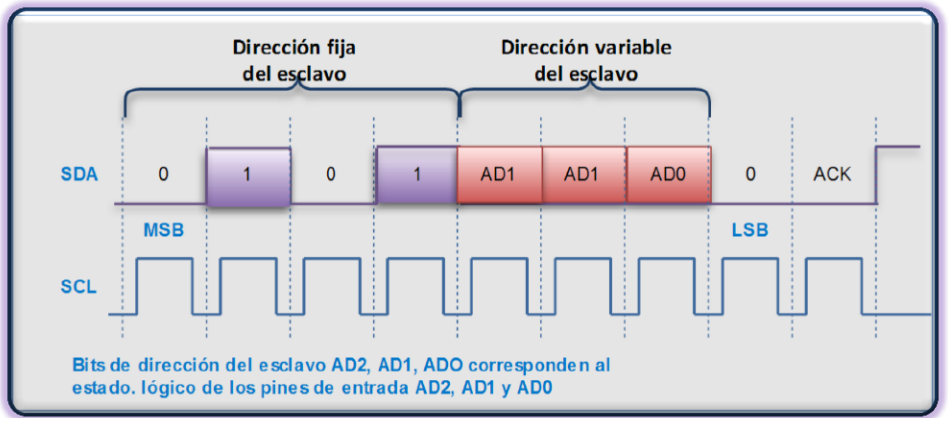


Figura 1.22 Byte de dirección MAX 127.

El segundo byte enviado al dispositivo será almacenado en su registro del control y es requerido para controlar la función de dispositivo. Ver Figura 1.23 para mayor detalle de la programación del byte de control del MAX127.

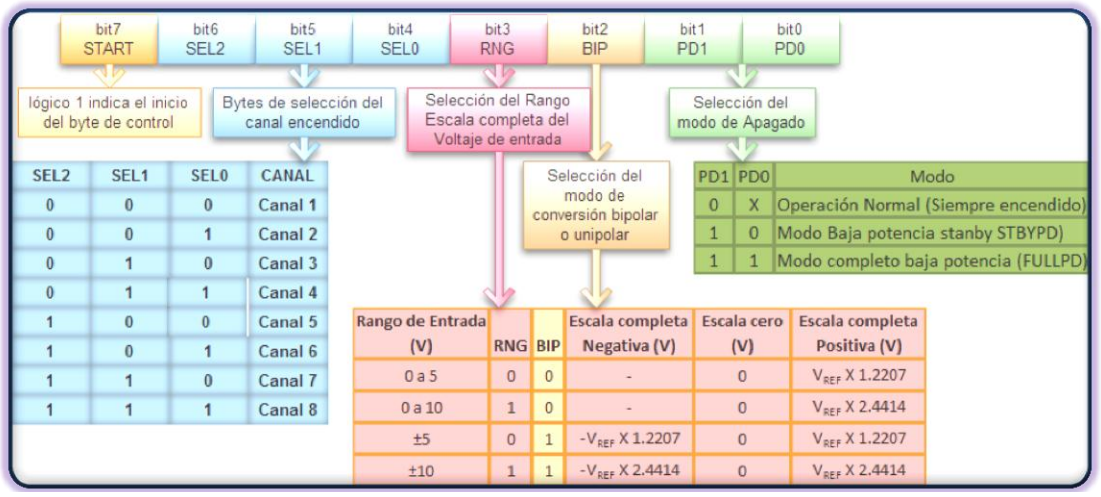


Figura 1.23 Byte de Control MAX 127.

Para conservar el consumo de Potencia el MAX127 debe configurarse en uno de los siguientes modos de baja Potencia.

- Baja Potencia en espera (STBYPD)
- Baja Potencia Completo (FULLPD)

En todos los modos de baja potencia la interface de comunicación permanece activa y las conversiones de señales pueden leerse. El pin $\overline{SHDN}$, sirve para configurar el modo de Baja potencia completo.

El MAX521 es un expansor I2C de 8 salidas análogas, con una resolución de 8 bits cada una y una interface I2C a través de la cual se comunica con los µC PIC18Fxx8 embebidos en los Controladores inteligentes. De las características más importantes de las cuales fue producto su elección se menciona su bajo consumo de corriente, salidas controladas con búfer, modos de bajo consumo de potencia programables y referencias de voltaje de salida independientes. La Figura 1.24, muestra el diagrama de bloques funcional del MAX521.

El circuito de operación típico para la implementación de varios dispositivos MAX521 en un bus se muestra en la Figura 1.25 y en la Tabla 1.23 la distribución de pines en el circuito integrado. El byte de dirección, como se observa en la Figura 1.26, consiste de una parte fija (01010) y una parte programable. Dos pines A0, A1 son utilizados para programar la dirección de hardware, permitiendo la utilización de hasta cuatro dispositivos (32 canales análogos de salida) conectados al bus I^2C sin hardware adicional.

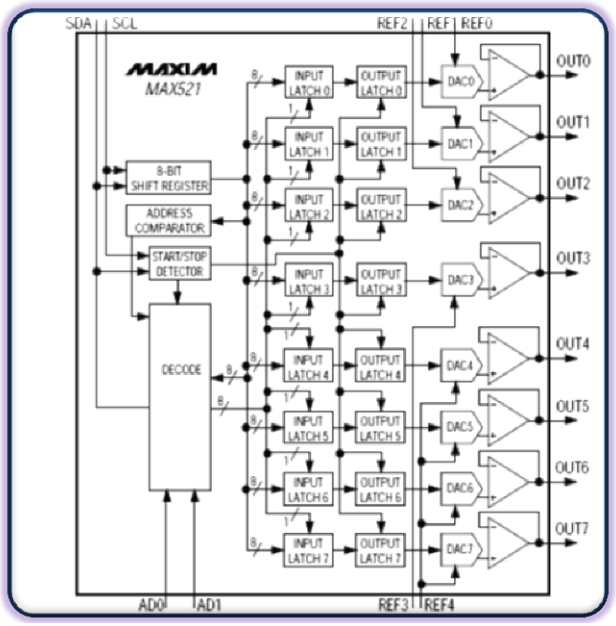


Figura 1.24 Diagrama de Bloques funcional del MAX521.

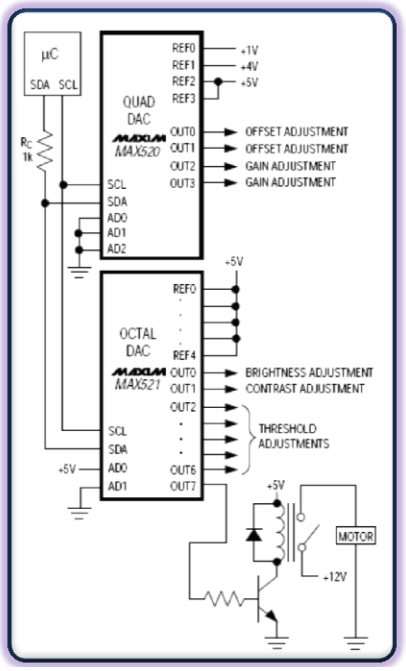


Figura 1.25 Circuito de operación típico MAX521.

Tabla 1.25 Descripción de pines del MAX521.

PIN	NOMBRE	FUNCION
1	**OUT1**	Voltaje de salida DAC1
2	**OUT0**	Voltaje de salida DAC0
3	**REF1**	Voltaje de Referencia DAC1
4	**REF0**	Voltaje de Referencia DAC0
5	**DGND**	Tierra Digital
6	**AGND**	Tierra Análoga
7	**SCL**	Entrada Reloj Serial
8	**SDA**	Entrada Datos Serial
9	**OUT4**	Voltaje de salida DAC4
10	**OUT5**	Voltaje de salida DAC5
11	**OUT6**	Voltaje de salida DAC6
12	**OUT7**	Voltaje de salida DAC7
13	**AD0**	Entrada Dirección 0
14	**AD1**	Entrada Dirección 1
15	**V_{DD}**	Fuente Poder, +5V
16	**REF4**	Voltaje de Referencia DAC 4, 5, 6, 7
17	**REF3**	Voltaje de Referencia DAC3
18	**REF2**	Voltaje de Referencia DAC4
19	**OUT3**	Voltaje de salida DAC3
20	**OUT2**	Voltaje de salida DAC2

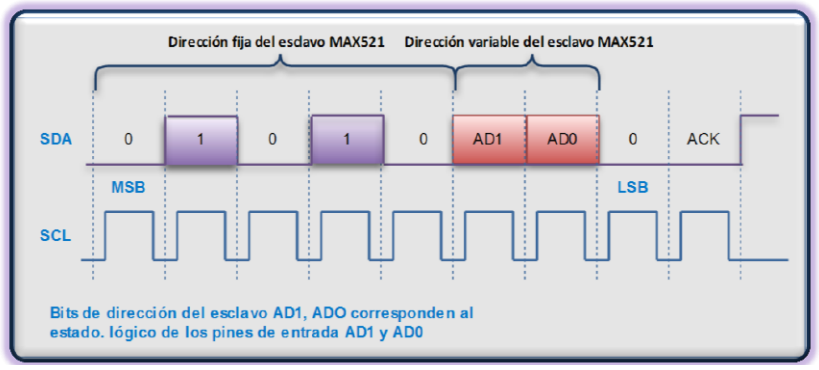


Figura 1.26 Byte de dirección MAX521.

El segundo byte enviado al dispositivo es almacenado en su registro de comando mostrado en la Figura 1.27, el cual es seguido por un byte de salida; los bits A0 y A1 indican la dirección digital del DAC cuyo latch de datos de entrada recibe el valor que será convertido en la salida análoga.

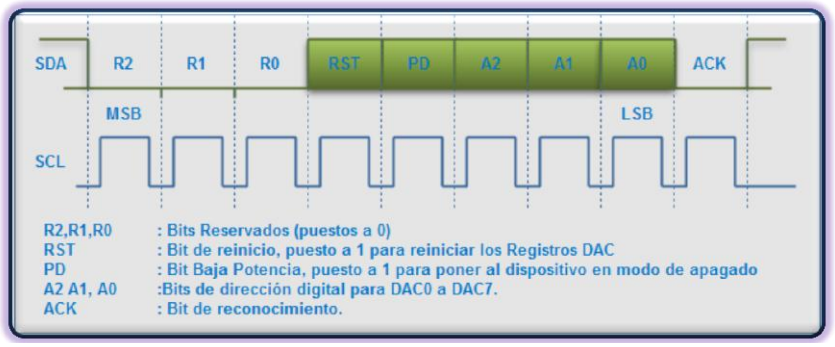


Figura 1.27 Registro de Comando MAX 521.

El voltaje de salida de cada DAC puede ser representado por una fuente de voltaje V_{OUT} digitalmente programable. Como es mostrada por la Figura 1.28

$$V_{OUT} = \frac{(N \times V_{REF})}{256} \text{V}$$

Figura 1.28 Voltaje de salida del DAC

Donde N es el valor numérico del código binario de entrada dado por el byte de salida y Vref = 5 voltios. La Tabla 1.26 muestra el código unipolar del byte de salida.

Tabla 1.26 Código Unipolar del MAX521.

PIN	FUNCION
11111111	$+V_{REF}\left(\frac{255}{256}\right)$
10000001	$+V_{REF}\left(\frac{129}{256}\right)$
10000000	$+V_{REF}\left(\frac{128}{256}\right) = \frac{V_{REF}}{2}$
01111111	$+V_{REF}\left(\frac{127}{256}\right)$
00000001	$+V_{REF}\left(\frac{1}{256}\right)$
00000000	0V

1.6 La Gama de Microcontroladores PIC 18Fxx8

El componente medular de cada Controlador Inteligente es el µC del fabricante MICROCHIP pertenecientes a la gama PIC18Fxx8, que cuentan con una memoria flash mejorada tanto en la manera de acceder a esta como en su rendimiento. La Tabla 1.27 y la Figura 1.29 muestran las características y la distribución de pines, respectivamente de los dos µC utilizados, el PIC18F258 y el PIC18F458.

Tabla 1.27 Características de los µC utilizados en SIPROE

DISPOSITIVO	MEMORIA DE PROGRAMA		MEMORIA DE DATOS		E/S	MSSP		USART	TEMPORIZADORES
	FLASH	INSTRUCCIONES 1 PALABRA	SRAM	EEPROM		SPI	I2C		
PIC18F258	32K	16384	1536	256	22	SI	SI	SI	1/3
PIC18F458	32K	16384	1536	256	33	SI	SI	SI	1/3
CARACTERISTICAS DE SUS PERIFERICOS									
• Alta corriente de drenaje 25mA • Tres pines de interrupciones externas • Temporizadores de 16 y 8 bits • Puerto maestro síncrono serial, dos modos de operación: PSI e I2C • Modulo USART direccionable									
CARACTERISTICAS CAN									
• Tasa de bit de hasta 1Mbps • Cumple con la especificación CAN2.0B: • Identificador 29 bit • Longitud de mensaje de 8 bytes • 3 búfer de transmisión con priorización • 2 búfer de recepción. • 6 filtros de aceptación de 29bits • Manejo avanzado de errores									
CARACTERISTICAS ESPECIALES									
• Reset de encendido • Temporización de encendido • Temporizador Perro Guardián • Protección de código programable • Memoria Flash mejorada de alta velocidad y baja potencia.									
CARACTERISTICAS DE CPU									
• Memoria de programa lineal direccionable hasta 2Mbyte • Memoria de datos lineal direccionamiento hasta 4Kbyte • Hasta 10 MIPS • Oscilador de 4-10 MHz • Instrucciones de 16bits • Niveles de prioridad									

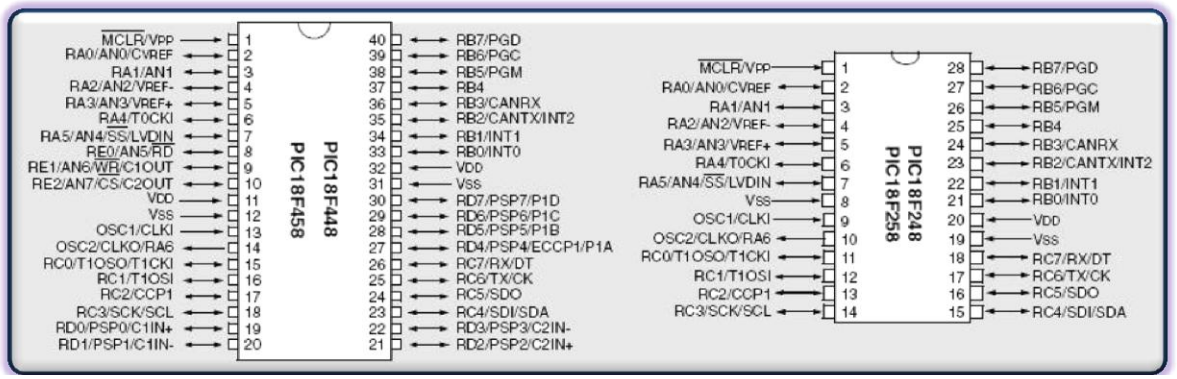


Figura 1.29 Distribución de pines de los µC PIC18Fxx8

Para desarrollar un método estandarizado del manejo de peticiones modbus, de comunicaciones CAN y de comunicación I2C, la memoria de los µC ha sido reestructurada como lo describe la Tabla 1.29. En tal sentido cada rutina desarrollada en el firmware para el manejo y atención de las funciones modbus enviadas desde la aplicación Web sabe que parte de la memoria (Registro de retención según la especificación modbus) debe leer para devolver la respuesta.

Como se observa en la Tabla 1.29, las peticiones modbus son recibidas al inicio de la memoria del programa a partir de la posición 0x0000 para las funciones 2, 3, 5 y como caso especial a partir de la posición 0x0008 para la función 16, debido principalmente a que la petición de esta función es más grande que su respuesta, caso contrario para las demás funciones. En el código del firmware cada función modbus está relacionada a variables que tienen un significado especifico a la función pero que ocupan la misma posición de memoria, así las variables FUNC_CODE2 y FUNC_CODE3 identifican claramente qué función se está procesando aunque ocupan la misma posición en memoria gracias a la funcionalidad del µC que permite traslapar bloques de memoria que son excluyentes entre sí; en el caso particular de modbus bajo ninguna circunstancia se procesan dos funciones diferentes a la vez.

Adicionalmente se ha reservado memoria para cada una de las entradas digitales, específicamente existen 4 bytes que permiten direccionar hasta 32 entradas digitales; para las salidas digitales se ha reservado el doble de bytes por dos razones importantes, la primera por contar con relés de doble bobina para set-reset y la segunda porque cada salida digital cuenta con entradas de retroalimentación que permiten corroborar el estado de las mismas. Para las entradas análogas se ha reservado espacio en memoria para manejar hasta 16 entradas cada una tiene reservado seis bytes para máximo, mínimo y valor actual. Las salidas análogas tienen reservada memoria para manejar hasta 16 salidas con resolución de 12 bits o 32 salidas con resolución de 8 bits.

El espacio reservado para cada uno de los dieciséis eventos que son atendidos por cada controlador inteligente está compuesto de tres partes, la primera, un espacio reservado de cinco bytes para la fecha, hora de inicio, la segunda, un byte caracterizando la

repetición del evento, y la tercera, un espacio de dos bytes para la hora y minutos de duración de los eventos contados a partir del inicio del día en formato de 24 horas; restricción es por tanto, que un evento en particular no puede durar más de un día, si esto es requerido debe configurarse el byte de repetición como se describe en la Tabla 1.28.

Debido a que el manejo de las cargas es configurable por el usuario, cada evento tiene reservado espacios dentro de la memoria del μC para indicar cuantas entradas-salidas digitales y análogas están asociadas a un evento en particular; en las secciones de memoria, LOGICAL_ID_VAR, LOGICAL_OD_VAR, LLOGICAL_IA_VAR y LOGICAL_OA_VAR es guardada la información relacionada con la asociación de las entradas y salidas a un evento particular. La configuración de usuario con la que cuenta SIPROE es básica, resumida a ecuaciones lógicas AND u OR que relacionan variables físicas discretas y continuas para actuar sobre las salidas disponibles. Las secciones de memoria LOGICAL_OR_VAR y LOGICAL_ADN_VAR están reservadas para guardar estas ecuaciones lógicas.

Tabla 1.28 Configuración del byte de repetición de eventos

BYTE DE REPETICION DEL EVENTO		TIPO DE REPETICION
NIBLE ALTO	**NIBLE BAJO**	
0000	**0000**	Ejecuta el evento una vez
0000	**0001**	Ejecuta el evento de Lunes a Viernes
0000	**0010**	Ejecuta el evento Lunes, Miércoles y Viernes
0000	**0100**	Ejecuta el evento Martes y Jueves
0000	**1000**	Reservado a posibles aplicaciones
0001	**XXXX**	Ejecuta el evento semanalmente
0010	**XXXX**	Ejecuta el evento mensualmente

Tabla 1.29 Mapeo de memoria de los µC PIC18Fxx8 en SIPROE.

SECCIÓN DE MEMORIA	TIPO DE SECCION	DIRECCION INICIAL	UBICACIÓN	TAMAÑO (BYTES)	DESCRIPCION
MEMORIA PETICION RESPUESTA MODBUS					
MODBUS_PETICION	Udata	0x000000	data	0x000008	Trama de petición modbus(exc. F16)
MODBUS_RESP	Udata	0x000008	data	0x000029	Trama de respuesta modbus(exc. F16)
MEMORIA COMUN I2C-MODBUS					
EXC_MB_E_I2C	Udata	0x000032	data	0x000009	Trama de excepción modbus
MEMORIA COMUN CAN, TIEMPO, MODBUS, I2C					
EXTRA_VAR	Udata	0x00003b	data	0x00000f	Variables temporales
CAN_TMP	Udata	0x00004a	data	0x000004	Variables temporales CAN
TIEMPO	Udata	0x00004e	data	0x000003	Variables de librería de Tiempo
CAN_ID	Udata	0x000062	data	0x000005	ID CAN de controladores
MODBUS_VAR	Udata	0x000067	data	0x00000c	Variables temporales modbus
I2C_VAR	Udata	0x000073	data	0x000019	Variables temporales I2C
.udata_ovr	Udata	0x000095	data	0x00001f	
CAN_BUFFER	Udata	0x0000b4	data	0x00004c	Buffer Tx y Rx CAN
ENTRADAS DIGITALES (32)					
INPUT_DIG_VAR	Idata	0x000100	data	0x000004	Entradas Digitales
SALIDAS DIGITALES (64)					
OUTPUT_DIG_VAR	Idata	0x000104	data	0x000008	Salidas Digitales
ENTRADAS ANALOGAS (16)					
INPUT_ANA_VAR	Idata	0x00010c	data	0x000064	Entradas Análogas
EVENTOS (16)					
EVENTO_VAR	Idata	0x000170	data	0x00008e	Eventos
ENTRADAS ANALOGAS (16)					
OUTPUT_ANA_VAR	Idata	0x000210	data	0x000020	Salidas Análogas
LOGICA COMBINACIONAL (16 EVENTOS X 16BYTES)					
LOGICAL_ID_VAR	Idata	0x000230	data	0x000020	Entradas Digitales asociadas a evento
LOGICAL_IA_VAR	Idata	0x000250	data	0x000020	Entradas Análogas asociadas a evento
LOGICAL_OD_VAR	Idata	0x000270	data	0x000020	Salidas Digitales asociadas a evento
LOGICAL_OA_VAR	Idata	0x000290	data	0x000020	Salidas Análogas asociadas a evento
LOGICAL_OR_VAR	Idata	0x0002b0	data	0x000040	Lógica OR asociada a evento
LOGICAL_AND_VAR	Idata	0x000300	data	0x000040	Lógica AND asociada a evento

1.7 Modelo de Aplicación de JAVA EE

El modelo de Aplicación JAVA EE está fundamentado en el lenguaje de programación Java y la maquina virtual JAVA (JVM). Las bases del modelo de aplicación la conforman la portabilidad, la seguridad y la productividad en el desarrollo. El modelo de aplicación Java EE define una arquitectura para implementar servicios como aplicaciones multicapas que dan la escalabilidad, accesibilidad y administración necesaria en las aplicaciones al nivel empresarial. Este modelo particiona el trabajo necesario para implementar un servicio multicapas en dos: La lógica de presentación y negocio implementada por el desarrollador y el sistema estándar de servicios proporcionado por la plataforma Java EE.

1.7.1 Aplicaciones distribuidas multicapas

La plataforma Java EE utiliza un modelo de aplicación distribuida multicapas para aplicaciones empresariales. La lógica de aplicación es dividida en componentes de acuerdo a la función, los diferentes componentes de aplicación que componen una aplicación Java EE son instalados en diferentes maquinas dependiendo de la capa en el ambiente multicapas a la cual el componente de aplicación pertenece. La Figura 1.30 muestra dos aplicaciones Java EE multicapas divididas en las capas siguientes:

- Componentes de la capa Cliente o de Presentación, corriendo en la maquina cliente.
- Componentes de la capa de Red “Web Tier” o Capa de Domino, corriendo en el Servidor Java EE.
- Componentes de la capa de Negocio “Business Tier” corriendo en el Servidor Java EE.
- Software de la capa del Sistema de Información Empresarial (EIS) corriendo en el Servidor EIS.

Aunque una aplicación Java EE puede consistir de las tres o cuatro capas mostradas en la Figura 1.30, las aplicaciones Java EE multicapas son consideradas generalmente aplicaciones de tres capas ya que están distribuidas sobre tres ubicaciones: las maquinas clientes (Capa de Presentación), la maquina servidor Java EE (Capas de dominio y Negocio) y las maquinas de bases de datos (Capa de datos). Las aplicaciones de tres capas que corren de esta forma extienden del modelo estándar cliente-servidor colocando un servidor de aplicación multi-hilos entre la aplicación cliente y el almacenamiento final.

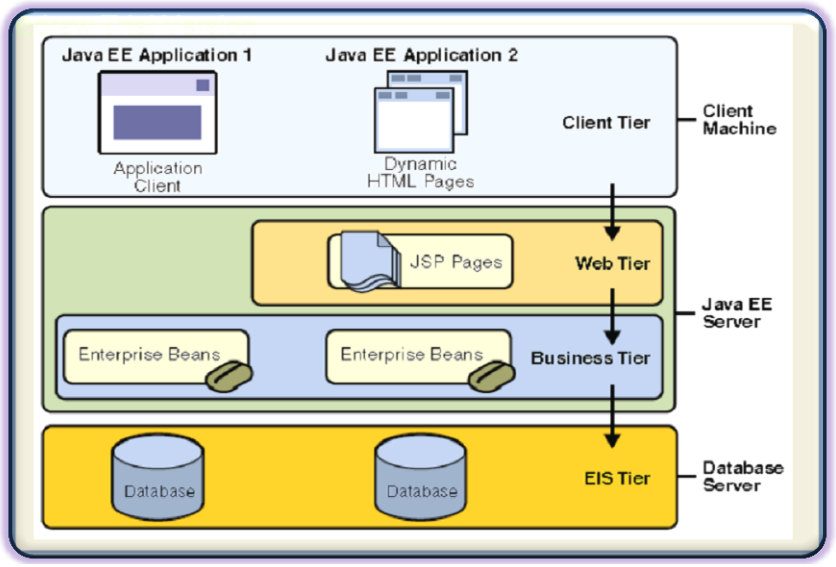


Figura 1.30 Aplicaciones Multicapas.

1.7.2 Componentes java EE

Las aplicaciones Java EE están formadas de componentes. Un componente Java EE es una unidad funcional de software auto contenida que es ensamblada dentro de una aplicación Java EE con sus clases relacionadas y archivos comunicándose con otros componentes.

La especificación Java EE define los siguientes Componentes:

- Aplicaciones Clientes y Applets son componentes que corren en el cliente.
- Componentes de tecnología Java Servlet, Java Server Faces y Java Server Pages (JSP) que son componentes de Red que corren en el servidor.
- Componentes Enterprise JavaBeans (EJB) que son componentes de negocio que corren en el servidor.

Estos Componentes son escritos en lenguaje Java y son compilados de la misma forma que cualquier otra clase Java la diferencia está en que los componentes Java EE son ensamblados dentro de una aplicación Java EE y son utilizados para producción, donde corren y son manejados por el servidor Java EE.

1.7.3 Clientes JAVA EE

Un cliente Java EE puede ser un cliente de red o una aplicación cliente. Un cliente de red consiste de dos partes: (1) Páginas Web Dinámicas conteniendo varios tipos de lenguajes de etiquetas (HTML, XML, etc.) las cuales son generadas por componentes de Red corriendo en la capa de Red y (2) Un Buscador de Red (Browser) el cual presenta las paginas recibidas del servidor. Normalmente es llamado un cliente débil el cual no ejecuta consultas a bases de datos, ni ejecuta reglas de negocio complejas.

Las operaciones más complejas son descargadas a los "Enterprise beans" ejecutándose en el Servidor Java EE, donde pueden llevar a cabo la seguridad, velocidad, servicios y confiabilidad de la tecnología proporcionada al lado del servidor. Una página Web recibida desde la capa de red puede incluir un Applet embebido. Un Applet es una pequeña aplicación cliente escrita en Java que se ejecuta en la JVM instalada en el buscador de Red.

Una aplicación cliente proporciona la forma en que los usuarios manejarán las tareas que requieren una interface enriquecida proporcionada por el lenguaje xml. Las aplicaciones cliente acceden directamente a los "Enterprise beans" corriendo en la capa de negocio. Sin embargo, si los requerimientos de la aplicación lo garantizan, una aplicación cliente puede abrir una conexión http para establecer comunicación con un servlet. En SIPROE la aplicación cliente desarrollada para llevar a cabo peticiones al servidor modbus se denomina **Aplicativo Intermedio**, y se encarga básicamente del manejo del protocolo Modbus sobre TCP/IP.

1.7.4 Arquitectura de Componentes JAVABEANS

Las capas de Servidor y Cliente incluyen componentes basados en la Arquitectura "JavaBeans" para manejar el flujo de datos entre una aplicación cliente o applet y los componentes corriendo en el Servidor Java EE, o entre los componentes de Servidor y una base de datos. Los componentes "JavaBeans" no son considerados componentes Java EE. Un JavaBeans componente (EJB) o Enterprise bean es un cuerpo de código teniendo campos y métodos para implementar módulos de lógica de negocio en el servidor JavaEE.

Existen dos tipos de enterprise beans: (1) "session beans" y (2) "message-driven beans". Un session bean representa una conversación transitoria con un cliente que finaliza junto con los datos asociados cuando el cliente termina la conversación. Un message-driven bean combina características de un session bean y un escuchador de mensajes, permitiendo a un componente de negocio recibir mensajes asíncronos. La Figura 1.31 muestra varios de los elementos que pueden componer la capa Cliente (Client Tier).

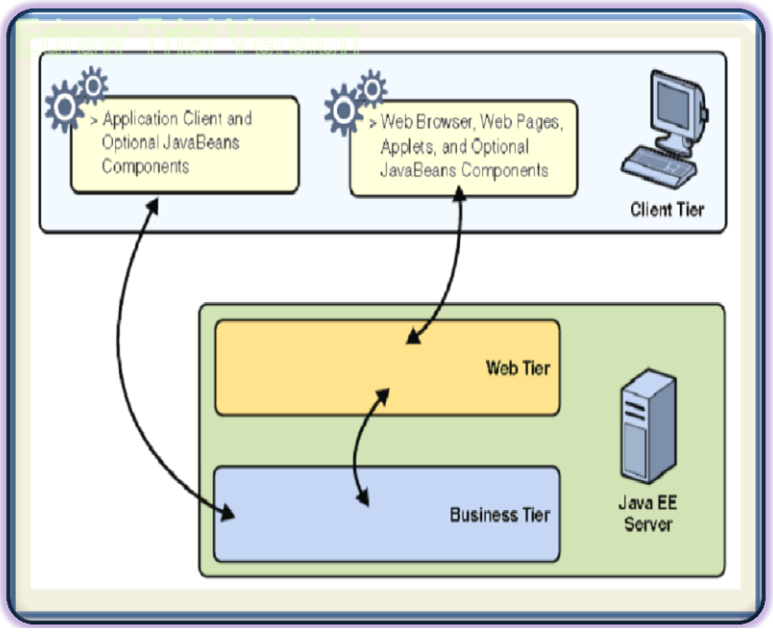


Figura 1.31 Comunicación con el Servidor de la capa cliente en J2EE.

El cliente se comunica con la capa de negocio (Business Tier) corriendo en el servidor Java EE directamente o en el caso de un cliente corriendo en un buscador, a través de páginas JSP o Servlets corriendo en la capa de red (Web Tier). La aplicación Java EE utiliza un cliente débil sobre un buscador, o una aplicación cliente fuerte. Con un cliente fuerte se mantiene la funcionalidad en el cliente y cercana al usuario, con uno débil se descarga la funcionalidad tanto como sea posible en el servidor. Entre mayor la funcionalidad descargada en el servidor más fácil es la distribución, utilización y manejo de la aplicación.

Los componentes de red son Servlet o paginas creadas usando tecnología JSP o tecnología Java Server Faces. Los Servlets son clases en lenguaje Java que dinámicamente procesan peticiones y construyen respuestas. Las paginas JSP son documentos texto que se ejecutan como servlet pero permiten una propuesta más natural para crear contenido estático. Las paginas estáticas HTML son agregadas a los componentes de Red durante el ensamblado de la aplicación pero no son consideradas componentes de Red. Las clases de Utilidad al lado del servidor también son agregadas a los componentes de Red pero tampoco se consideran componentes de Red.

La Figura 1.32**,** muestra la capa de red así como la de cliente podría incluir un componente “JavaBeans” para manejar las entradas de usuario y enviar esa entrada a los “Enterprise beans” corriendo en la capa de negocio para ser procesadas.

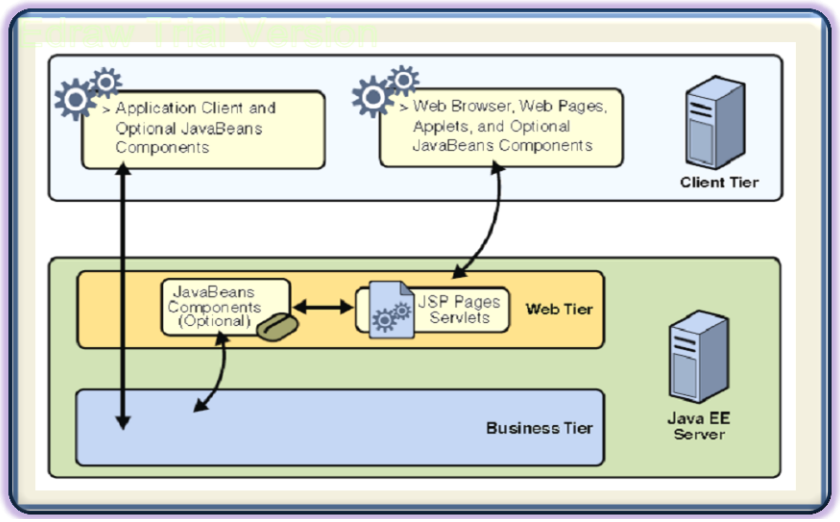


Figura 1.32 La aplicación Java EE y la Capa de Red.

El código de negocio, el cual es la lógica que resuelve o satisface las necesidades de un dominio de negocio en particular es manejado por "Enterprise beans" corriendo en la capa de negocio. La Figura 1.33 muestra como un "Enterprise beans" recibe datos desde los programas cliente, los procesa si es necesario y los envía a la capa EIS para almacenarlos. Un "Enterprise bean" también recupera datos desde el almacenamiento, los procesa si es necesario y los envía de regreso al programa cliente, en dicho caso la entidad beans es reemplazada por una entidad de la API de persistencia Java o Hibernate.

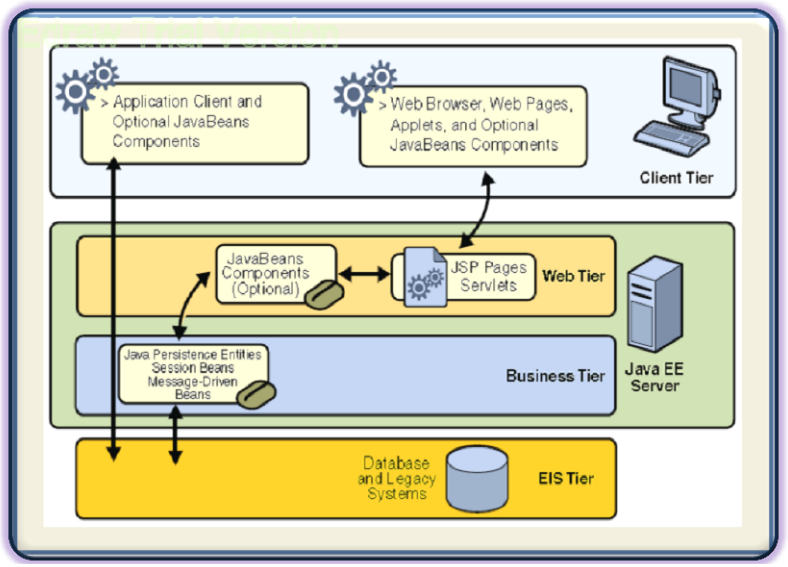


Figura 1.33 Capas de Negocio y Red de J2EE.

La capa del Sistema de Información Empresarial maneja el software EIS y los sistemas de infraestructura empresarial incluidos tales como el planeamiento de recursos empresariales (ERP), La unidad central de procesamiento de transacciones, los sistemas de bases de datos y otros sistemas de información heredados.

1.7.5 Contenedores JAVA EE

La arquitectura Java EE basada en componentes e independiente de la plataforma hace más fácil desarrollar las aplicaciones ya que la lógica de Negocio está organizada en componentes reutilizables. Además, el servidor Java EE proporciona servicios esenciales en forma de un contenedor para cada tipo de componente.

Los Contenedores son la interfaz entre un componente y la funcionalidad de bajo nivel de la plataforma específica que soporta el componente. Antes que un componente de Red, Enterprise Bean o aplicación cliente pueda ser ejecutado, este debe ser ensamblado dentro de un modulo Java EE y utilizado dentro de su contenedor. El proceso de ensamblado involucra la configuración específica del contenedor de cada componente de la aplicación Java EE y para la aplicación misma. La configuración del contenedor produce el soporte esencial dado por el Servidor Java EE, incluyendo servicios de seguridad, manejo de transacción y conectividad remota. Debido a que la arquitectura Java EE proporciona servicios configurables, componentes dentro de una misma aplicación pueden comportarse diferente dependiendo de donde están siendo utilizados. El contenedor también maneja servicios no configurables tales como ciclos de vida de servlets y Enterprise bean, conexión a recursos compartidos de bases de datos, persistencia de datos y acceso a las APIs de la plataforma Java EE.

1.7.5.1 Tipos de contenedores

El proceso de despliegue instala componentes de aplicación Java EE en los contenedores como se muestra en la Figura 1.34, la Tabla 1.30 describe los diferentes tipos de contenedores dentro de Java EE**.**

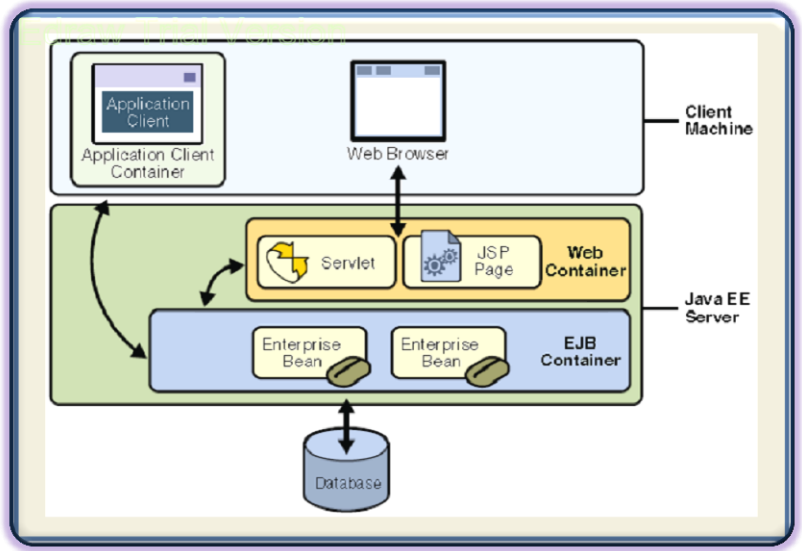


Figura 1.34 Servidor Java EE y los Contenedores.

Tabla 1.30 Contenedores Java EE

CONTENEDOR	DESCRIPCION
Servidor Java EE	La porción en tiempo de ejecución de un producto Java EE. Un Servidor Java proporciona contenedores de Red y EJB.
Contenedor Enterprise JavaBeans (EJB)	Maneja la ejecución de Enterprise beans para aplicaciones Java EE. Los Enterprise beans y sus contenedores corren en el servidor Java EE.
Contenedor de Red	Maneja la ejecución de componentes de página JSP y servlet para aplicaciones Java EE. Los componentes de red y sus contenedores corren en el servidor Java EE.
Contenedor de Aplicación Cliente	Maneja la ejecución de componentes de aplicaciones cliente. Las aplicaciones cliente y sus contenedores corren en el cliente.
Contenedor Applet	Maneja la ejecución de Applets. Consiste de un buscador de red y plug-in Java corriendo en conjunto en el cliente.

1.7.6 XML

XML es un estándar textual, extensible, multiplataforma para la representación de datos. Cuando datos XML son intercambiados entre diferentes partes, estas son libres de crear sus propias viñetas (tags) para describir los datos, configurar esquemas para determinar cuáles viñetas pueden usarse en una forma particular de documento XML.

1.8 Servidor de bases de datos

MySQL es un sistema de gestión de bases de datos SQL Open Source, desarrollado, distribuido y soportado por MySQL AB, es un sistema de gestión de bases de datos[10] (BD) Puede ser cualquier cosa, desde una simple lista de compra a una galería de pintura o las más vastas cantidades de información en una red corporativa. Para añadir, acceder, y procesar los datos almacenados en una base de datos, se necesita un sistema de gestión de base de datos como MySQL Server. Los sistemas de gestión de bases de datos juegan un papel central en computación, como aplicaciones autónomas o como parte de otras aplicaciones.

Es un sistema de gestión de bases de datos relacionales: Una base de datos relacional almacena datos en tablas separadas en lugar de poner todos los datos en un gran almacén. Esto añade velocidad y flexibilidad. La parte SQL de "MySQL" se refiere a "Lenguaje de Consulta Estructurado". SQL es el lenguaje estándar más común para acceder a bases de datos y está definido por el estándar ANSI/ISO SQL.

MySQL es "Open Source" significa que es posible para cualquiera usar y modificar el código fuente. Cualquiera puede bajar el software MySQL desde Internet y usarlo sin pagar nada. Si se desea, puede estudiarse el código fuente y cambiarlo para adaptarlo a sus necesidades. El software MySQL usa la licencia GPL (GNU General Public License). El servidor de base de datos MySQL es muy rápido, fiable y fácil de utilizar, se desarrolló originalmente para tratar grandes bases de datos mucho más rápido que soluciones existentes y ha sido usado con éxito en entornos de producción de alto rendimiento durante varios años. MySQL Server ofrece hoy en día una gran cantidad de funciones.

1.9 Mapeo de objetos relacional

El mapeo objeto-relacional (en inglés, Object-Relational mapping, ORM) 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 relacional. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos

[10] Una base de datos es una colección estructurada de datos.

(básicamente herencia y polimorfismo). Hay paquetes comerciales y de uso libre disponibles que desarrollan el mapeo relacional de objetos.

Hibernate es una API que se utiliza como herramienta de mapeo objeto/relacional para ambientes Java. Hibernate no sólo se encarga del mapeo desde las clases de Java hacia las tablas de bases de datos (y desde tipos de datos Java a tipos de datos SQL), sino proporciona también consulta y facilidades de recuperación de datos y puede reducir significativamente el tiempo de desarrollo comparado con el manejo manual de datos en SQL y JDBC. En SIPROE Hibernate permite recopilar la información, proveniente del controlador central, relativo a los periféricos conectados a través de él y guardar esta información en una base de datos desarrollada en mysql para posteriormente presentarla al usuario.

CONCLUSIONES DEL CAPITULO I

- El Sistema Inteligente SIPROE cuenta con tres protocolos importantes que controlan la comunicación a nivel de hardward: CAN, RS232 e I2C.

- La estructuración de SIPROE a nivel de software está desarrollada en una arquitectura multicapas, basadas en tecnología Java; siendo las principales librerías Java: Hibernate para persistencia de datos, Jamod como interface de comunicación entre el aplicativo intermedio web y el servidor modbus tcp/ip, MySQL como servidor y gestor de Bases de Datos.

- El protocolo modbus es implementado en modo RTU, tanto en el medio de comunicación Ethernet como en el medio serial, para optimizar la velocidad de transferencia y el tiempo de procesamiento de las tramas.

- El protocolo CAN está configurado para manejar tramas con identificador extendido de 29 bits, es utilizado para manejar la comunicación entre controladores y proveer al bus de comunicación de inmunidad al ruido.

- El protocolo I2C es utilizado en todos los dispositivos con el formato de dirección de 7 bits.

- El dispositivo utilizado para la adquisición de señales de entrada digitales es el MCP23008 que cuenta con 8 entradas las cuales son programadas para generar interrupciones al cambio y una interface I2C para comunicación.

- El dispositivo utilizado para el control de señales de salida digitales es el MCP23016 el cual cuenta con 16 salidas y una interface I2C para comunicación.

- El dispositivo utilizado para la adquisición de señales de entrada análogas es el MAX127 que cuenta con 8 entradas análogas de 12 bit de resolución cada una y con una interface I2C para comunicación.

- El dispositivo utilizado para el control de señales de salida análogas es el MAX521 el cual cuenta con 8 salidas análogas con resolución de 8bits y una interface I2C para comunicación.

CAPITULO II

DISEÑO E IMPLEMENTACION DEL HARDWARE

Introducción.

El presente capitulo muestra la Ingeniería de diseño aplicada al Sistema Inteligente que implementa el prototipo SIPROE v1.0, los criterios de construcción para los diferentes Controladores Inteligentes describiendo las características particulares en hardware y firmware de cada uno de ellos. Logrando un diseño compacto y funcional obteniendo como producto final un prototipo SIPROE V1.0 con características de administración, control y monitoreo de sistemas de iluminación, aire acondicionado y acceso.

De igual forma, se describen a detalle cada una de las partes que componen el hardware de cada Controlador Inteligente, el cual está formado por un Controlador Inteligente Maestro (CIM), un Controlador Inteligente de Reloj de Tiempo Real (CIRTR), un Controlador Inteligente de Propósito General para iluminación o aire acondicionado (CIPG) y un Controlador Inteligente de Acceso (CIA).

Se describen los bloques funcionales de SIPROE, sus características eléctricas y constructivas, las características de los medios e interfaces que permiten al Sistema comunicarse con sus Expansores del lado de los transductores y actuadores, con sus Controladores desde el punto de vista modular y desde el punto de vista Cliente-Servidor, con el mundo exterior a través de un Aplicativo Intermedio gestor de la comunicación con los usuarios del sistema.

2.1 Criterios de diseño

Con la finalidad de implementar las funcionalidades de control, administración y monitoreo para la optimización en el uso de la energía y control de acceso. El diseño a nivel de hardware está basado en los criterios mostrados en la **Tabla 2.1.**

Tabla 2.1 Criterios de Diseño de SIPROE

CRITERIO	DESCRIPCION	SOLUCION QUE SATISFACE EL CRITERIO		
		FABRICANTE	PRODUCTO	DESCRIPCION
Selección de las tecnologías	La tecnología hardware que: • Implemente el protocolo I2C, CAN y permita desarrollar un sistema de adquisición de datos eficiente. • Permita la transferencia de información en el protocolo libre y ampliamente utilizado MODBUS. • Proporcione un control del tiempo real independiente del software.	MICROCHIP	PIC18FXX8	Microcontrolador
			MCP230XX	Expansor E/S digital
		MAXIM	MAX127	Expansor Entradas digital
			MAX521	Expansor Salidas digital
			DS1337	Reloj de tiempo real, calendario y alarmas
		GRIDCONECT	XPORT MBTCP	Servidor modbus TCP/RTU
	La tecnología software que: • Eficiente el almacenamiento de información, la persistencia de la misma en aplicaciones graficas. • Proporcione un servidor de aplicaciones web • Sea un Lenguaje de programación que sea eficiente, libre y compatible con las tecnologías empleadas que permita optimizar costos. • Sea un aplicativo compatible con el lenguaje de programación que permita la transferencia de información a través de un medio Ethernet y con el protocolo modbus.	MySQL AB	MySQL	Servidor de bases de Datos
		Apache	Tomcat	Servidor de paginas web y aplicaciones
		SUN	Java	Lenguaje de programación Orientado a Objetos
		Jamod	Jamod	API de transacciones MODBUS TCP

Tabla 2.1 Criterios de Diseño de SIPROE (continuación)

CRITERIO	DESCRIPCION	SOLUCION QUE SATISFACE EL CRITERIO		
		FABRICANTE	PRODUCTO	DESCRIPCION
Protocolos de comunicación	El protocolo que: • Permita transmisión de datos en ambientes industriales con alto nivel de ruido y permita expandir la red de comunicación de forma modular y eficiente. • Que permita una comunicación entre dispositivos electrónicos rápida y eficientemente. • Que sea ampliamente utilizado en el campo de los controladores lógicos • Que permita la comunicación a través de Internet para romper las barreras de distancia.	BOSCH	CAN	Control de Area de Red
		Philips	I2C	Inter Circuitos Integrados
		Modicon	MODBUS	Protocolo basado en el modelo OSI
		Vin Cert-Robert Khan	TCP/IP	Protocolo de control de transmisiones
Adaptabilidad	Permitir el desarrollo de un sistema modular y distribuido que permita la adaptabilidad del Sistema a diferentes entornos de utilización con cambios mínimos en su estructura constructiva.			
Control de cargas	Dotar al sistema con la capacidad manejar cargas de baja potencia eléctrica de hasta 120VAC 5ª reduciendo en consumo de energía de los actuadores.	Panasonic	Relé DPDT	Relé doble bobina a pulso con enclavamiento
		HP	HCPL3700	Transductor AC/DC

2.2 Funcionalidades MODBUS implementadas en SIPROE

Las funcionalidades que implementa el prototipo SIPROE para controlar, administrar y monitorear sistemas de iluminación, aire acondicionado, control de acceso y seguridad se realizan por medio de una aplicación web, la cual envía peticiones MODBUS al aplicativo intermedio, el encargado de la interacción entre el hardware y la aplicación web el cual procesa las peticiones de MODBUS.

En el firmware se desarrollan cuatro funciones del protocolo MODBUS: función 02, 03, 05, y 16. Para ejecutar las acciones pertinentes solicitadas por el cliente MODBUS, este cliente es una aplicación web con privilegios según rol administrativo para ejecutar todas las opciones o con rol usuario con restricciones. En la Tabla 2.2 se muestra un resumen de las funciones del hardware.

2.3 Diagrama en bloques de SIPROE V1.0

El diagrama de bloques del sistema inteligente de SIPROE V1.0, se muestra en la Figura 2.1 se detalla las partes que componen el sistema inteligente, la interface de usuario, el servidor web, servidor de base de datos, el controlador o aplicativo intermedio de SIPROE y los diferentes controladores.

El hardware se ha dividido en varios controladores a los que se les denomina en forma genérica controladores inteligentes, estos son:

- Controlador Maestro Integrador de SACME V1.0 y SIPROE V1.0
- Controlador para un sistema de Reloj de Tiempo Real “RTC”
- Controlador para un sistema de Control de Acceso
- Controlador de propósito general con funcionalidades para sistemas de Iluminación y Aire Acondicionado

Con esta división del hardware se cumple con el criterio de diseño[11] de hacer el hardware de forma modular y así poder adaptarlo a los requerimientos de diferentes aplicaciones. El controlador maestro puede funcionar solo y realizar las funciones desempeñadas por el modulo electrónico SACME, según se requiera se van agregando controladores, los cuales expanden las funciones del sistema inteligente.

[11] Ver criterios de diseño apartado 2.1, pág. 54

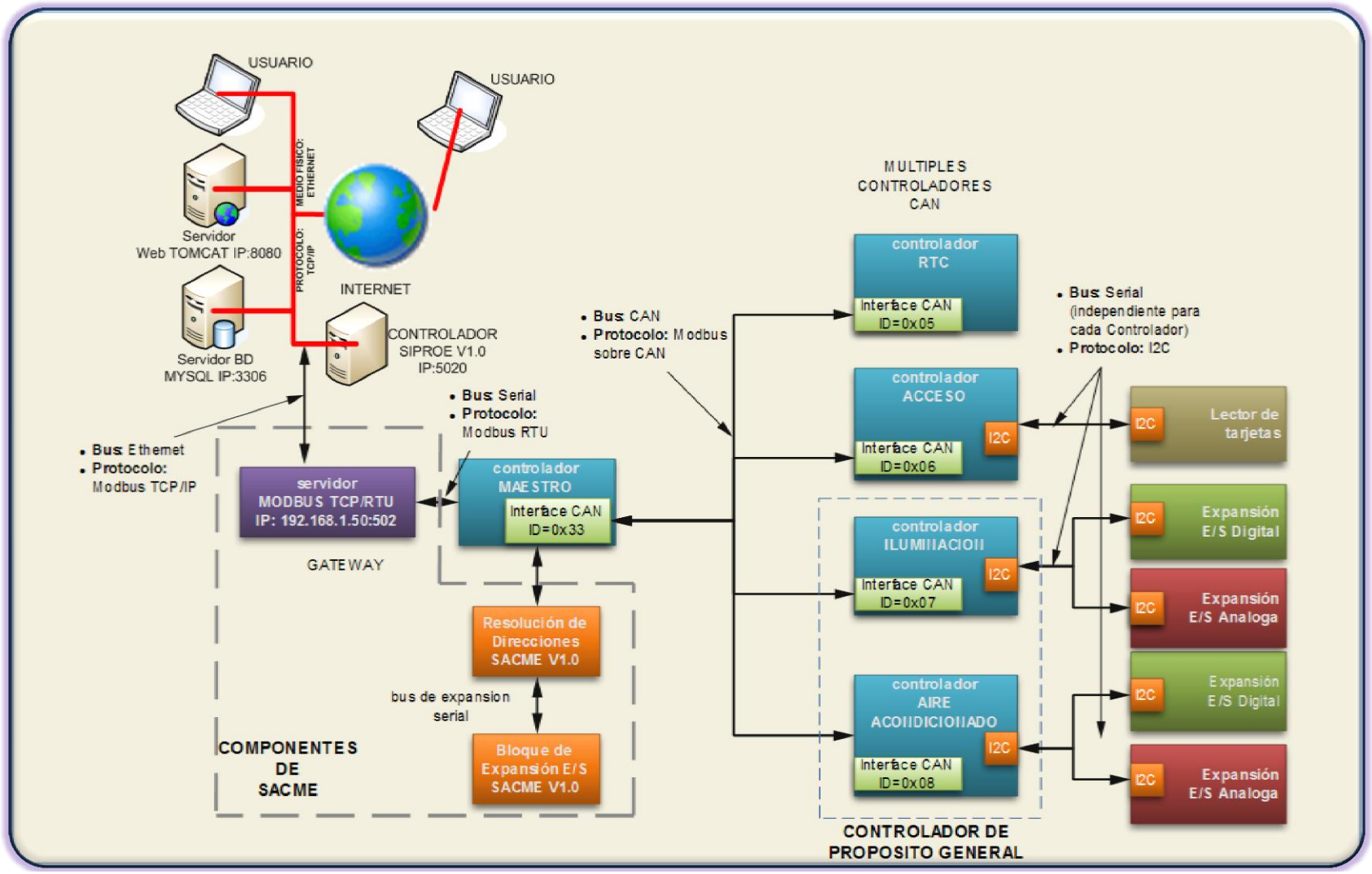


Figura 2.1 Diagrama de Bloques esquemático de SIPROE

Tabla 2.2 Funciones del hardware.

Función	Propósito
Leer entrada discreta (02)	Esta función permite ver el estado de una o varias de entradas discretas, las cuales muestran el estado de las cargas que se están controlando. En una petición se hace una lectura de 32 entradas máximo.
Leer registros (03)	Esta función permite leer registros de configuración de cualquier controlador, la tarjeta de acceso, registros de estado del reloj de tiempo real.
Forzar salida discreta (05)	Esta función permite activar una salida con la cual se controla una carga determinada.
Escritura de registros (16)	Esta función permite cargar la configuración a cada uno de los controladores para manejar los eventos generales por los diferentes sensores y entradas análogas, cargar los datos de identificación en la tarjeta de acceso, escribir en una salida analógica y programar la hora, fecha y la alarma del reloj de tiempo real.

El diagrama de bloques de SIPROE muestra la conservación de las funcionalidades del modulo electrónico SACME[12] V1.0, para el control de entradas y salidas on – off. Las características del modulo electrónico SACME se muestran en la Tabla 2.3, las características del sistema inteligente SIPROE se muestran en la Tabla 2.4, y sus características eléctricas en la Tabla 2.5

Tabla 2.3 Características del modulo electrónico SACME

DESCRIPCION
Manejo de 160 entradas o salidas on-off desde un único modulo central.
Metodología de encriptación en la seguridad de acceso al aplicativo WEB basado en el API de seguridad y encriptación SHA1BASE64, de Java.
Asignación de Roles para el monitoreo, control y administración, por medio de una base de datos desarrollada en MySQL.
Asignación de pines por rol y usuario, para el acotamiento del acceso restringido a los usuarios del sistema.
Micro controlador de la gama 16F con una velocidad de 4MHz (1 MIPS[13])
Expansión de entradas y salidas basada en multiplexores, demultiplexores y en decodificadores de bytes.
Manejo de cargas por medio de relés de pulsos de consumo de energía despreciable.
Retroalimentación de señales para la verificación de estados.

[12] Para mayor detalle de todas las funcionalidades del modulo electrónico SACME, revisar el 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

[13] MIPS: Millón de Instrucciones por segundo

Tabla 2.4 Las características funcionales de SIPROE

DESCRIPCION
Sistema distribuido de entradas y salidas digitales y análogas.
Velocidad de procesamiento y ejecución por el uso de µC de la gama 18Fxx8 con un reloj de 10MHz (2.5 MIPS), incorporando el protocolo CAN e I2C.
Incorpora el uso de dispositivos periféricos I2C para expansión de entradas y salidas análogas o digitales**,** para mejorar el método de lectura y escritura e incrementar eficientemente los tiempos de procesamiento de las peticiones MODBUS. Los expansores utilizados son de 8 o 16 bits para, con una sola instrucción, leer hasta 8 entradas digitales o análogas o escribir hasta 8 salidas digitales o análogas.
Incorporación de relés de pulso de doble bobina disminuyendo el hardware necesario para activación y desactivación de contactos.
Incorporación de entradas y salidas análogas para manejo de entornos físicos en tiempo real, ampliando la utilización de SIPROE V1.0 al manejo de variables físicas más complejas.
Incorporación de un controlador de reloj calendario de tiempo real.
El firmware implementa las funciones 2 y 5 del protocolo MODBUS adaptándolo a los requerimientos del µC. Adicionalmente se ha agregado la implementación de las funciones 3 y 16 para la lectura/escritura de registros de retención de entradas y salidas análogas.
Incorpora el protocolo CAN como comunicación a nivel de bus local.
El sistema inteligente implementa el protocolo MODBUS sobre CAN, para poder incorporar a SIPROE a redes MODBUS que están diseñadas sobre el protocolo CAN.

Tabla 2.5 Características eléctricas de Entradas y Salidas

TIPO	CARACTERISTICAS ELECTRICAS
ENTRADA DIGITALES	TTL
SALIDAS DIGITALES	TTL
SALIDAS ANALOGAS	Vsal= 0.3VDC – Vref +0.3Vdc Vref=4Vpp f=10kHz Imax = 50mA
Control de cargas Con salidas digitales	Cargas de 110VAC 5Amp máx.

El servidor Modbus TCP, tiene un único Identificador de esclavo modbus (ID=1) por tanto el enrutamiento de las peticiones modbus dentro de SIPROE se lleva a cabo en el controlador maestro utilizando los seis bits más significativos de la parte alta de la dirección de inicio ("STARTING ADDRESS HI") dentro del DPU. Cuando los controladores inteligente recibe íntegramente la petición; estos seis bits son enmascarados para dar como resultado la verdadera dirección de la entrada, bobina o registro a leer o escribir según la función MODBUS; por tanto, desde el punto de vista del controlador inteligente, estos 6 bits dejan de ser parte del procesamiento de la trama modbus y pasan a formar la parte del identificador CAN como el byte menos significativo; con esto se logra redireccionar las peticiones para que sean procesadas por los diferentes controladores. La Figura 2.2 muestra el enmascarado del ID de los controladores. Por ejemplo se le hace una petición al controlador reloj de tiempo real con ID=0x05 y se pone en los 6 bits más significativos del "STARTING ADDRESS HI" con esto se logra direccionar hasta 64 controladores.

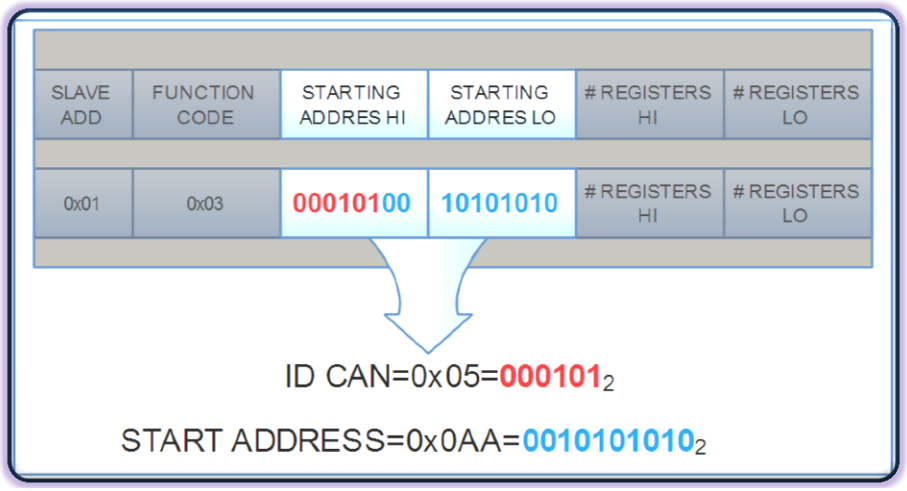


Figura 2.2 Enmascarado del ID de los controladores.

2.4 Diseño e implementación del controlador maestro

El Controlador Maestro de SIPROE cumple las funcionalidades de SACME v1.0 y sirve como un "gateway" para los controladores inteligentes; es decir, desde el punto de vista de SACME, es el núcleo central del procesamiento de peticiones modbus; desde el punto de vista SIPROE cumple la función de enrutador de peticiones y puente de interconexión con la aplicación de usuario. Para una adecuada adaptación de SIPROE a SACME se utiliza un µC PIC18F458, que tiene la misma distribución de pines que el PIC16F877.

2.4.1 Servidor modbus TCP/IP

El controlador maestro incorpora un servidor MODBUS que atiende a los clientes del sistema SIPROE, este servidor actúa como un "Gateway Ethernet-serial". Las peticiones recibidas por este gateway son enviadas por el bus CAN a los demás controladores de funcionalidades, La Figura 2.3 muestra el diagrama en bloque del servidor MODBUS.

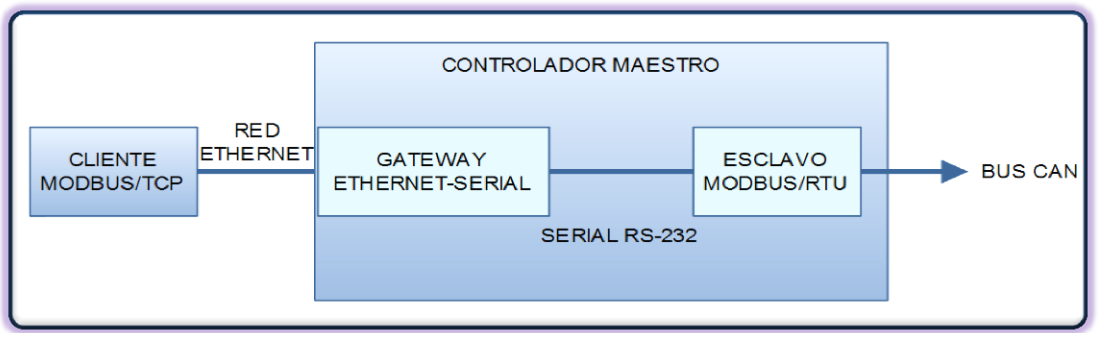


Figura 2.3 Diagrama de bloque controlador maestro

2.4.2 Firmware del controlador maestro

El desarrollo del firmware[14] se estructuro en módulos para facilitar la depuración, el código se escribió en lenguaje ensamblador y se editó, compiló y depuró en MPLAB[15] IDE. La Tabla 2.6 y la
Muestran un resumen de los archivos del firmware que se deben incluir en el proyecto del controlador maestro llamado SIPROEmain0.mcw[16].

Tabla 2.6 Descripción de archivos asm

Archivo	Descripción
18F458TEMP.asm	Código principal del controlador maestro, desde aquí se llaman las librerías CAN, modbus, i2c, USART y time.
Can18xx8.asm	Librería CAN proporcionada por MICROCHIP
funcionesModbusM0.asm	Librería modbus implementa funciones 02, 03, 05 y 16
I2C.asm	Librería I2C implementa comunicación con dispositivos I2C, LCD y teclado
Time.asm	Librería para manejar funciones de retardo y configuración de timer0 y timer1.
USART.asm	Librería cara utilizar el modulo de comunicación serial.

[14] Ver anexo A2.1
[15] Software proporcionado por MICROCHIP para el desarrollo de firmware
[16] El proyecto SIPROEmain0.mcw incluye todos los archivos .asm y .inc necesario para compilarlo, para ver el firmware del modulo central revisar cd adjunto al documento

Tabla 2.7 Descripción de archivos inc

Archivo	Descripción
CAN18xx8.inc	Lista de funciones CAN que pueden ser utilizar, solo es necesario incluir este archivo en el archivo .asm que se utiliza
CANDef.inc	Archivo donde se realizan las definiciones de variables, reserva de memoria y definición de macros utilizadas por la librería CAN
Definiciones1.inc	Archivo donde se definen métodos, macros y variables utilizadas por las librerías I2C, time y funcionesModbusM0
I2Clink.inc	Lista de funciones de la librería I2C Y funciones de LCD que pueden ser utilizadas desde cualquier archivo .asm
MacrosI2C.inc	Definiciones de macros utilizadas en la librería I2C
Modbuslink.inc	Lista de funciones modbus que pueden ser utilizadas, para utilizar las funciones es necesario que en la librería estén definidas como global.
P18f458.inc	Definición de variables reservadas definidas por MICROCHIP
Time.inc	Lista de funciones de la librería time que pueden ser utilizadas.
USART.inc	Lista de funciones de la librería USART que pueden ser utilizadas.
Var_modbus.inc	Definición de variables utilizadas por la librería modbus
Variables.inc	Definición de variables generales

El código está dividido en código principal, librerías de funciones específicas, archivos de definición de variables y macros. A continuación se presenta una sección de firmware para la configuración del CAN:

```
;*****************************************************************************
;               CONFIGURACION DEL PROTOCOLO CAN                                 *
;               configuracion inicial 125kbps@10MHz all valid messages          *
;               CANInitialize  SJW, BRP, PHSEG1, PHSEG2, PROPSEG, Flags         *
;               PHSEG1+PHSEG2+PROPSEG=          #TQ-1                           *
;*****************************************************************************
Init_CAN:
                CLRWDT
                CANInitialize 1, 5, 3, 3, 1,CAN_CONFIG_ALL_VALID_MSG ; Inicializar el Modulo a 125kbps a 10MHz
                ;CANInitialize          1, 1, 3, 3, 1,CAN_CONFIG_ALL_VALID_MSG ; configuracion con 4Mhz, 125kbps
#ifdef  RX_TEST
;Set configuration mode
                CANSetOperationMode             CAN_OP_MODE_CONFIG     ; Activar modo Configuracion
;configurar Mascara B0 to 0xfffffeff=11111111111111111111111011111111
                CANSetReg   CAN_MASK_B0, 0xFFFFFEFF, CAN_CONFIG_XTD_MSG
                ;CANSetReg  CAN_MASK_B0, 0xFFFFFEFF, CAN_CONFIG_STD_MSG
;Set Filter 0 with
                CANSetReg   CAN_FILTER_B0_F1, 0x011, CAN_CONFIG_XTD_MSG;mensajes del CONTROLADOR DE ACC
;Set Filter 1 with
                CANSetReg   CAN_FILTER_B0_F2, 0x22, CAN_CONFIG_XTD_MSG  ;mensajes del CONTROLADOR A/C E
ILUMINACION
    CANSetReg CAN_MASK_B1,0xFEFFFF7F, CAN_CONFIG_XTD_MSG
;Set Filer 2 with 0x33
    CANSetReg CAN_FILTER_B1_F1,0x33, CAN_CONFIG_XTD_MSG                         ;MENSAJES DEL RELOJ
```

Este segmento de código fuente está incluido en el archivo 18F458TMP.asm, el cual realiza la configuración del protocolo CAN, con la función CANinitialize, la cual recibe 6 parámetros. Con estos se fija una comunicación de 125Kbps a 10Mhz. También se configuran los filtros CAN en los que escucha a los controladores, la función CANsetReg recibe como parámetro el filtro o la máscara del buffer a configurar, el ID del filtro y el tipo de filtro estándar o extendido, para nuestro caso extendido.
El controlador maestro escucha a los demás controladores en diferentes filtros por ejemplo: al control de acceso lo escucha en el filtro uno del buffer cero con ID=0x11, al controlador de propósito general lo escucha en el filtro dos del buffer cero con ID=0x22.
La Figura 2.4 y Figura 2.5 muestran el flujo grama de la parte más importante de la lógica del firmware.

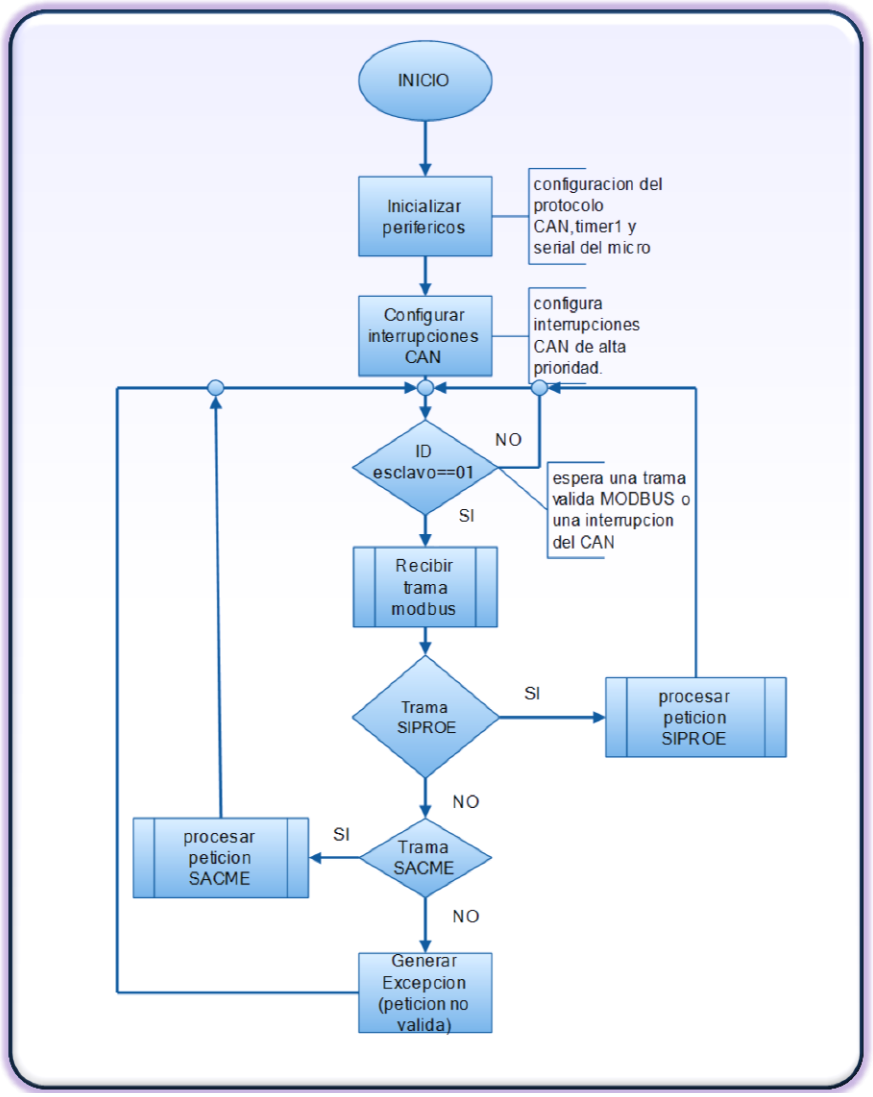


Figura 2.4 Flujo grama de controlador inteligente maestro

En el sistema inteligente SIPROE a los diferentes controladores se les han asignado ID por defecto los cuales son:

- Controlador de reloj calendario de tiempo real ID = 0x05
- Controlador de acceso ID = 0x06
- Controlador de propósito general ID = 0x07
- Controlador maestro ID = 0x33

Siempre que se realice una funcionalidad de administrar, monitorear y controlar a cual quier controlador se debe de hacer a estas direcciones. Por ejemplo el controlador maestro se monitorear haciendo una petición de la lista de controladores conectados en el bus CAN, esto se hace a la dirección 0x33

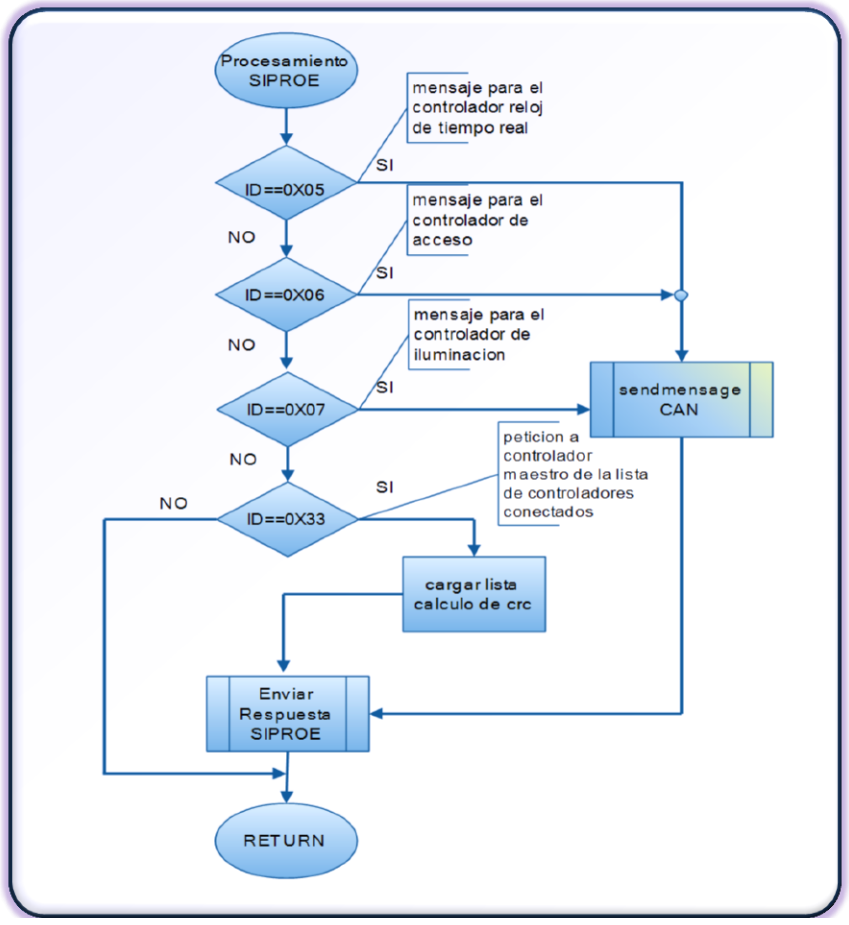


Figura 2.5 Flujo grama del firmware del controlador inteligente maestro (continuación)

2.4.3 Diagrama eléctrico del Controlador Maestro

La Figura 2.6 muestra el diagrama eléctrico del Controlador Maestro SIPROE v1.0.

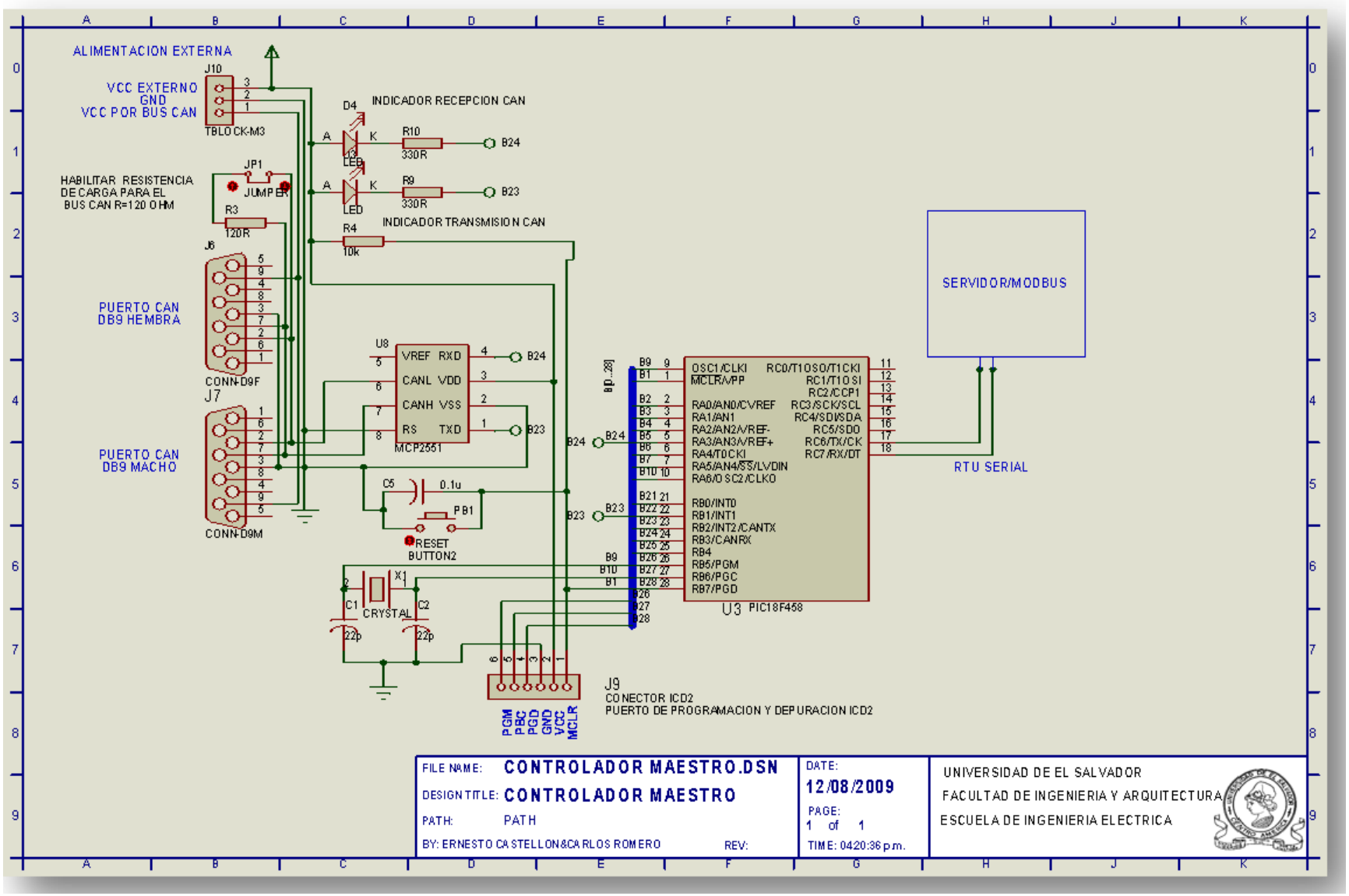


Figura 2.6 Diagrama Eléctrico del Controlador Maestro

2.4.4 Costo de implementación de controlador maestro

Los dispositivos utilizados en la implementación del controlador maestro incrementa un poco los costos de envió a pesar de eso el costo de implementación es relativamente bajo de **$71.38**[17]

2.5 Diseño e implementación del controlador de Reloj de tiempo real

El módulo SIPROE cuenta con controladores cuya función es proveer una interface hardware para los diferentes entornos físicos de aplicaciones para los cuales ha sido diseñado. Cada controlador inteligente está constituido por un µC PIC18F258, que tiene embebido un controlador CAN y una interface I2C como bloques básicos para su funcionamiento.
Debido a que el sistema se definió como autónomo, se diseño un controlador inteligente reloj calendario de tiempo real para proporcionar la hora y la fecha a los diferentes controladores. Por lo que si ocurriera una falla a nivel del servidor web esto no afectaría al sistema SIPROE. Porque cuenta con un reloj embebido en el bus CAN para manejar eventos.
El chip que implementa el reloj es DS1337[18], este IC se encarga de proporcionar tanto el calendario como la hora en tiempo real al modulo SIPROE V1.0.En este controlador estarán disponibles todos la eventos programados por los usuarios y será consultado el tiempo a través de la interface CAN embebida en el µC. En la Figura 2.7 se muestra el diagrama de bloques completo de este Controlador.

[17] Ver anexo A 3.1 para mayor detalle de la lista de materiales y sus precios locales
[18] Las características del chip DS1337 se describen en la sección 1.5.2 página 26

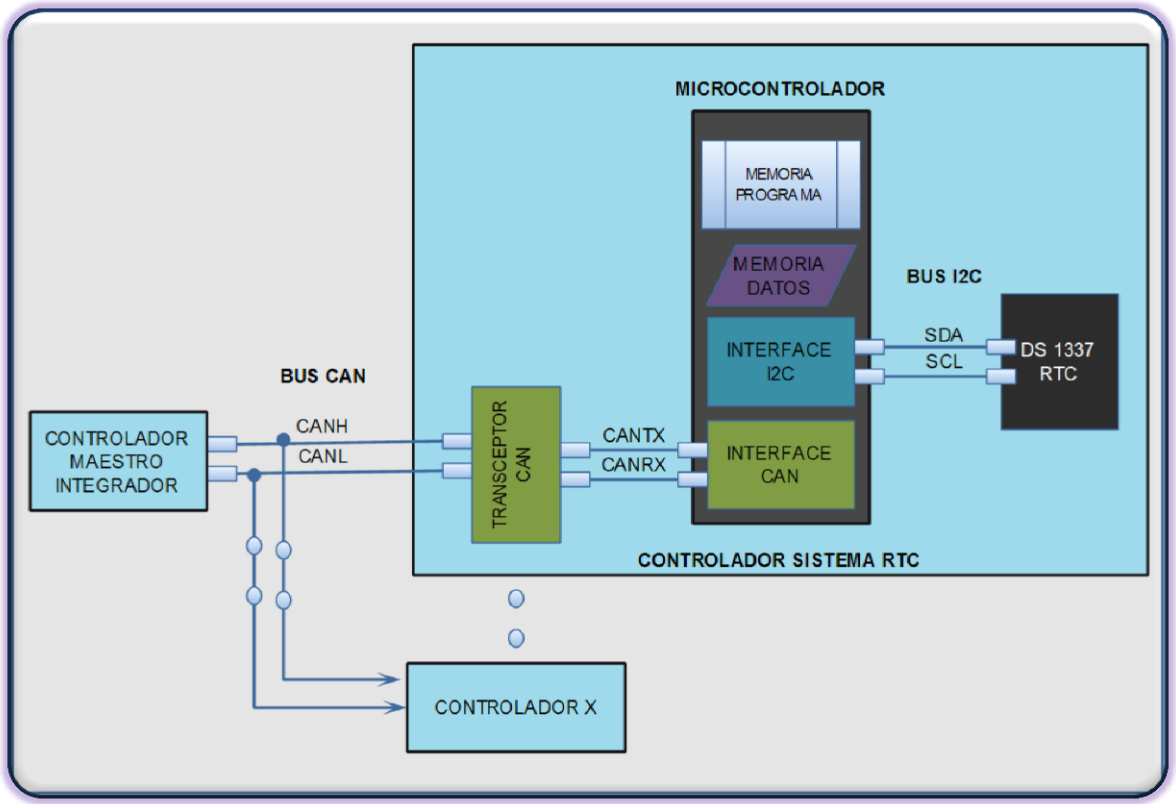


Figura 2.7 Diagrama de bloques del controlador RTC.

2.5.1 Firmware del controlador de reloj calendario

El desarrollo del firmware del controlador de reloj calendario se escribió en forma modular, en un proyecto llamado relojCAN.wcm[19]. La descripción de los archivos incluidos en se muestra en Tabla 2.8 y Tabla 2.9

Tabla 2.8 Descripción de firmware controlador reloj calendario

Archivo	Descripción
18F258TEMP.asm	Código del controlador reloj calendario, desde aquí se llaman las librerías CAN, modbus, i2c, USART y time.
Can18xx8.asm	Librería CAN proporcionada por MICROCHIP
funcionesModbusM0.asm	Librería modbus implementa funciones 02, 03, 05 y 16
I2C.asm	Librería I2C implementa comunicación con dispositivos I2C reloj calendario de tiempo real
Time.asm	Librería para manejar funciones de retardo y configuración de timer0 y timer1.

[19] El proyecto está en el cd adjunto al documento, solo es necesario cargarlo en el MPLAB IDE y compilarlo.

Tabla 2.9 Archivos inc del firmware reloj calendario

Archivo	Descripción
CAN18xx8.inc	Lista de funciones CAN que pueden ser utilizadas, solo es necesario incluir este archivo en el archivo .asm que se utiliza
CANDef.inc	Archivo donde se realizan las definiciones de variables, reserva de memoria y definición de macros utilizadas por la librería CAN
Definiciones1.inc	Archivo donde se definen métodos, macros y variables utilizadas por las librerías I2C, time y funcionesModbusM0
I2Clink.inc	Lista de funciones de la librería I2C que pueden ser utilizadas desde cualquier archivo .asm
MacrosI2C.inc	Definiciones de macros utilizadas en la librería I2C
Modbuslink.inc	Lista de funciones modbus que pueden ser utilizadas, para utilizar las funciones es necesario que en la librería estén definidas como global.
P18f258.inc	Definición de variables reservadas definidas por MICROCHIP
Time.inc	Lista de funciones de la librería time que pueden ser utilizadas.
Var_modbus.inc	Definición de variables utilizadas por la librería modbus
Variables.inc	Definición de variables generales

La Figura 2.8 y Figura 2.9 muestra el flujo grama de la sección de código más importante del desarrollo del firmware del controlador de reloj. Como se puede ver hay un rutina de atención a interrupción, estas interrupción son generadas por el periférico reloj y por el CAN. Todas las interrupciones son de alta prioridad.

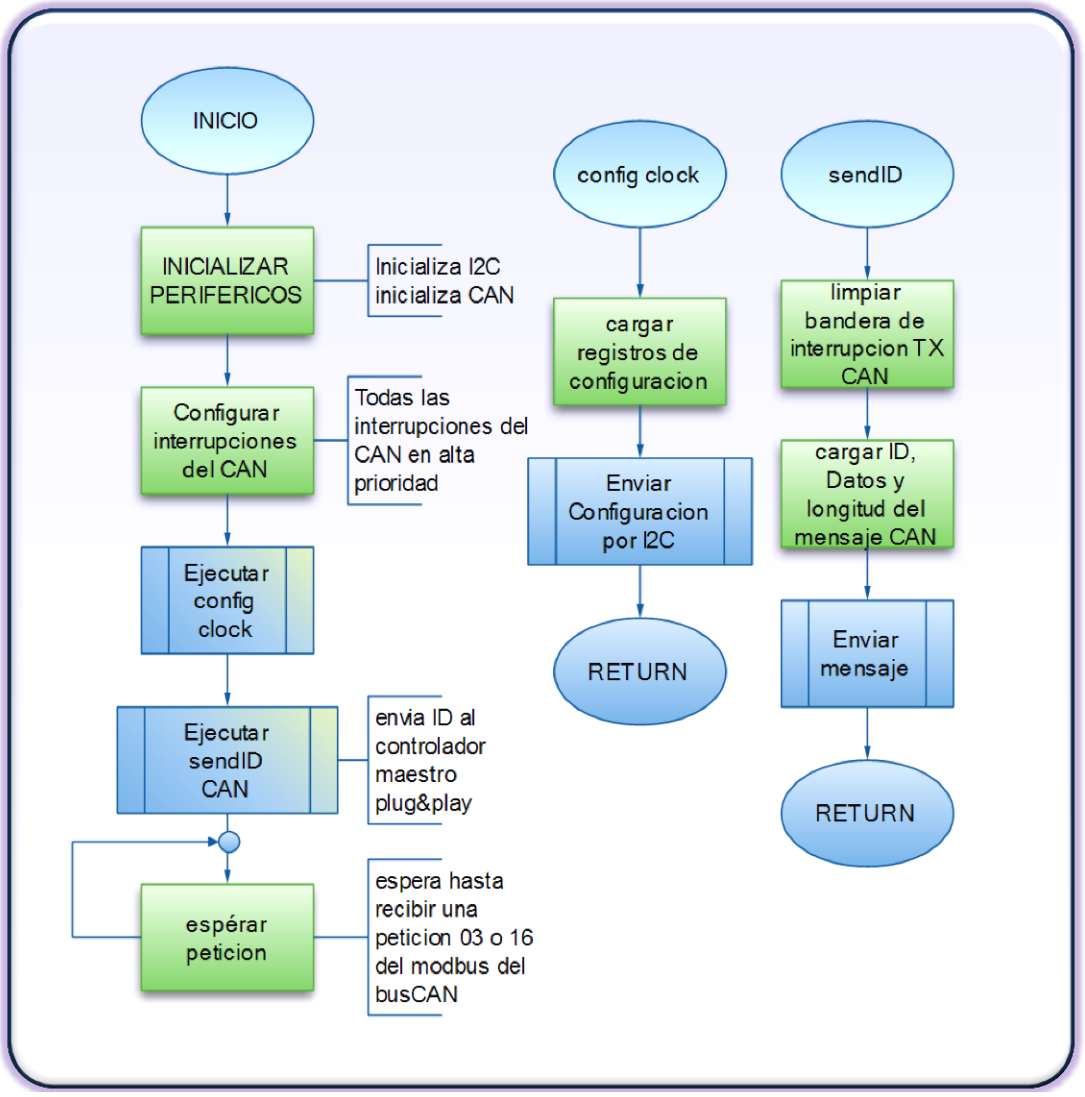


Figura 2.8 Flujo grama de código del controlador de reloj

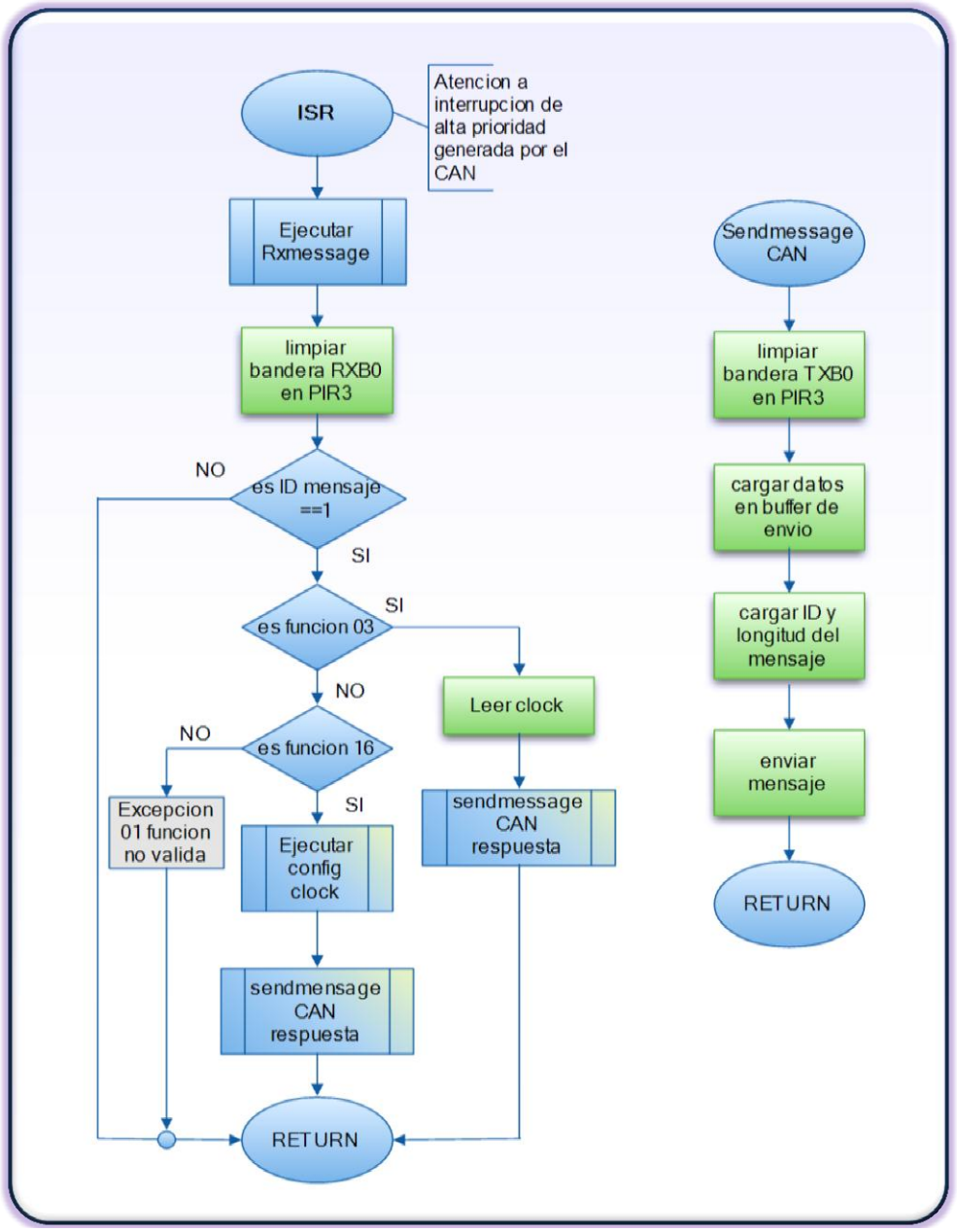


Figura 2.9 Flujo grama de código del controlador de reloj (continuación)

2.5.2 Diagrama eléctrico del Controlador de reloj de tiempo real

La Figura 2.10 muestra el diagrama eléctrico del Controlador Inteligente de tiempo real. La lista de materiales y el costo de implementación[20] es relativamente bajo $21.00, lo cual es un indicio de la factibilidad de su implementación.

[20] Ver anexo A 3.2 para mayor detalle de materiales y precios locales.

El funcionamiento del Controlador RTC es sencillo. Está programado para generar un pulso de reloj de 1Hz en su salida INTB, y para generar un pulso de reloj cada minuto en su pin INTA; el cual es interpretado por el micro controlador PIC18F258 como una interrupción en pin RB0, atendiendo la interrupción leyendo el reloj y enviando una trama broadcast a través del bus CAN replicando su hora actual a todos los controladores conectados al bus. Los diferentes controladores pueden hacer peticiones de la hora al reloj utilizando la función 3 del modbus. El único controlador capaz de actualizar la hora de este controlador RTC es el maestro a partir de una petición de actualización de hora solicitada por el usuario con rol administrativo.

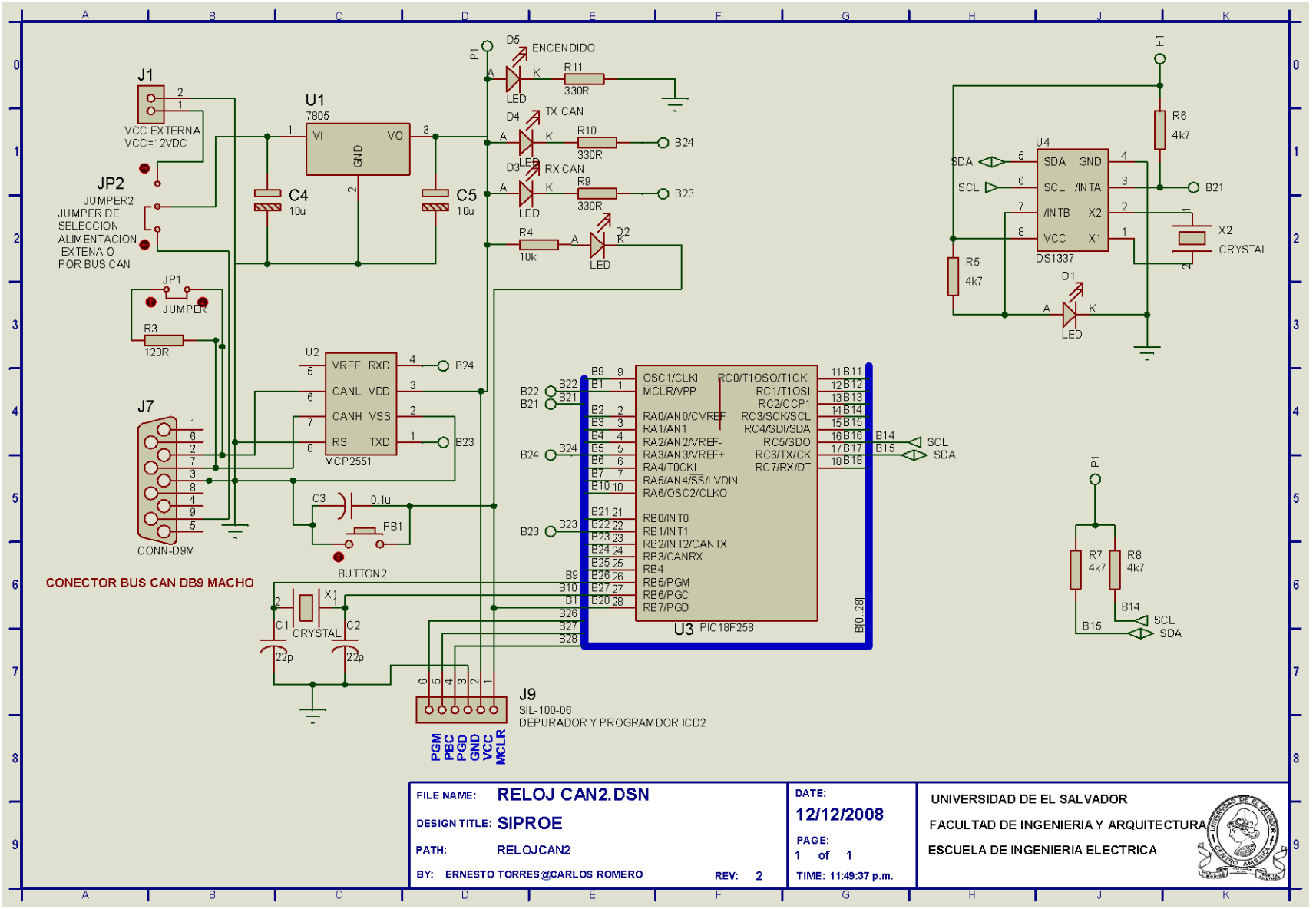


Figura 2.10 Diagrama eléctrico del Controlador Inteligente de Reloj de Tiempo Real

2.1 Diseño e implementación del controlador de Propósito General

El prototipo del controlador inteligente de propósito general está construido en forma modular, para fácil mantenimiento. En el diseño de la solución para este controlador se consideran el manejo de cargas de potencia, por lo que incluye una etapa de buffers para el manejo de los relés de doble pulso. Además una interface con un acople óptico con opto acopladores en las entradas digitales. También maneja entradas y salidas análogas. En la Tabla 2.10 se muestra un cuadro resumen de las características del controlador.

Tabla 2.10 Características del controlador de propósito general

Características
Micro controlador de rango alto PIC18F258
Interface CAN con dos conectores DB9
Puerto de programación y depuración ICD2
8 salidas de manejo de cargas de 120 VAC a 8 Amp.
8 entradas digitales 5V max
16 entradas analógicas 5V max
8 salidas análogas 5V max

2.1.1 Firmware de controlador de propósito general

El firmware del controlador de propósito general también se estructura de forma modular, esto nos permite reutilizar las librerías desarrolladas en los otros controladores.

El funcionamiento de este controlador es secuencial, Al reiniciar ejecuta una rutina de configuración de sus periféricos, posteriormente lee sus entradas análogas, digitales, solicita la hora y espera que una petición modbus sea solicitada, si no hay peticiones, verifica que evento está activo a la fecha, si hay uno que no lo esté se debe ajustarse la hora para repetirlo o debe borrarlo. El proceso continúa con el análisis de la lógica secuencial y termina ejecutando ya sea la petición del usuario si existe una o el resultado de la lógica secuencial según corresponda. El proceso es cíclicamente repetido sin la re inicialización de los periféricos.

El nombre del proyecto del controlador de propósito general con funcionalidades es ilumina.mcp. los archivos que se deben incluir en el proyecto se presentan en la Tabla 2.11 y Tabla 2.12.

Tabla 2.11 Descripción del firmware del controlador de propósito general

Archivo	Descripción
18F258TEMP.asm	Código del controlador de propósito general, desde aquí se llaman las librerías CAN, modbus, i2c, USART y time.
Can18xx8.asm	Librería CAN proporcionada por MICROCHIP
funcionesModbusM0.asm	Librería modbus implementa funciones 02, 03, 05 y 16
I2C.asm	Librería I2C implementa comunicación con dispositivos I2C expansores de entrada salida, convertidores análogos, DAC,
Time.asm	Librería para manejar funciones de retardo y configuración de timer0 y timer1.

Tabla 2.12 Archivos inc del firmware del controlador de propósito general

Archivo	Descripción
CAN18xx8.inc	Lista de funciones CAN que pueden ser utilizadas, solo es necesario incluir este archivo en el archivo .asm que se utiliza
CANDef.inc	Archivo donde se realizan las definiciones de variables, reserva de memoria y definición de macros utilizadas por la librería CAN
Definiciones1.inc	Archivo donde se definen métodos, macros y variables utilizadas por las librerías I2C, time y funcionesModbusM0
I2Clink.inc	Lista de funciones de la librería I2C que pueden ser utilizadas desde cualquier archivo .asm
MacrosI2C.inc	Definiciones de macros utilizadas en la librería I2C
Modbuslink.inc	Lista de funciones modbus que pueden ser utilizadas, para utilizar las funciones es necesario que en la librería estén definidas como global.
P18f258.inc	Definición de variables reservadas definidas por MICROCHIP
Time.inc	Lista de funciones de la librería time que pueden ser utilizadas.
Var_modbus.inc	Definición de variables utilizadas por la librería modbus
Variables.inc	Definición de variables generales
Basedatos.inc	Definición de variables para manejar los eventos

Además se presenta la lógica del segmento de firmware más importante del controlador de acceso, esto se puede apreciar en la Figura 2.11 y la Figura 2.12

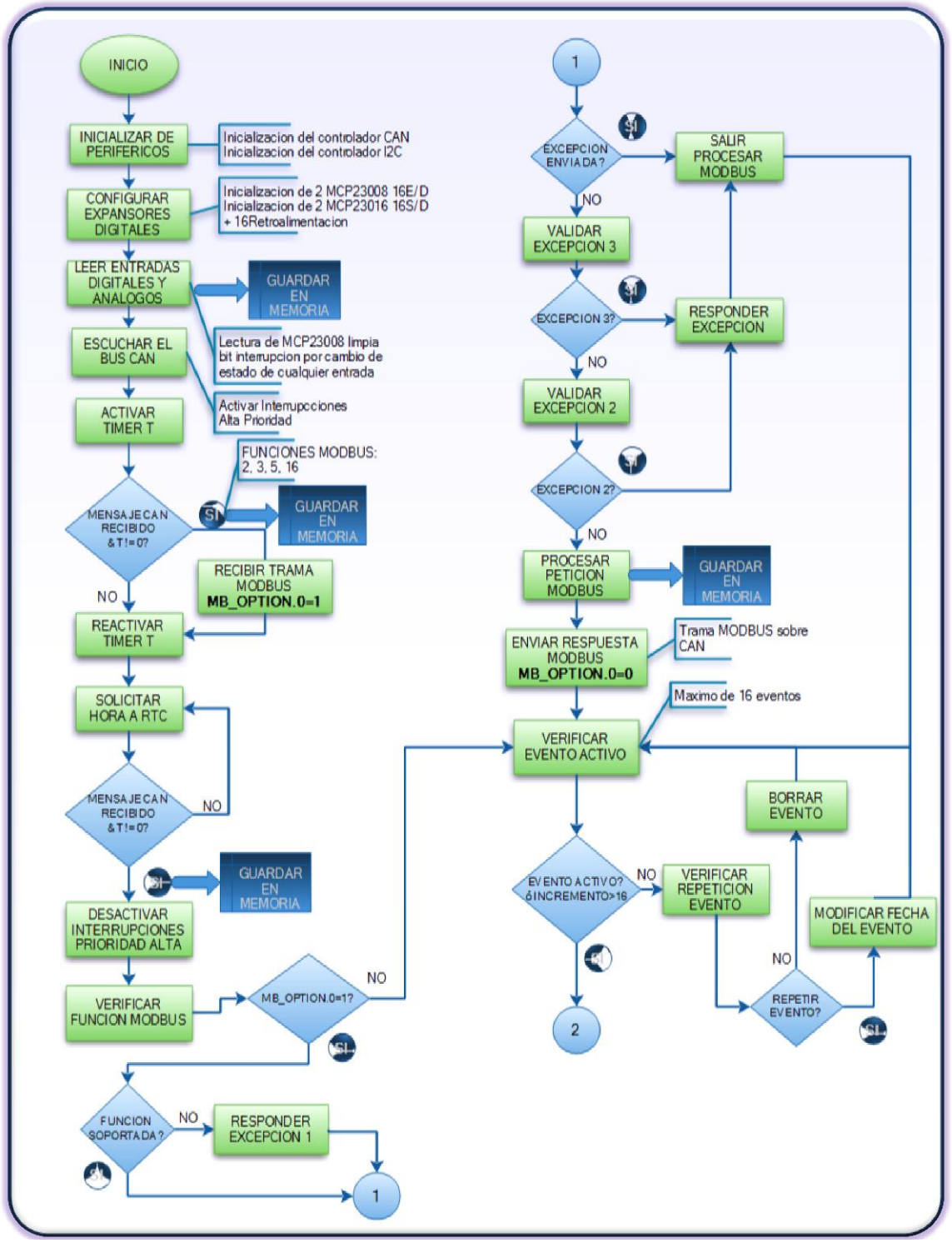


Figura 2.11 Flujo grama del controlador de propósito general

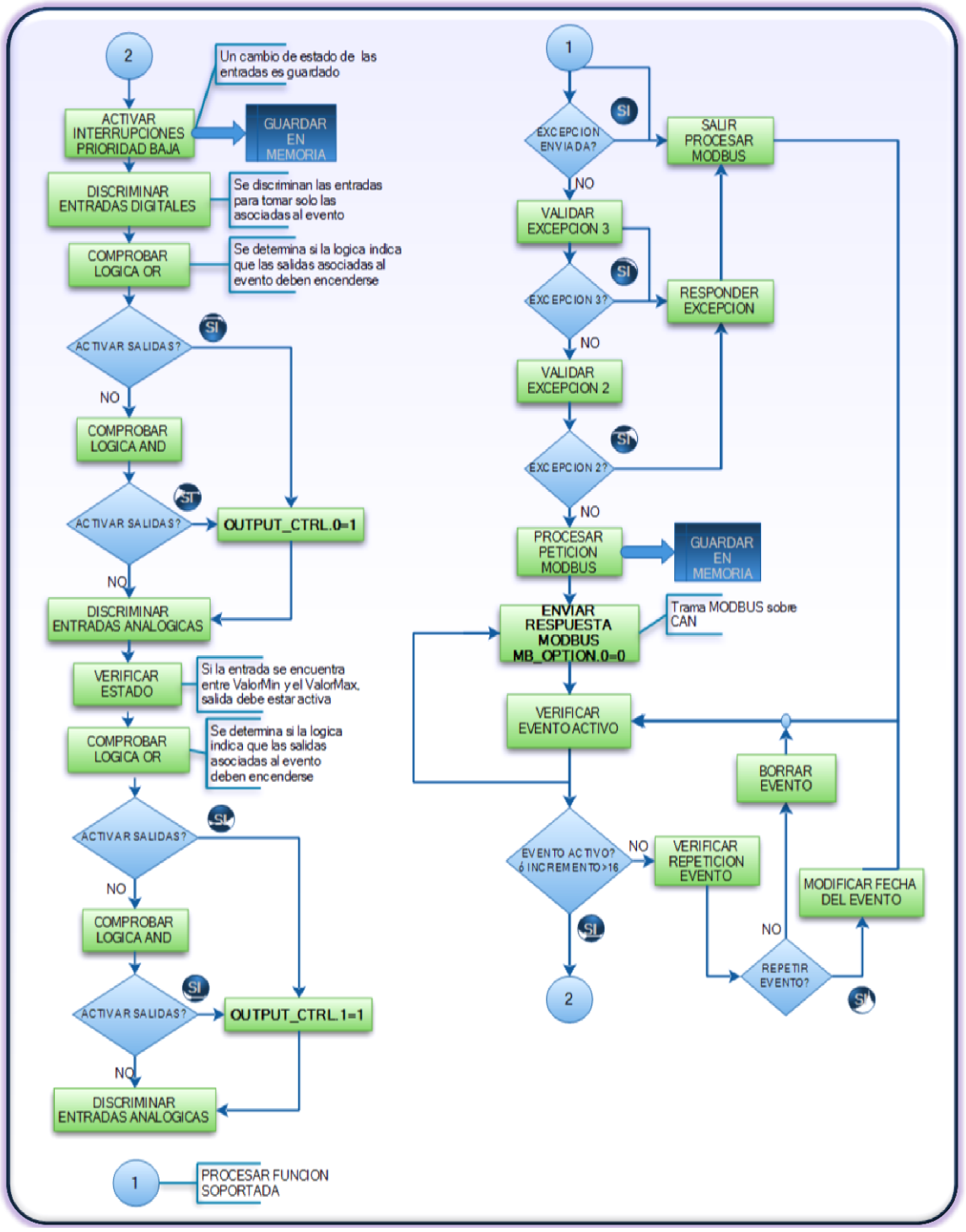


Figura 2.12 Flujo grama del controlador de propósito general (continuación)

2.1.2 Costo del controlador y tiempo de recuperación de la inversión

El diseño de la solución requiere que el sistema prototipo implementado tenga el menor consumo de potencia posible, por eso se tomaron en cuenta aquellas tecnologías que tuvieran las funcionalidades requeridas y que se implementaran en dispositivos de bajo consumo de energía. En la Tabla 2.13 se puede ver un cuadro resumen del consumo que genera el controlador de propósito.

Tabla 2.13 Consumo máximo de potencia del Controlador de Propósito General

DISPOSITIVO	POTENCIA DISCIPADA	
PIC18F258	1	W
MAX 127	0.762	W
MAX521	0.842	W
OTROS	0.5	W
Consumo máximo	3.104	W

El costo de implementar el controlador de propósito general[21] aproximadamente es de $200; ver detalle de costo en Tabla 2.14. Llevando a cabo un simple cálculo de recuperación de la inversión, asumiendo un ahorro energético de quince minutos diarios en cada uno de las 8 salidas del Controlador, con un factor de carga del 80% y tomando en cuenta los precios vigentes de la energía; se estima que el tiempo de recuperación de la inversión oscila en 12 meses. Ver Tabla 2.15.

Tabla 2.14. Costo total Controlador de Propósito General

Costo total del Controlador de Propósito General	**$**	**194.40**
Costo por Controlador de Propósito General	$	78.58
Costo Acondicionamiento de señales	$	115.82

[21] Ver anexo A3.3 para mayor detalle

Tabla 2.15 Cálculo de recuperación de Inversión.

Parámetro	Valor calculado	Unidades	Memoria de calculo
Carga por circuito (Relé 120V@5A)	0.48	KW	120*5*0.8
Carga por Controlador (8 relés)	3.2	KW	0.48*8
KWh al día de carga instalada (6 horas encendidas)	92.2	KWh	3.2*3600*6
Porcentaje de Perdida (15 min en cada circuito)	4%	1/4h	0.25h/6h
KWh perdidos al día	3.8	KWh	276.5*2%
costo KWh promedio Punta, Resto, Valle	0.15	$	$KWh(P,R,V)/3
Costo de la energía perdida	0.58	$ diarios	3.8*.15
Tiempo de recuperación de Inversión	333	dias	194.4/.58
	11.1	meses	222/30

2.1.3 Diagrama eléctrico del controlador de propósito general

Los diagramas del controlador de propósito general se presentan desde la Figura 2.13 hasta la Figura 2.19, ahí se puede ver las diferentes partes que componen el controlador. Se ha dividido según sus características en una parte se encuentra la parte principal donde se encuentra el micro controlador, luego se presentan la etapa de entradas y salidas digitales, así como también la etapa de potencia que maneja cargas de 120VAC, también las etapas de entradas y salidas análogas.

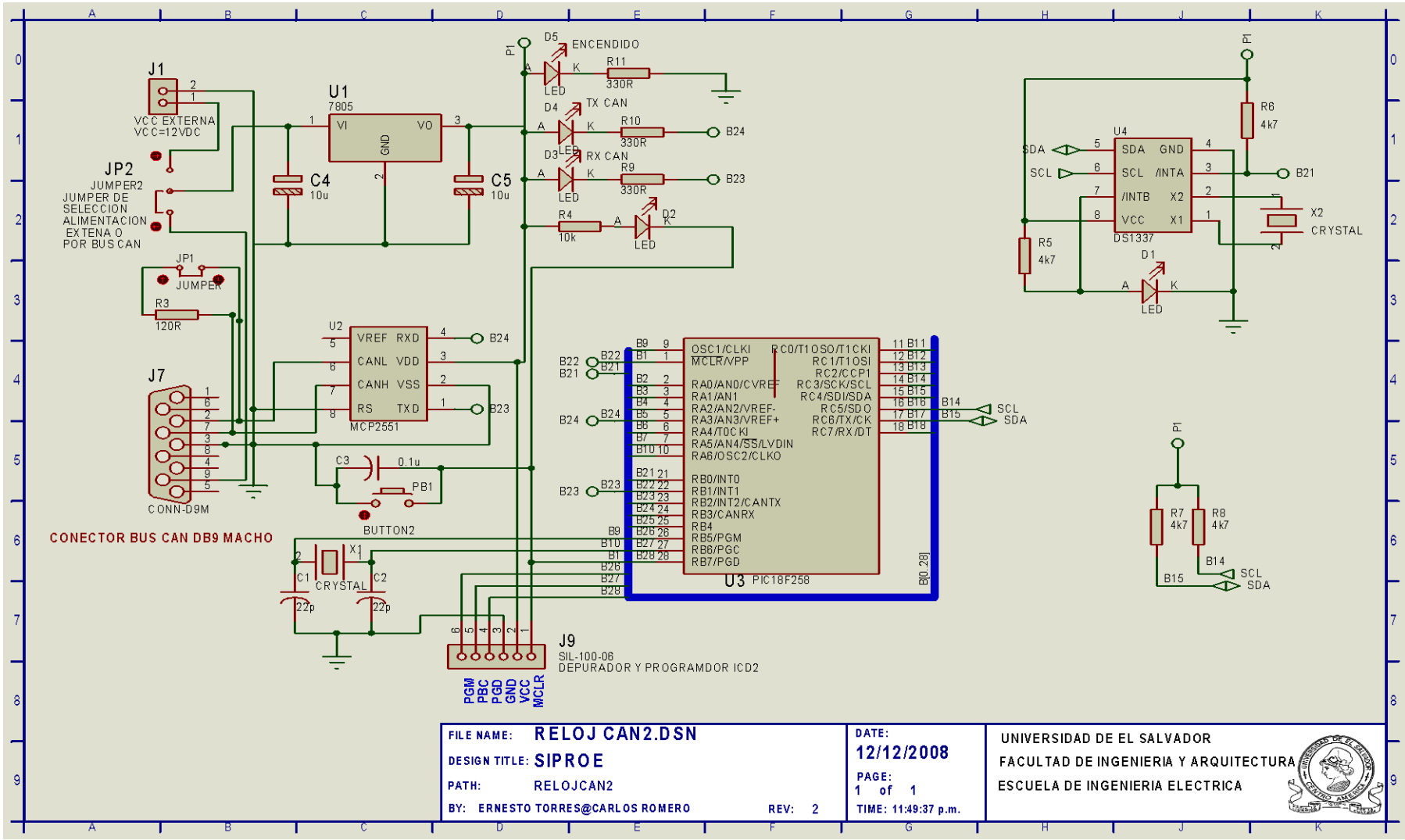


Figura 2.13 Esquema eléctrico del controlador de propósito general PIC18F258

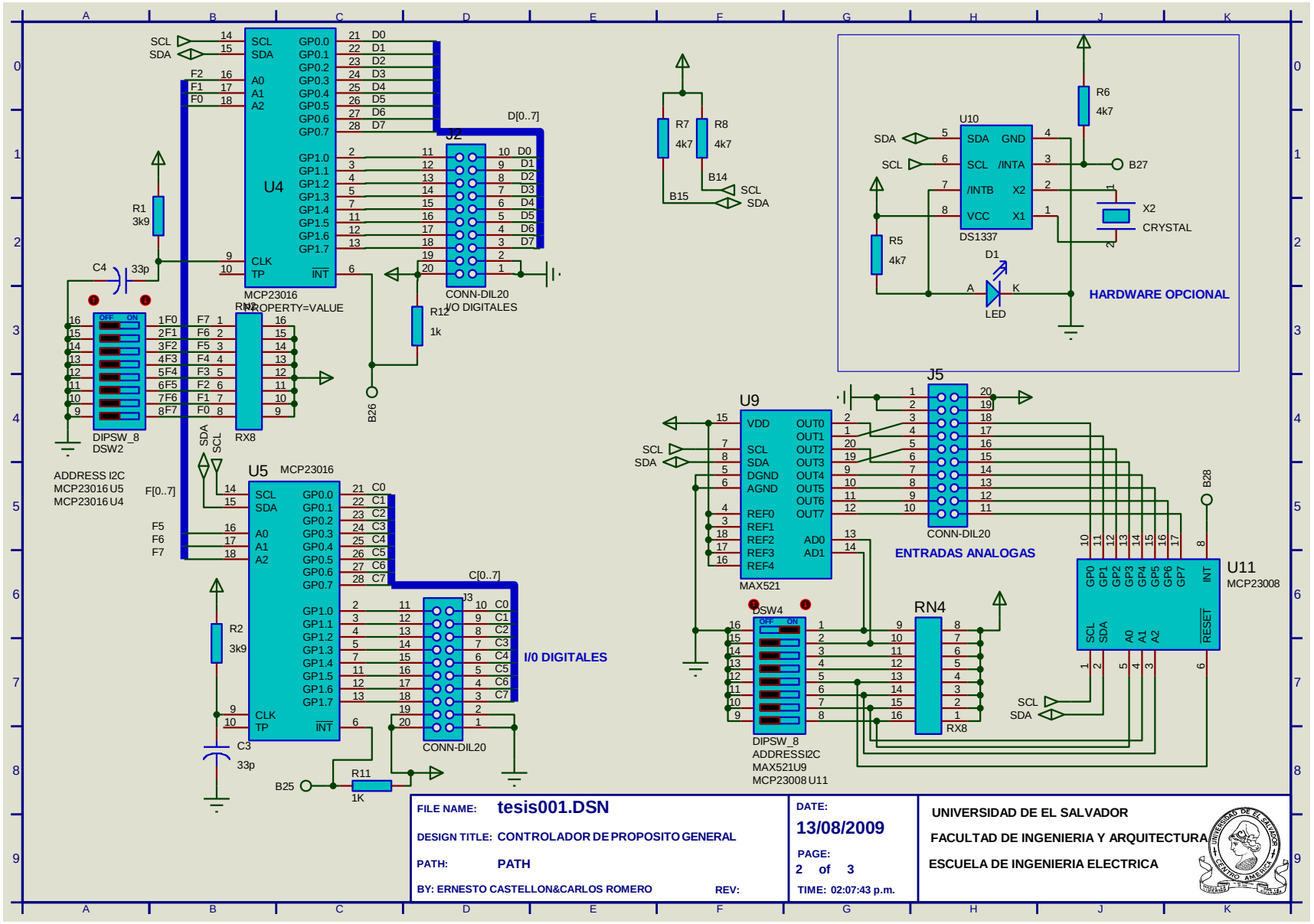


Figura 2.14 Diagrama eléctrico de expansores de entradas análogas MAX521

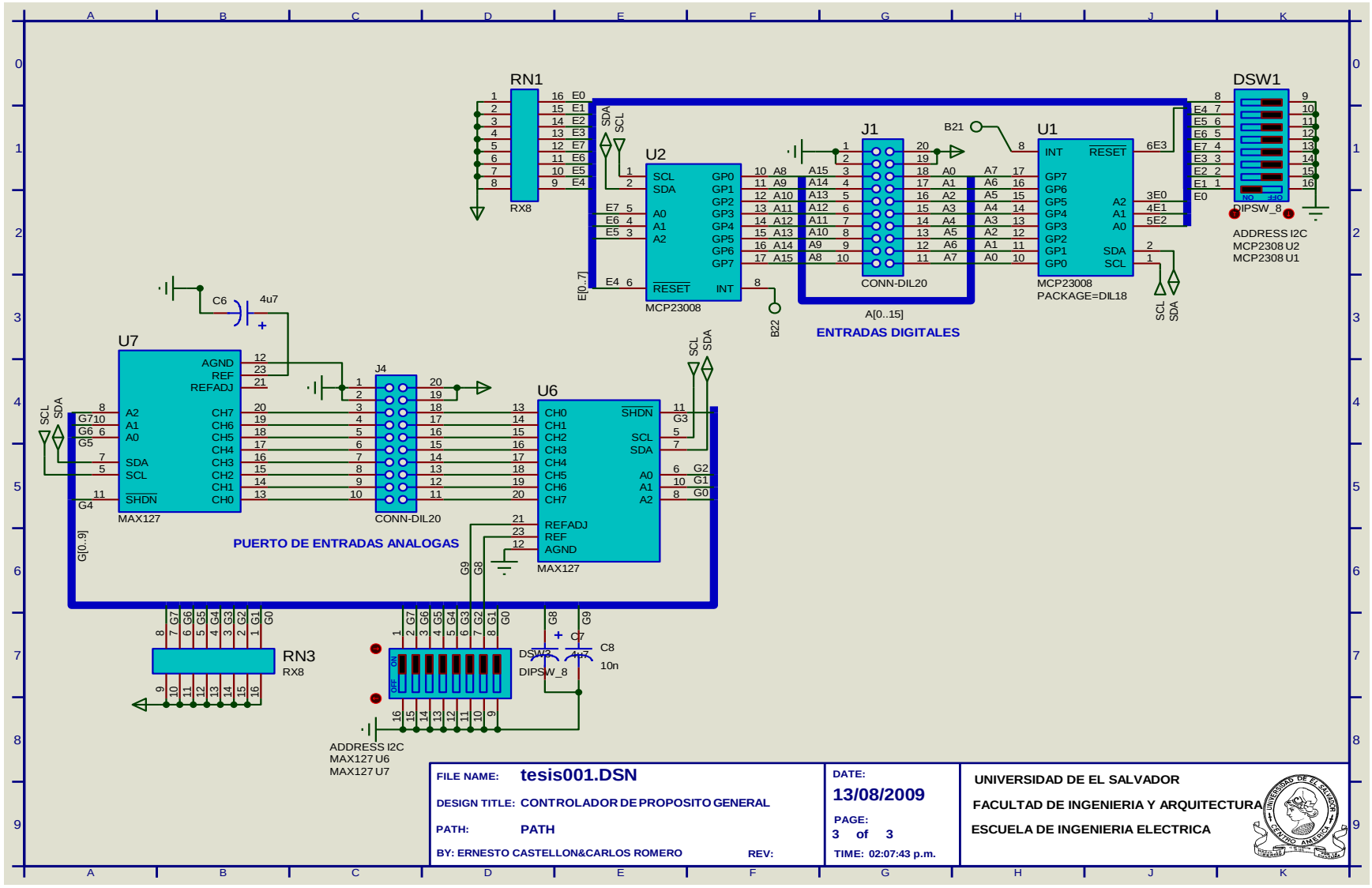


Figura 2.15 Diagrama eléctrico de conexión de los Relés de retención de doble bobina

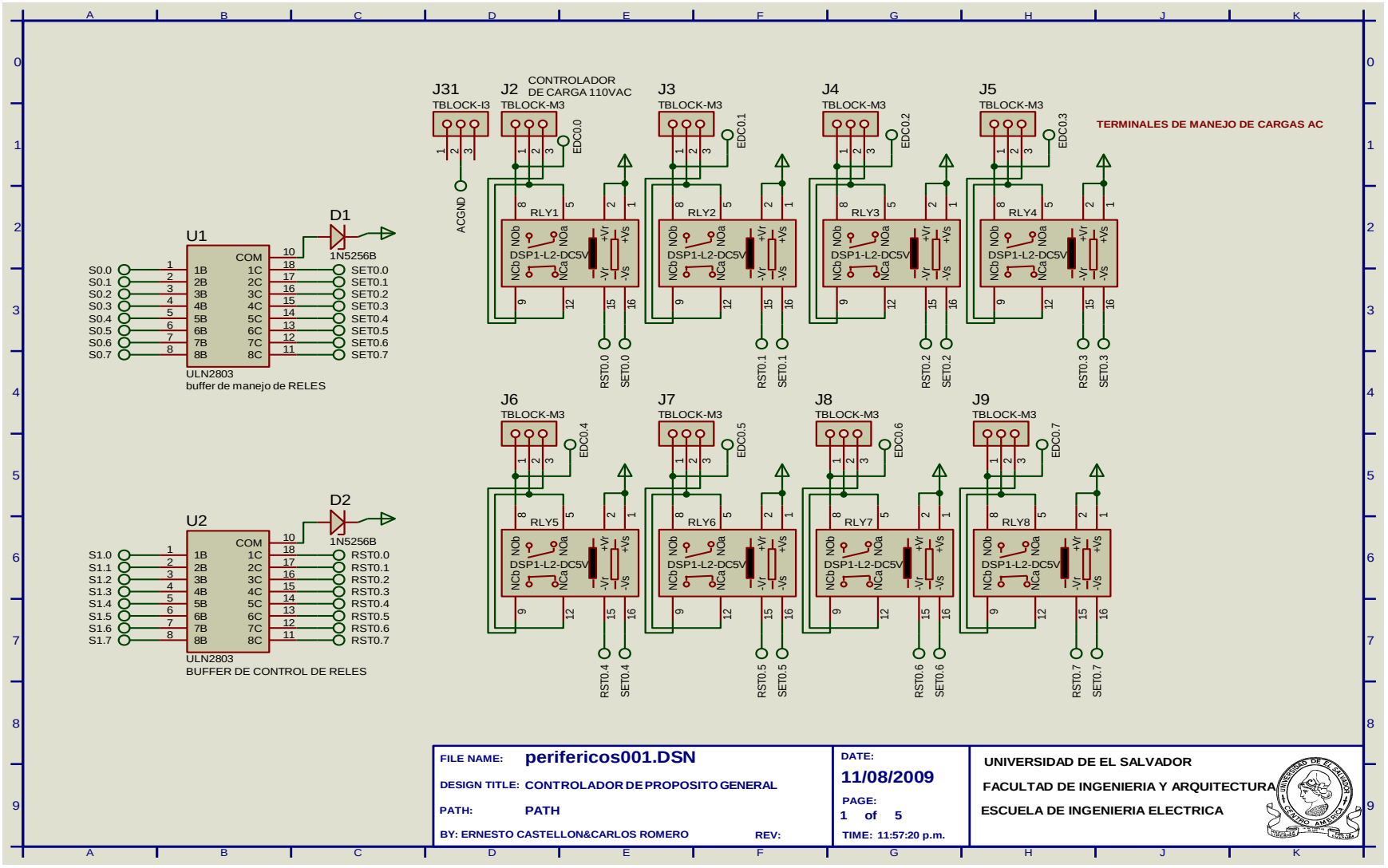


Figura 2.16 Diagrama eléctrico de conexión de los Relés de retención de doble bobina

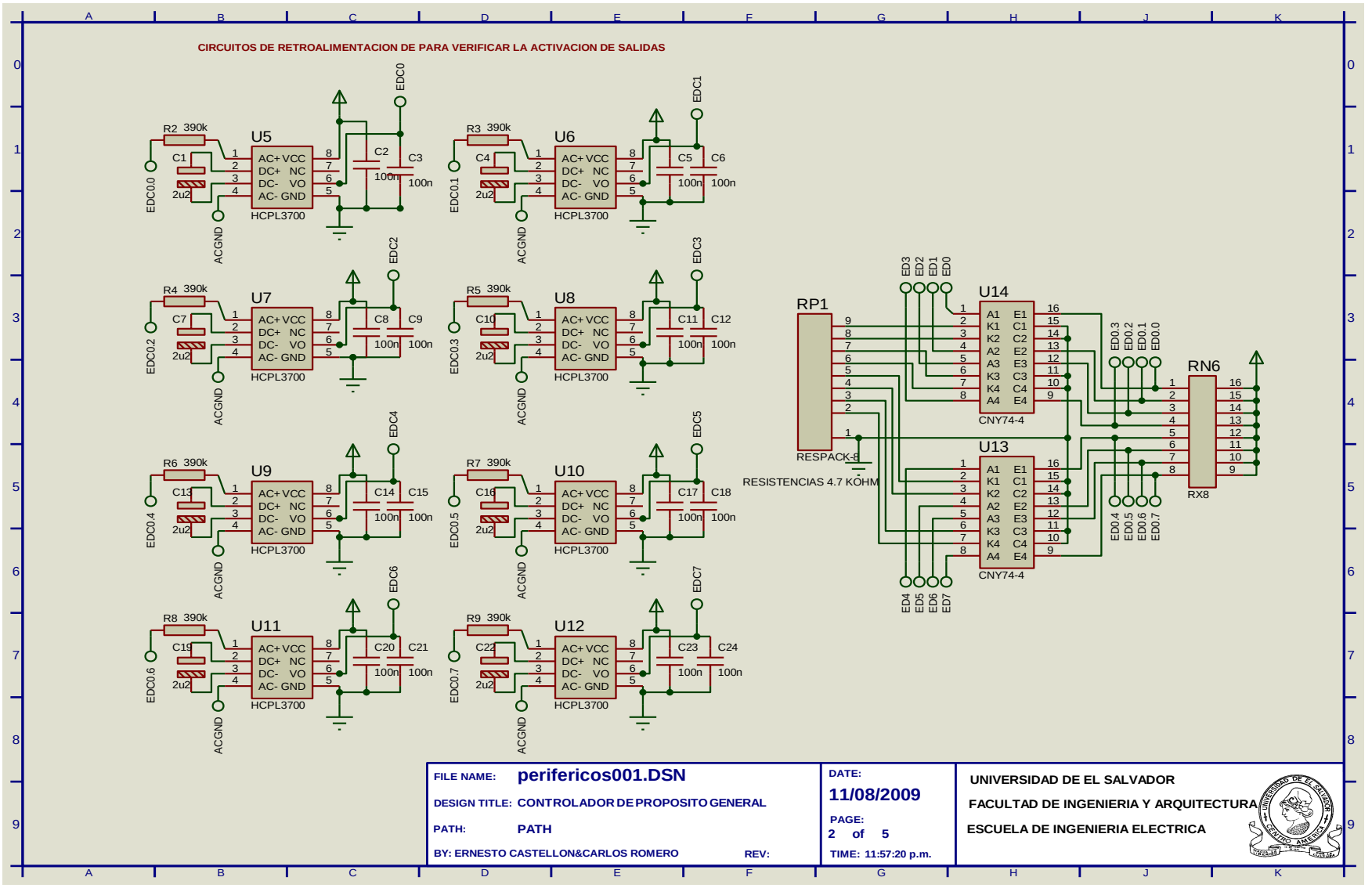


Figura 2.17 Diagrama Eléctrico de conexión de sensores de chequeo de estados digitales HCPL3700

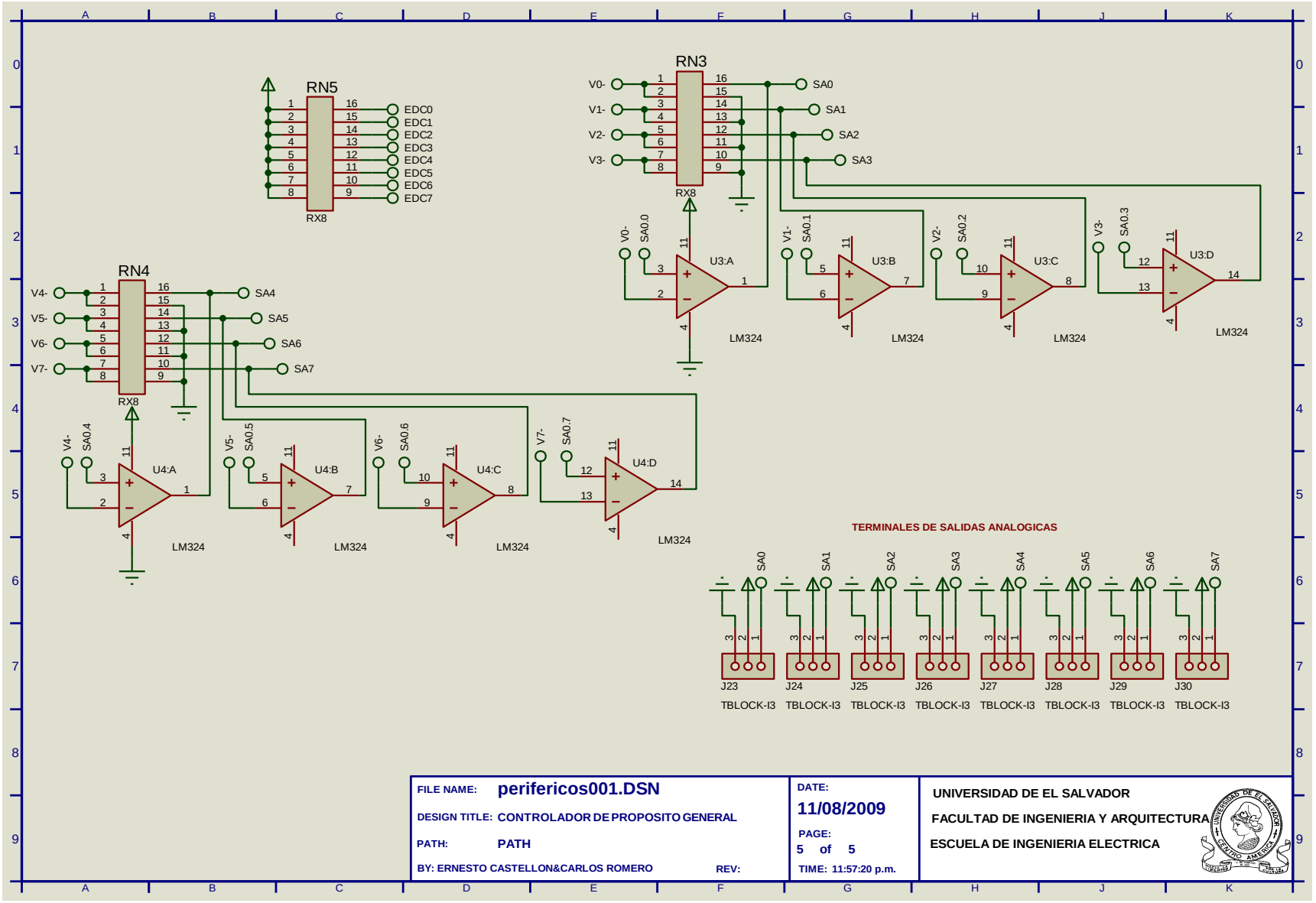


Figura 2.18 Diagrama eléctrico de acondicionamiento de señales análogas de entrada LM324

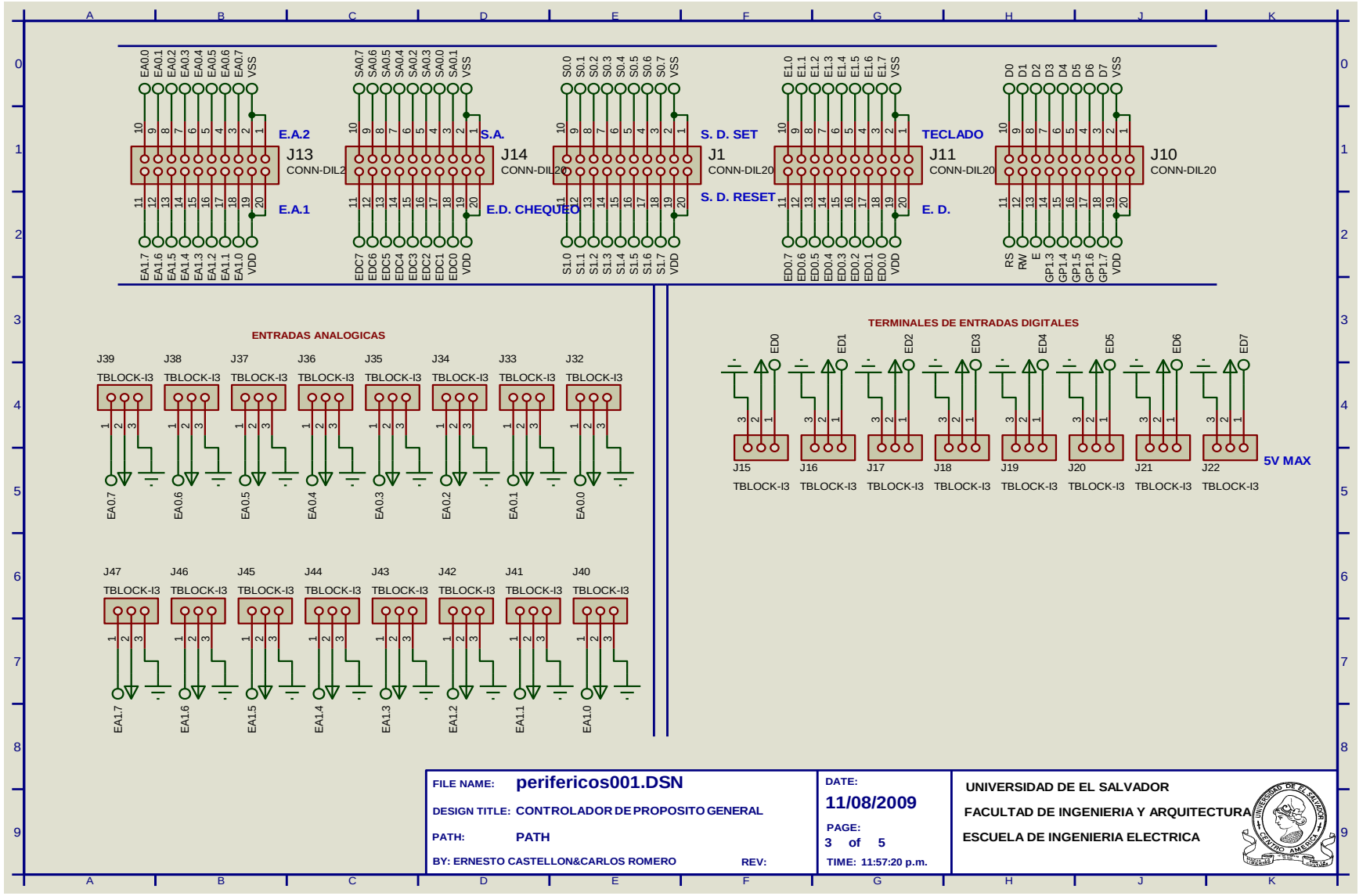


Figura 2.19 diagrama eléctrico de conexión de los bloques terminales de entradas análogas y digitales

2.2 Diseño e implementación del controlador inteligente de acceso

El control de acceso es necesario en cualquier empresa e instalación para restringir el acceso, tener control del personal y por medidas de seguridad.
El controlador inteligente de acceso se desarrolla con los criterios de portabilidad, fácil manejo y ampliable.

2.2.1 Diagrama en bloques

El diagrama en bloques del controlador de acceso se muestra en la Figura 2.20, como se puede ver consta de diferentes partes la tarjeta EEPROM, el lector de la tarjeta de acceso, interfaz de potencia, interfaz de comunicación CAN, etapa reguladora de voltaje y el micro controlador PIC18F258

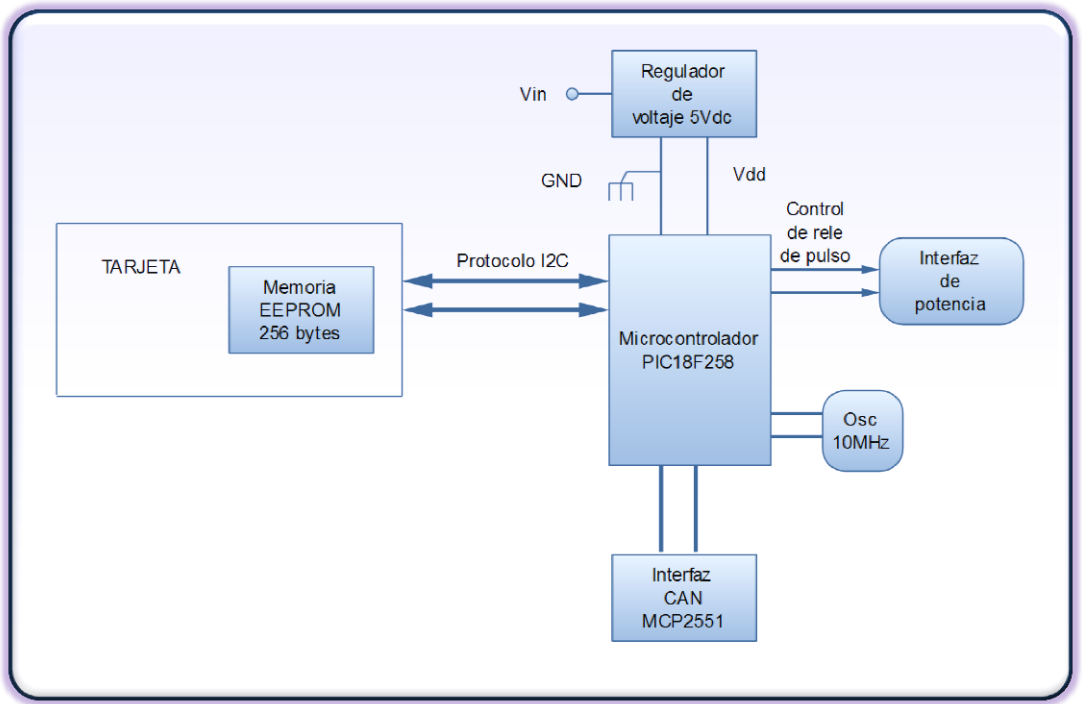


Figura 2.20 Diagrama en bloque controlador de acceso

La tarjeta de acceso es una memoria EEPROM[22] de 256 bytes en un encapsulado de tarjeta, esta tarjeta será utilizada como llave electrónica, es decir se almacenaran códigos de acceso en la memoria de la tarjeta; dándole al usuario acceso a las áreas previamente definidas por el administrador. El acceso se hace asignando claves de 64 bits para control de acceso o chapa electrónica. Obteniendo hasta 32 diferentes acceso con una sola tarjeta.

[22] Memoria programable y borrable eléctricamente

La Figura 2.21 muestra una tarjeta EEPROM de 256 bytes utilizada para control de acceso.

Figura 2.21 Tarjeta EEPROM de 256 bytes

Una tarjeta nueva presenta todos los valores de memoria seteados a 1 lógico. Las características de la tarjeta EEPROM se presentan Tabla 2.16

Tabla 2.16 Características de la tarjeta de acceso

Características
Voltaje de alimentación 3V-5.5VDC
Capacidad de memoria de 256 bytes
Interface I2C
Escritura secuencial de múltiples bytes
Escritura de páginas de memoria
Autoincremento de dirección de memoria
Retención de datos de 10 años
Borrado y escritura de un millón de veces

La forma de leer y escribir la tarjeta de acceso es por medio del bus I2C en la dirección de acceso única para todas las tarjetas de 1010000 estos son los bits más significativos del primer byte cuando el controlador empieza la comunicación con la memoria de la tarjeta, y luego se realiza una lectura o escritura secuencial de 8 bytes consecutivos los cuales son comparados con el ID configurado en duro en cada control de acceso. En la Figura 2.22 se muestra el diagrama lógico de contactos con que cuenta la tarjeta, la Tabla 2.17 muestra la descripción de cada uno de los contactos de la tarjeta.

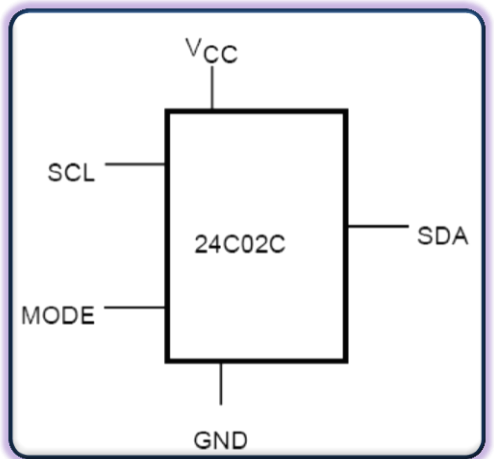


Figura 2.22 Diagrama lógico de la tarjeta de acceso

Tabla 2.17 Descripción de contactos de tarjeta de acceso

Nombre de contacto	Descripción
SDA	Serial de Datos I2C
SCL	Señal de reloj I2C
VCC	Alimentación de voltaje 5V
GND	Referencia de tierra de alimentación
MODE	Modo de lectura

Estos contactos tienen una posición específica en la tarjeta. La Figura 2.23 muestra la distribución de contactos.

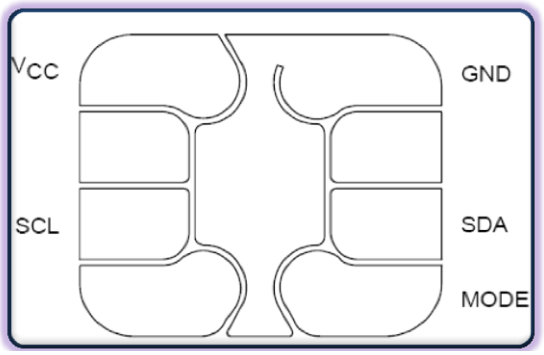


Figura 2.23 contactos de memoria EEPROM en la tarjeta de acceso

2.2.2 Firmware del controlador de acceso

El desarrollo del firmware del controlador de acceso, se hace igualmente que los otros controladores, se desarrolla el proyecto para este caso llamado: Acceso.mcp en MPLAB IDE y se incluyen los archivos listados en la Tabla 2.18 y Tabla 2.19

Tabla 2.18 Código de controlador de acceso

Archivo	Descripción
18F258TEMP.asm	Código del controlador acceso, desde aquí se llaman las librerías CAN, modbus, i2c y time.
Can18xx8.asm	Librería CAN proporcionada por MICROCHIP
funcionesModbusM0.asm	Librería modbus implementa funciones 02, 03, 05 y 16
I2C.asm	Librería I2C implementa comunicación con dispositivos I2C tarjeta EEPROM de acceso
Time.asm	Librería para manejar funciones de retardo y configuración de timer0 y timer1.

Tabla 2.19 Archivos inc del controlador de acceso

Archivo	Descripción
CAN18xx8.inc	Lista de funciones CAN que pueden ser utilizadas, solo es necesario incluir este archivo en el archivo .asm que se utiliza
CANDef.inc	Archivo donde se realizan las definiciones de variables, reserva de memoria y definición de macros utilizadas por la librería CAN
Definiciones1.inc	Archivo donde se definen métodos, macros y variables utilizadas por las librerías I2C, time y funcionesModbusM0
I2Clink.inc	Lista de funciones de la librería I2C que pueden ser utilizadas desde cualquier archivo .asm
MacrosI2C.inc	Definiciones de macros utilizadas en la librería I2C
Modbuslink.inc	Lista de funciones modbus que pueden ser utilizadas, para utilizar las funciones es necesario que en la librería estén definidas como global.
P18f258.inc	Definición de variables reservadas definidas por MICROCHIP
Time.inc	Lista de funciones de la librería time que pueden ser utilizadas.
Var_modbus.inc	Definición de variables utilizadas por la librería modbus
Variables.inc	Definición de variables generales

A continuación se presenta una parte del firmware que implementa la característica de plug and play del controlador de acceso.

```
MsgID:
                movlw   low(CANDt1)
                movwf   FSR0L
                movlw   high(CANDt1)
                movwf   FSR0H
                movlw   0xFF
                movwf   POSTINC0
                movlw   0x06            ;ID del controlador a registrar en el modulo maestro
                movwf   POSTINC0
    ;CANSendMessage   msgID, DataPtr, DataLngth, Flags
     CANSendMessage  0x33,CANDt1,2,CAN_TX_XTD_FRAME
                addlw   0x00                    ;Check for return value of 0 in W
                bz      MsgID
```

Cuando el controlador de acceso es energizado, este espera un tiempo y luego envía un mensaje por el bus CAN al controlador maestro para registrarse e informar que está disponible. La función de la librería CAN para enviar mensajes es CANsendMessage, la cual toma como parámetros el ID del filtro a quien va dirigido, un buffer de datos a enviar y el tipo de mensaje; además se desarrolla la sección más importante del firmware en los flujos gramas presentados en la Figura 2.24 y la Figura 2.25.

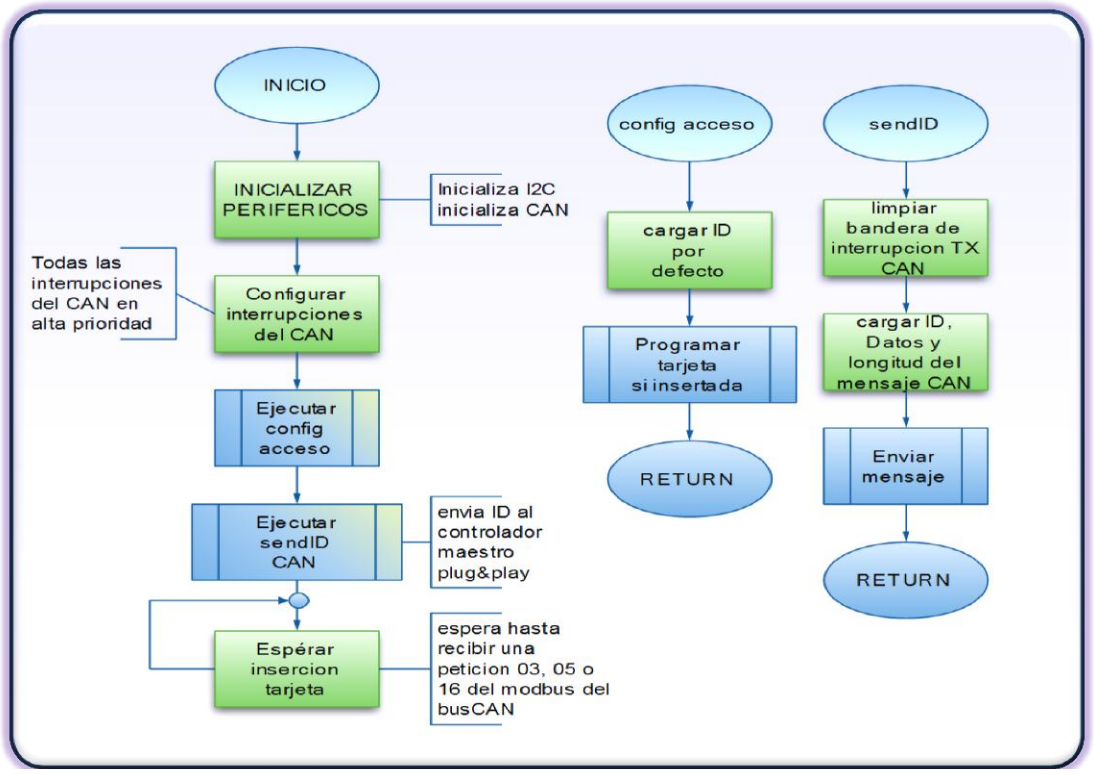


Figura 2.24 Flujo grama de control de acceso

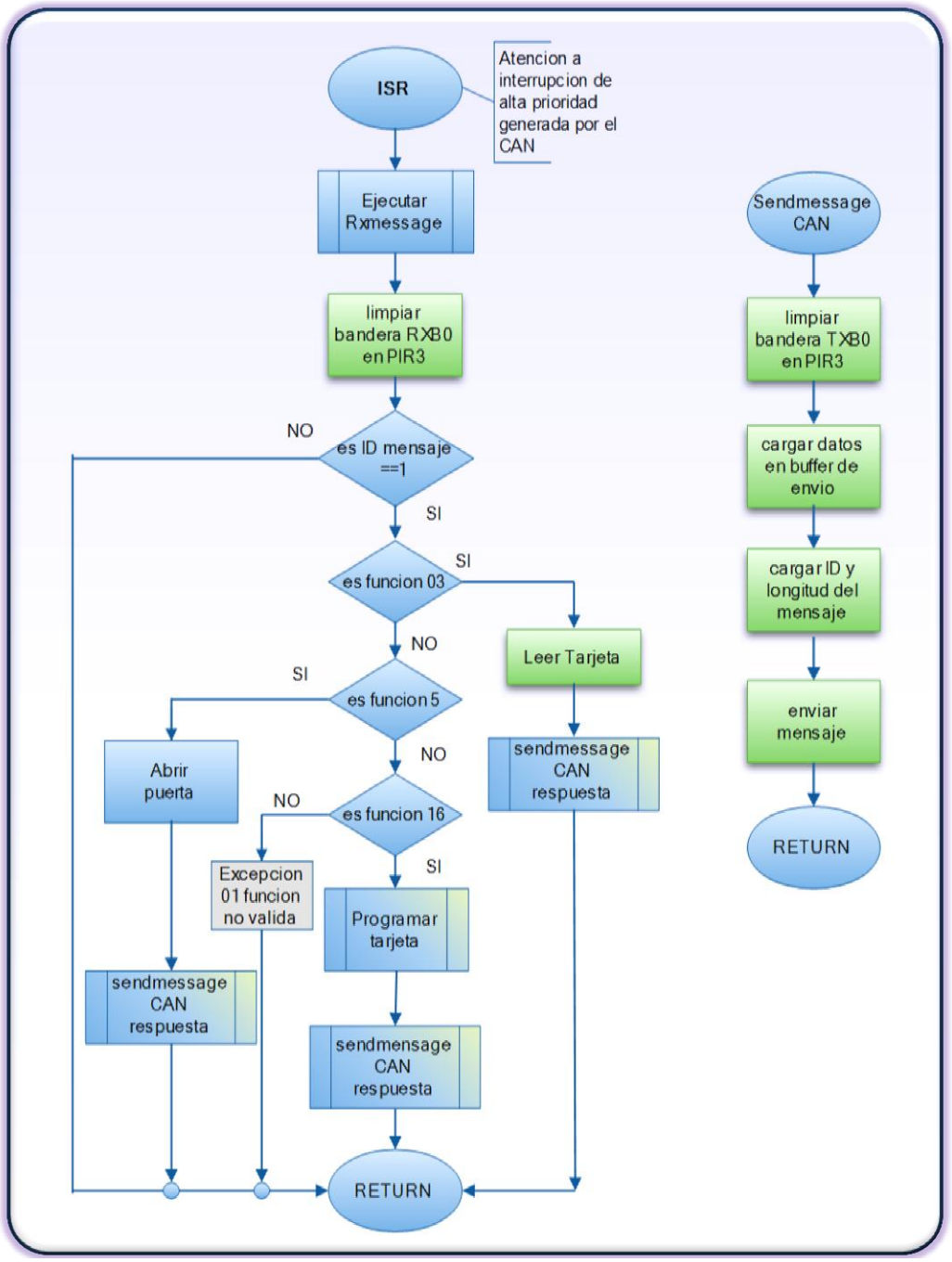


Figura 2.25 Flujo grama de controlador de acceso (continuación)

2.2.3 Diagrama eléctrico de controlador de acceso

A continuación se presenta el diagrama eléctrico del controlador de acceso

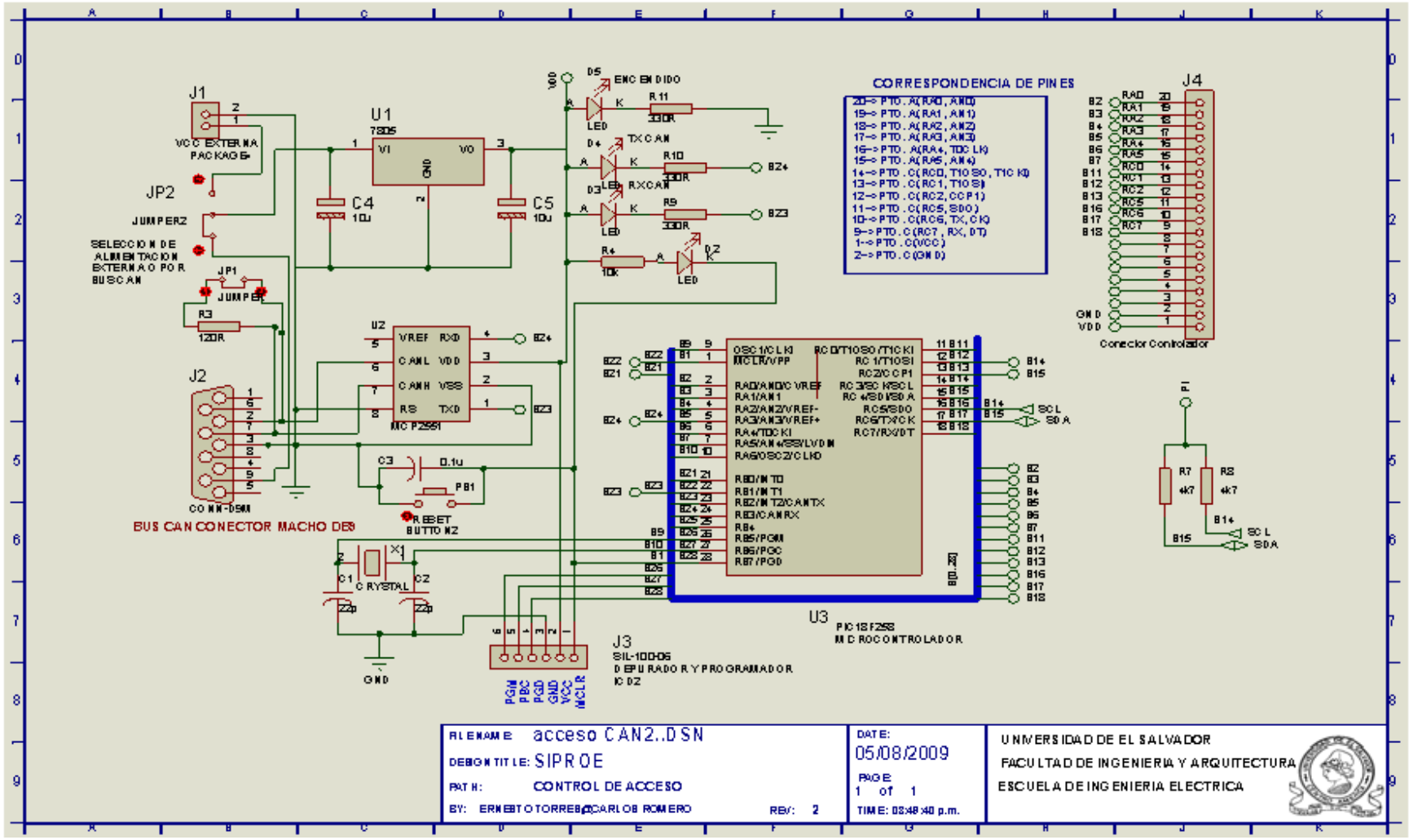


Figura 2.26 Diagrama eléctrico de controlador de acceso

CONCLUSIONES DEL CAPITULO II

- El desarrollo de la mayor parte del firmware de SIPROE está estructurado como librería, para facilitar el desarrollo del resto del firmware, que funciona como aplicación en los diferentes controladores.

- El sistema de acceso es factible de usar en cualquier instalación dando al usuario una alternativa para accesar a áreas restringidas.

- El diseño de equipos basados en micro controlador permite dar solución, a muchas necesidades en la rama de electricidad a donde se requiere un uso optimo de recursos.

- El uso de un bus de comunicación en el diseño e implementación de hardware permite modularizar el circuito, haciendo fácil su mantenimiento.

- El protocolo CAN permite optimizar la comunicación entre controladores, dando la facilidad de agregar más dispositivos al bus, sin que este pierda su rendimiento.

- Los expansores MCP2308 Y MCP23016 permiten manejar 8 Y 16 salidas/entradas respectivamente, son una solución muy apropiada cuando se requiere el manejo de muchas salidas/entradas digitales.

- El controlador de reloj calendario es una solución práctica para compartir recursos de horas y fechas necesarios en un sistema a donde se requiere manejar eventos en tiempo real.

- El protocolo industrial MODBUS permite manejar dispositivos remotos independiente del medio que se utilice para hacer peticiones, se TCP/IP, serial o CAN

- El control de los sistemas de iluminación y aire acondicionado son requeridos para optimizar el uso de la energía eléctrica.

CAPITULO III

DISEÑO Y DESARROLLO DEL SITIO WEB

Introducción.

En este capítulo 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 sistemas de iluminación, aire acondicionado y control de acceso que son las diferentes opciones que el sistema SIPROE proporciona. Para con el fin de ahorrar energía eléctrica

3.1 FUNDAMENTOS DE DISEÑO: ENFOQUE SOFTWARE

Retomando la funcionalidad de su versión predecesora, el módulo electrónico SACME V1.0, el prototipo SIPROE V1.0 cuenta con una arquitectura cliente servidor, la cual a través de un aplicativo WEB desarrollado en ambiente JAVA y con funcionalidad de cliente, denominado CLIENTE SIPROE, ejecuta peticiones hacia y recibe respuestas desde un controlador maestro denominado CONTROLADOR SIPROE M0, el cual lleva a cabo las siguientes funcionalidades:

- Sirve como GATEWAY o puente de enlace entre el CLIENTE SIPROE y los controladores de funcionalidad, los cuales se describirán más adelante.
- Ejecuta la funcionalidad de servidor MODBUS, recibiendo peticiones MODBUS, las cuales al ser recibidas por el CONTROLADOR SIPROE M0, son transmitidas tanto al modulo SACME V1.1 como a controladores de funcionalidad para ser ejecutadas por estos.
- Una vez ejecutada la petición MODBUS, por el modulo SACME V1.1 o los controladores de funcionalidad, el CONTROLADOR SIPROE M0 transmite las respuestas MODBUS hacia el cliente.
- Aunque la arquitectura maestro esclavo no es aplicable a una red CAN, al haberse desarrollado el protocolo MODBUS sobre CAN, el CONTROLADOR SIPROE M0, es en esencia un controlador MAESTRO dentro de la arquitectura de SIPROE V1.0; ya que se encarga de re direccionar las funcionalidades implementadas de MODBUS hacia los controladores de funcionalidad y recibir de estos la respuesta a la petición del CLIENTE SIPROE.
- Ejecuta las funcionalidades de administración de la red CAN.

El prototipo SIPROE está diseñado con tecnología JAVA para entornos empresariales, en tal sentido cuenta con una arquitectura de diseño multicapas, definida como se muestra en la Figura 3.1

Adicionalmente SIPROE incorpora parte del hardware de SACME tal como el Gateway MODBUS TCP/RTU para la comunicación con el cliente del sistema (usuario); El controlador maestro incorpora COMPONENTES de SACME para el manejo completo de las funcionalidades de SACME y un transceptor CAN para la comunicación con los demás controladores de funcionalidad; cada controlador de funcionalidad adicionalmente incorpora un transceptor individual que lo conecta al bus CAN y le permite comunicarse a través de ese medio físico a los demás controladores. Se observa en la Figura 3.1 que la interface I2C proporciona a cada controlador de funcionalidad la posibilidad de comunicarse con el entorno físico para el cual está diseñado.

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.

3.1.2 Escalabilidad

Es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para 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.

3.1.3 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.

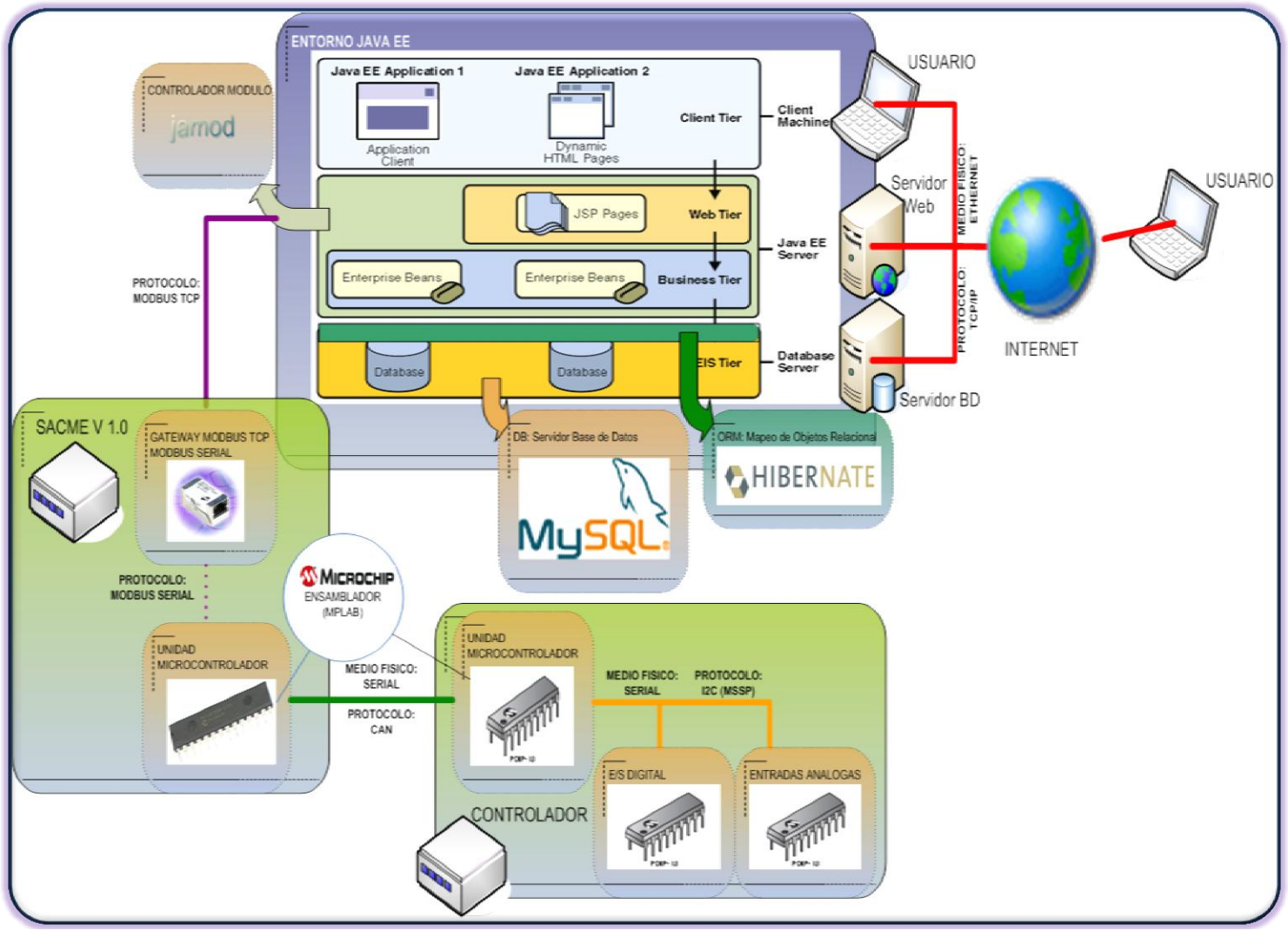


Figura 3.1 Entorno de la aplicación visto desde la perspectiva Software

3.1.4 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.

3.1.5 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 no apropiado de esta técnica puede y suele llevar a un consumo excesivo de los recursos del sistema (memoria).

3.1.6 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.

3.1.7 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.

3.1.8 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[23].

3.1.9 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 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:

[23] **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.

3.1.10 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.

3.1.11 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.

3.1.12 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[24] con presentación ofrecerá a esta los servicios que se requieran para el funcionamiento de la microaplicación en sí, y por otro, conjunto de servicios serán publicados por medio (recomendablemente) de servicios web. Estos últimos estarán orientados a la colaboración con otros sistemas

[24] 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.

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

3.1.13 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.

3.1.14 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).

3.1.15 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[25] 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

[25] 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.

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

- Servicio de Log.
- Pool de conexiones JDBC[26] (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.

3.2 Implementación del Sitio Web

Para implementar el sitio web de Siproe que tiene las posibilidades de administrar, controlar y monitorear los recursos con funcionalidad de iluminación, aire acondicionado y acceso con el fin de ahorrar 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[27] 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 más robustas debido al encapsulamiento, mantenimiento y soporte más sencillo, mayor

[26] **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.

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

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[28] 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.

3.3 Esquema del Sitio Web

A continuación se presenta la Figura 3.2 del sitio web y las entidades que implementa describiendo cada uno de los componentes.

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

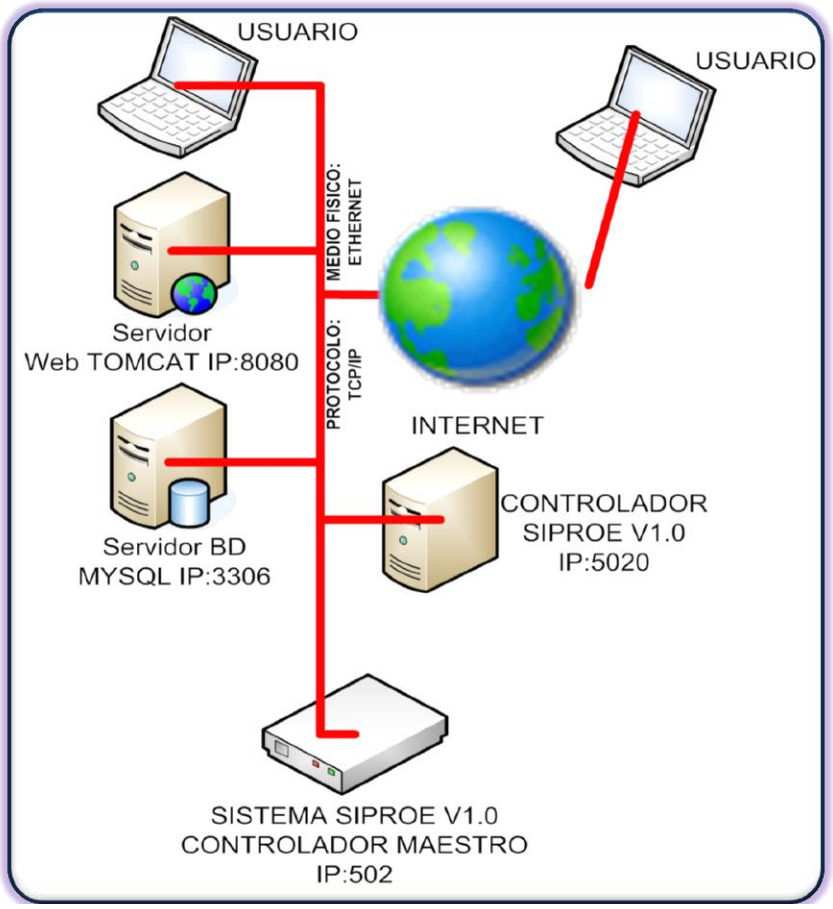


Figura 3.2 Esquema del sitio web.

3.3.1 El Cliente

Aunque el cliente no forma parte de la implementación como tal, 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 SIPROE, 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[29] 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 SIPROEv1.0 en Tomcat, se puede acceder al sistema SIPROE a través de una interface grafica de usuario desde la Web, así un usuario podrá hacer uso del sistema cargando una página web a través del navegador con la url: http://IpHost:8080/SIPROEv1.0/, en la tabla siguiente se describe el significado de cada parámetro

Tabla 3.1 Detalle de la url http://IpHost:8080/SIPROEv1.0/

Parámetro	Descripción
IpHost	Es la dirección IP de la PC donde está instalado tomcat.
8080	Es el puerto por defecto en el cual Tomcat escucha peticiones http.
SIPROEv1.0	Es el nombre de la aplicación que navega el sitio web del sistema SIPROE.

[29] 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.

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 siproe es persistir tablas de horarios de los eventos programados.

3.3.4 Servicio Controlador del sistema inteligente siSiproe

El Servicio Controlador al sistema inteligente; 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 controlador inteligente maestro.
De aquí en adelante llamaremos siSiproe al Servicio Controlador del Sistema inteligente.

3.4 Comunicación con el siSiproe

Este tiene la particularidad de ser servidor al lado de la aplicación web y cliente Modbus/tcp al lado del controlador inteligente maestro.

Para entender con claridad como interactúa siSiproe tanto con la aplicación web como el mismo controlador inteligente, se presenta a continuación el Diagrama de Secuencia.

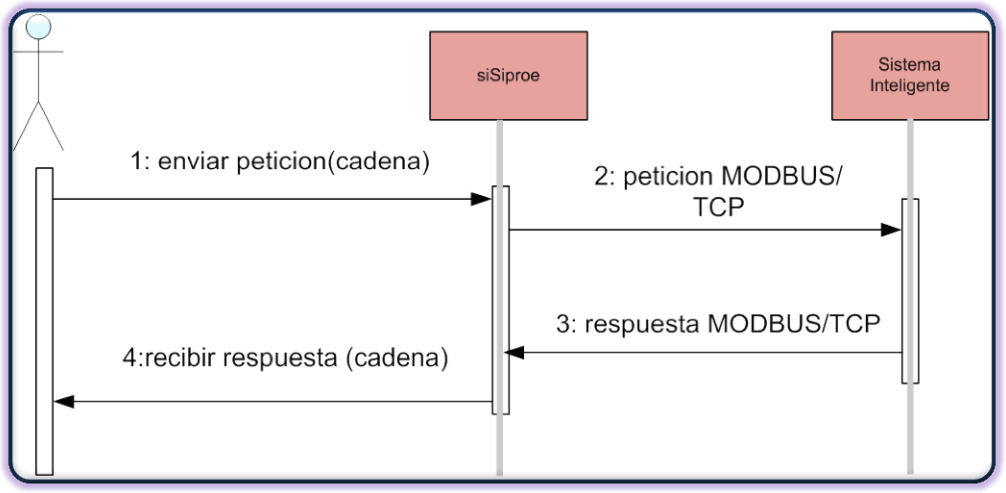


Figura 3.3 Diagrama de Secuencia de comunicación con siSiproe.

Para establecer comunicación con siSiproe, 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 controlador inteligente maestro también proporcionada por la solicitud.

La solicitud tiene el formato siguiente:

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

Tabla 3. 1 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, 03, 05 y 16.
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 carácter "#"	Sirve como separador de parámetros en la cadena solicitud.

siSiproe está compuesto por un conjunto de tres clases: Siproe_SI.class, NuevoCliente.class y ModbusFunc.class.

Siproe_SI.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 vez, inicia un nuevo hilo que le da seguimiento a la solicitud. Cuando Siproe_SI 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 sistema inteligente 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], read register[03], Write Single Coil [5] y write register[16]) haciendo uso de la librería del proyecto JAMOD[30] 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.

siSiproe 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 siSiproe es mostrado a continuación.

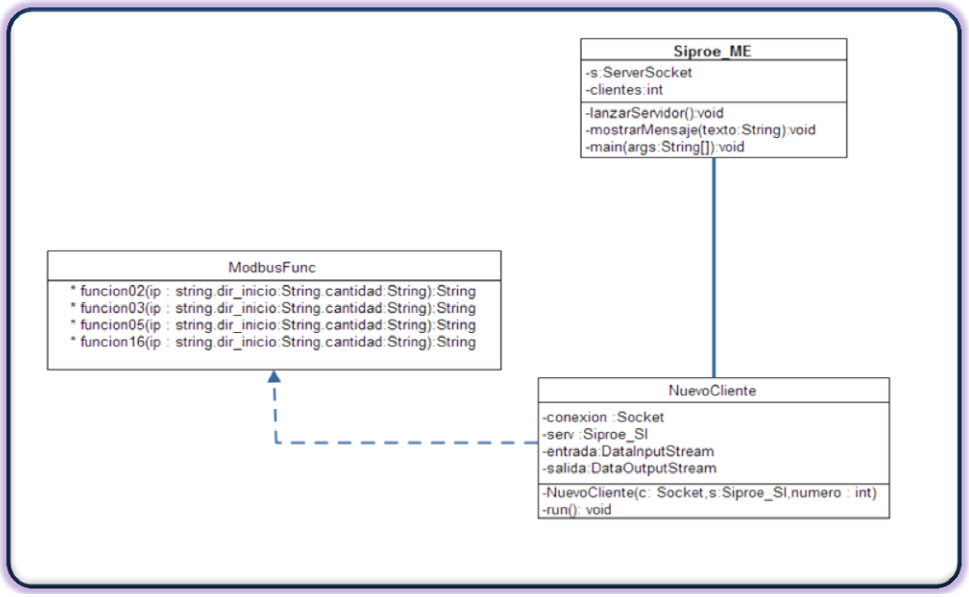


Figura 3.4 Diagrama de Clase para siSiproe

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.

[30] 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)

3.5.1 Arquitectura N-Capas.

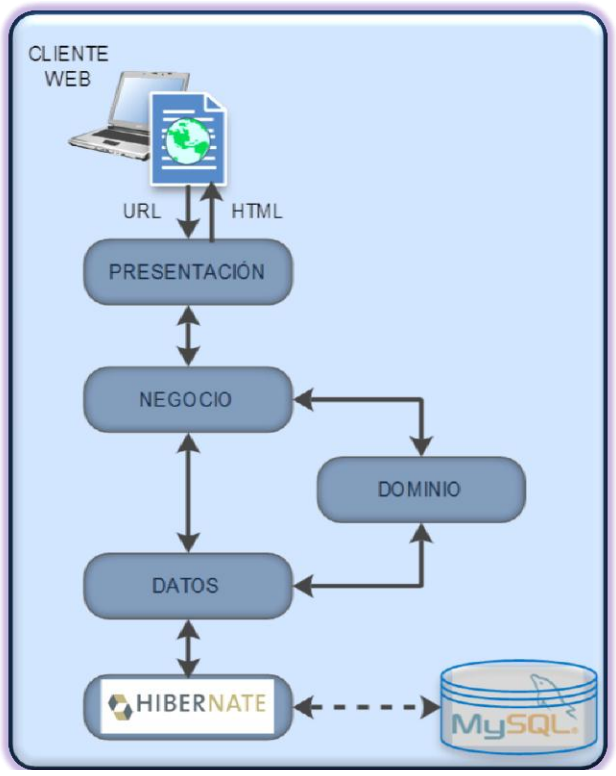


Figura 3.5 Arquitectura N-Capas

3.5.2 Cliente Web

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

3.5.3 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 está 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.

3.5.4 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. Estas son las que realmente suponen el motor del sistema, dado que se basan en el funcionamiento del modelo real.

Está 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.

3.5.5 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.

3.5.6 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.

3.5.7 El Motor de Persistencia Hibernate.

Hibernate es una herramienta ORM[31] (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 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

[31] 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.

dentro del grupo JBoss[32] 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 fábrica 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 se comienza creando el archivo de mapeo.

3.5.7.1 Archivos de Correspondencia

Los archivos de correspondencia o mapeo son archivos XML, con los que 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 controladores inteligentes que son adoptados por siproe 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.siproe.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 mapeo modulo.hbm.xml

[32] **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, estos archivos son listados en la Tabla 3.2.

Tabla 3.2. Archivos de correspondencia de hibernate

Archivo de Mapeo	Clase Java	Tabla MySQL
Aula.hbm.xml	Aula	aula
Departamento.hbm.xml	Departamento	departamento
EventoAnalogo.hbm.xml	EventoAnalogo	evento_analogo
EventoDigital.hbm.xml	EventoDigital	evento_digital
InputAna.hbm.xml	InputAna	input_ana
InputDig.hbm.xml	InputDig	input_dig
LlaveAcceso.hbm.xml	LlaveAcceso	llave_acceso
LogicaAnaloga.hbm.xml	LogicaAnaloga	logica_analoga
LogicaDigital.hbm.xml	LogicaDigital	logica_digital
Modulo.hbm.xml	Modulo	modulo
OutputAna.hbm.xml	OutputAna	output_ana
OutputDig.hbm.xml	OutputDig	output_dig
RelojRtc.hbm.xml	RelojRtc	reloj_rtc
Rol.hbm.xml	Rol	rol
Submodpg.hbm.xml	Submodpg	submodpg
Tarjeta.hbm.xml	Tarjeta	tarjeta
Tipo.hbm.xml	Tipo	tipo
Usuario.hbm.xml	Usuario	usuario

3.5.7.2 Configuración de Hibernate

Hibernate está 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

3.5.7.3 Crear una fábrica 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.

3.5.7.4 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 fábrica 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 3.3 Propiedades **de configuración de 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	Número máximo de conexiones concentradas

Estas propiedades se pueden configurar por medio del 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 Siproe se muestra en la Figura 3.7.

```
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.url">jdbc:mysql://localhost/siproe</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.

Debido a que se está 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 con el compatible al servidor de la base de datos que se este utilizando.

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

3.5.7.5 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:

3.5.7.6 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).

Tabla 3.4 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

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

3.5.7.7 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.

3.5.7.8 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 interface "Query"
La interface 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

3.6 La Interfaz de Usuario

Cuando un usuario accede al sitio web del sistema SIPROE a través de un navegador de páginas 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 página de inicio del SIPROE 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 está 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 encriptado como texto plano en la base de datos.

Una vez identificado el usuario, se le presentan de manera gráfica las opciones con las que puede interactuar, según el rol asignado de la sesión que se use.

3.6.1 Funcionalidad de monitoreo

La funcionalidad de monitoreo se puede realiza en el sistema inteligente, desde la aplicación web propietario, pero el sistema queda abierto para poder ser acezado por medio de otra interface por ejemplo Labview. Con esta funcionalidad se puede monitorear los siguientes controladores inteligentes:

- **Controlador inteligente reloj de tiempo real:** al monitorear este controlador se puede ver la fecha y la hora a la cual está configurado, con esto se monitorea la hora que se tiene en el bus CAN, y se verifica si necesita ser actualizada.

- **Controlador inteligente de control acceso:** debido a que el controlador de acceso y seguridad, es para brindar seguridad y mayor control solo puede ser monitoreado por el administrador del sistema, mostrando el ID del controlador, si la puerta está abierta o cerrada, el código de 64 bits de la ultima tarjeta insertada.

En la Figura 3.9 se muestra el diagrama de secuencia de monitoreo y se explica en el siguiente Ejemplo: se requiere monitorear un recurso se selecciona de la lista disponible, luego se presiona el botón “Ver Estado” que está 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 siSiproe escucha, este a su vez 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 controlador inteligente haga la acción sobre el recurso especificado, la respuesta es presentada al usuario en la interfaz grafica.

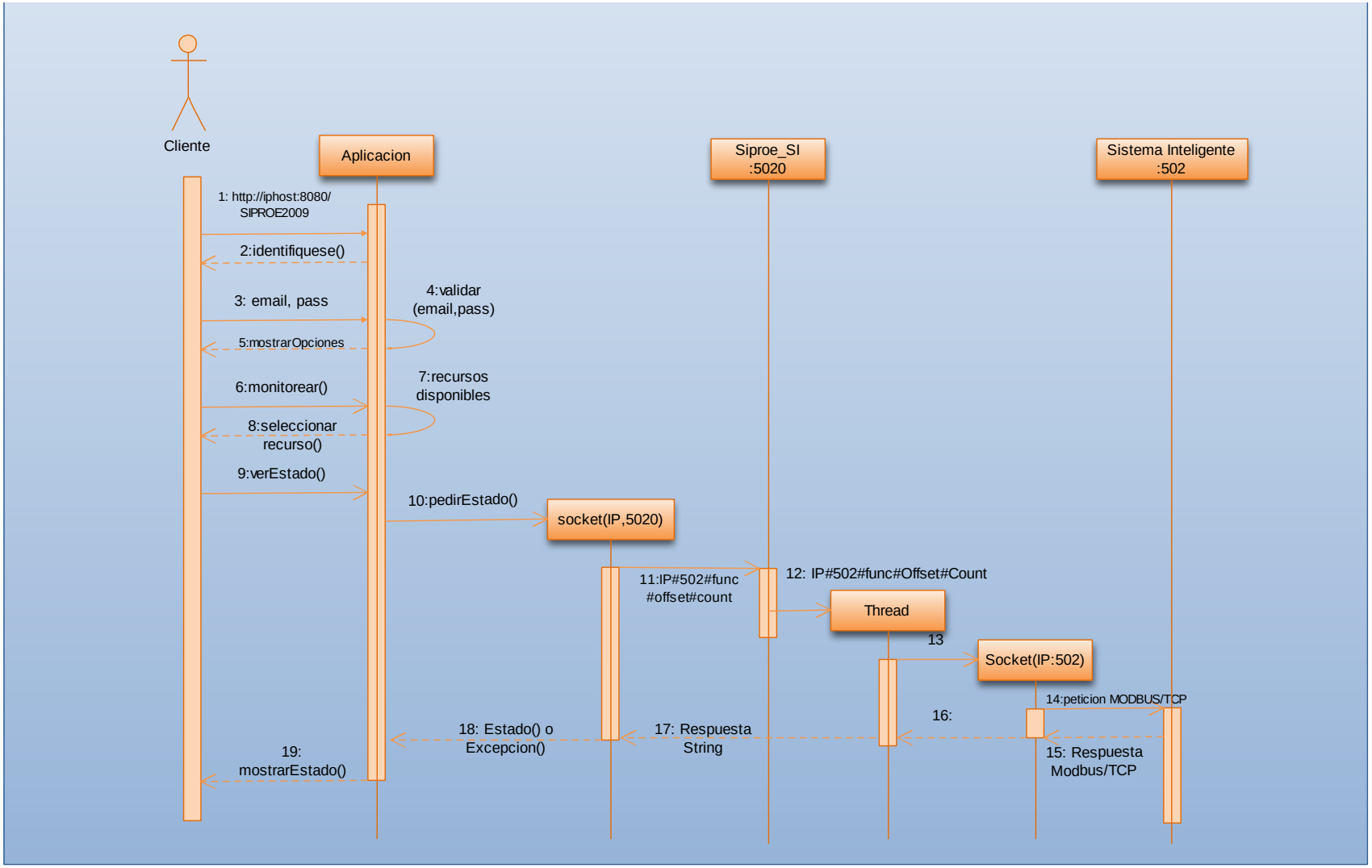


Figura 3.9 Diagrama de Secuencia de monitoreo

3.7 Diagrama Entidad/Relación

La información que se guarda en la base de datos necesaria para el buen funcionamiento de SIPROE, está distribuida en entidades que se encuentran relacionadas entre sí. Las asociación de las entidades van de acuerdo a las necesidades con que fue diseñado SIPROE, 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 en la Figura 3.10.

El archivo **siproe**.sql 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.

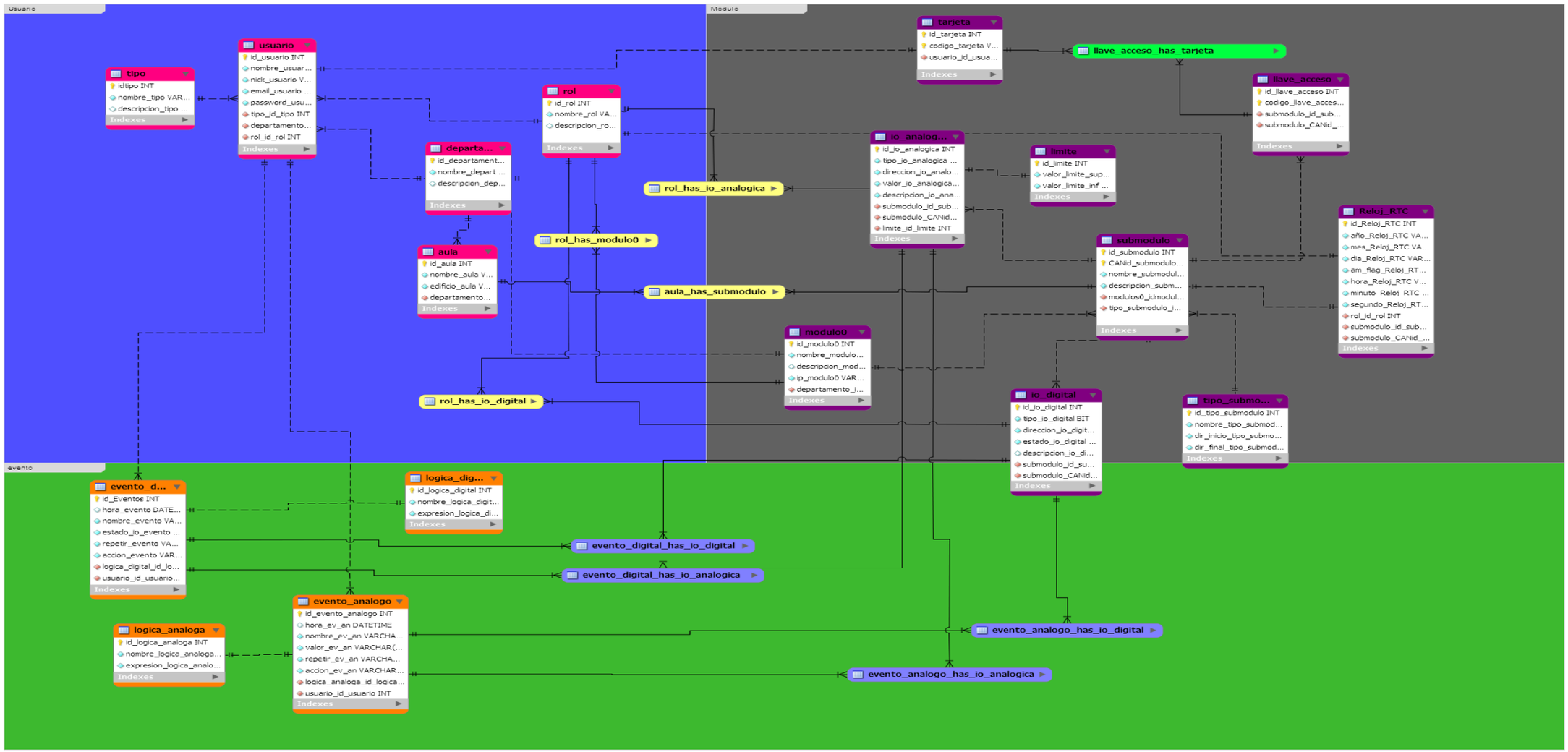


Figura 3.10 Diagrama Entidad Relación.

CONCLUSIONES DEL CAPITULO III

- Utilizar los medios de comunicación proporcionados por la tecnología actual como internet, es una herramienta muy útil para dar soluciones ingenieriles a problemas como el manejo, control y administración de sistemas eléctricos.

- Java es una herramienta muy potente para desarrollar software independiente de la plataforma del sistema operativo.

- El diseño de la solución tiene las características de interfaz amigable, con funcionalidades de monitoreo, administración y control de sistemas de iluminación, aire acondicionado y control de acceso.

- El uso de un software motor de persistencia hace posible la independencia del gestor de base de datos.

- El software motor de persistencia hibernate nos permite migrar a otro gestor de bases de datos sea este código abierto o comercial.

- La API Jamod, implementa todas las funcionalidades del protocolo MODBUS sobre TCP/IP.

- El uso de código abierto permite la implementación de aplicaciones ingenieriles de bajo costo.

REFERENCIAS BIBLIOGRAFICAS

- Gómez Pérez, Gustavo y Villatoro Ríos, Wilman Roberto. "Diseño y construcción de un modulo electrónico programable vía Ethernet desde la aplicación Web, para monitorear, controlar y administrar energía en aplicaciones comerciales" Tesis para optar para el grado de Ingeniero Electricista. Biblioteca de la facultad de Ingeniería y Arquitectura de la Universidad de El Salvador, publicado 2007

- Estandar I2C http://ocw.weber.edu/automotive-technology/ausv-1320-automotive-electronics/17-serial-data/iso-11898

- 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

http://www.microchip.com

CONCLUSIONES GENERALES

- El diseño del prototipo SIPROE consta de un hardware funcional, flexible, y modular. Por el uso de módulos según sea necesario. Permitiendo su expansión atraves de controladores.

- El diseño del software SIPROE tiene las características de interfaz amigable, con funcionalidades de monitoreo, administración y control de sistemas de iluminación, aire acondicionado y control de acceso.

- El protocolo CAN permite optimizar la comunicación entre controladores, dando la facilidad de agregar más dispositivos al bus, sin que este pierda su rendimiento.

- El protocolo industrial MODBUS permite manejar dispositivos remotos; independiente del medio físico que se implemente para hacer peticiones, sea Ethernet, RS232 o CAN.

RECOMENDACIONES

- Agregar funcionalidades al sistema SIPROE de manejo de cargas industriales como: sistemas de medición de energía trifásica, corrección de factor de potencia, manejo de motores o variadores de frecuencia. Utilizando las librerías desarrolladas del bus CAN

- Desarrollar una versión del software de SIPROE para equipos móviles en Java para poder controlar el sistema SIPROE desde un teléfono celular y otros.

ANEXOS

A.1 Manual de Usuario

A.1.1 Propósito de esta guía

El propósito de esta guía es proporcionar a los usuarios del SIPROE una referencia rápida para poder implementar el sistema cuando se requiera, esta incluye las configuraciones en hardware y software

A.1.1.1 Resumen de secciones

Este manual se divide en varias secciones las cuales se resumen en la siguiente:

Tabla A.1 Secciones del manual de usuario

SECCION	PROPOSITO
Configuración del controlador inteligente maestro	Provee una guía para configurar el controlador maestro
Configuración del controlador inteligente de reloj de tiempo real	Provee la forma en que se configura el reloj de tiempo real
Configuración del controlador de Acceso	Provee la forma de configurar el controlador de acceso
Configuración del controlador inteligente de propósito general	Provee la manera en que se configura el controlador de propósito general
Instalación de aplicación SIPROE en Tomcat	Provee una guía para la puesta en marcha de los servicios necesarios para el funcionamiento de SIPROE
Interfaz grafica de la aplicación web	Provee una descripción de la interfaz grafica de la aplicación web de SIPROE

A.1.1.2 Información Adicional

Todo el firmware, el software, hoja de especificaciones de componentes y los programas necesarios para la implementación del sistema inteligente se encuentran en el CD anexo al libro, esto para facilitar su implementación.

A.1.2 Configuración del controlador inteligente maestro

El servidor embebido en el controlador inteligente maestro usa el protocolo TCP/IP para comunicarse en la red ethernet, soporta ARP, UDP, TCP, ICMP, Telnet, TFTP, DHCP, y SNMP. Toda la información acerca 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 SIPROE; 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[33]; toda la información es guardada en la memoria no volátil que posee el módulo Xport.

A. 1.2.1 Buscando un Controlador Inteligente Maestro en la Red

Cuando existe un nuevo controlador inteligente maestro 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 controlador maestro, ahí está 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: 00-20-4A-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 está buscando.

A. 1.2.2 Accesando a Modo configuración

Para configurar el controlador maestro 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

[33] 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.

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.

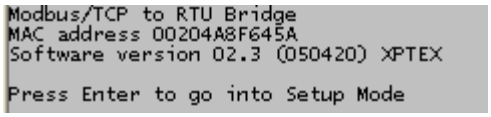


Figura A. 1. 1 Pantalla de inicio en Modo Configuración del controlador maestro.

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

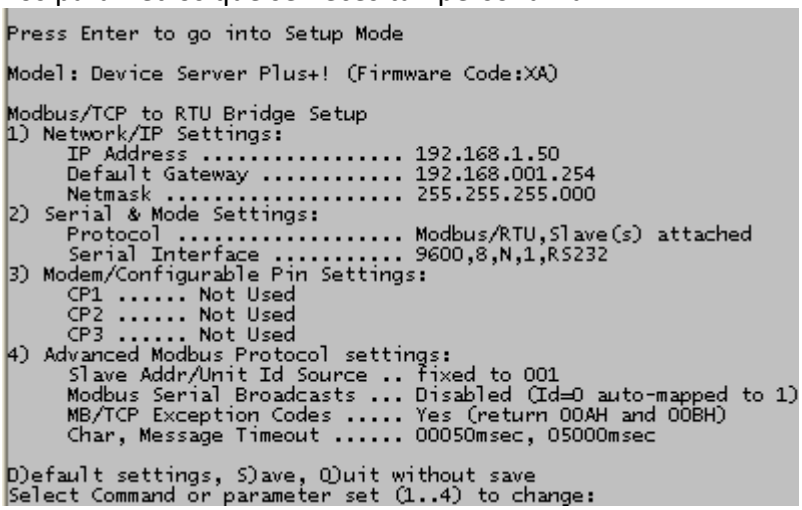


Figura A.1. 2 Pantalla de Opciones en Modo Configuración del controlador maestro.

A. 1.2.3 Configuración IP del servidor Modbus/TCP

Seleccione 1 para configurar los parámetros de red del controlador maestro 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 está 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 siproe 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 Controlador maestro esta ahora configurado y listo para que SIPROE funcione satisfactoriamente

A.2 Configuración CAN

Setup Criteria	
Oscilator Frequency	10.000 MHz
Target CAN Bus Baud Rate	125.000 kbps

Selected Options	
BRP-1 (Baud Rate Prescaler)	4
Tq (Time Quanta)	1.000 µs
Number of Time Quanta	8
% Error of Target Baud Rate	0.0 %

Bit Timing Setup in Tq	
Propagation Delay	1
Phase Segment 1	3
Phase Segment 2	3
Syncronization Jump Width (SJW)	1

Multiple bit sampling is off. Wakeup filter is off.

Bit Timing Diagram

Configuration Register Setup (PIC18/MCP251X) (neoVI blue/green, ValueCAN 2)		
Register	Binary	Hexadecimal
CNF1/BRGCON1	b'00000100'	0x04
CNF2/BRGCON2	b'10010000'	0x90
CNF3/BRGCON3	b'00000010'	0x02

A. 3 Lista de materiales

A. 3.1 lista de materiales controlador maestro

La lista de los componentes que forman parte del controlador Maestro y el costo de los mismos se muestra en el siguiente cuadro resumen.

Tabla A.2 Lista de materiales y precios del controlador maestro

CANTIDAD	TIPO		PRECIO UNITARIO	SUBTOTAL
Resistor				
1	R4	10k	$ 0.15	$ 0.15
2	R9,R10	330R	$ 0.15	$ 0.30
1	R3	120R	$ 0.15	$ 0.15
Capacitor				
1	C5	0.1u	$ 0.22	$ 0.22
2	C1,C2	22p	$ 0.18	$ 0.36
Circuito Integrado				
1	U3	PIC18F458	$ 9.90	$ 9.90
1	U8	MCP2551	$ 1.50	$ 1.50
Diodos				
2	D3,D4	LED	$ 0.35	$ 0.70
Miscelaneos				
1	J6	CONN-D9F	$ 0.55	$ 0.55
1	J9	CONECTOR	$ 1.00	$ 1.00
1	PB1	BUTTON2	$ 0.25	$ 0.25
1	X1	CRYSTA	$ 0.75	$ 0.75
1	J7	CONN-D9M	$ 0.55	$ 0.55
1	XPORT	DIGICONNECT	$ 55.00	$ 55.00
Costo Total Controlador Maestro				**$ 71.38**

A. 3.2 Presupuesto de materiales para el Controlador Inteligente RTC

Tabla A.3 Lista de materiales y precios del controlador reloj calendario

CANTIDAD	NOMBRE	VALOR O ID	PRECIO UNITARIO	SUBTOTAL
Resistores				
1	R3	120R	$ 0.15	$ 0.15
1	R4	10k	$ 0.15	$ 0.15
4	R5-R8	4k7	$ 0.15	$ 0.60
3	R9-R11	330R	$ 0.15	$ 0.45
Capacitor				
2	C1,C2	22p	$ 0.18	$ 0.36
1	C3	0.1u	$ 0.18	$ 0.18
2	C4,C5	10u	$ 0.18	$ 0.36
Circuito Integrado				
1	U1	7805	$ 0.60	$ 0.60
1	U2	MCP2551	$ 1.50	$ 1.50
1	U3	PIC18F258	$ 8.00	$ 8.00
1	U4	DS1337	$ 3.00	$ 3.00
Diodo				
5	D1-D5	LED	$ 0.35	$ 1.75
Miscelaneos				
1	J1	VCC EXTERNA	$ 0.25	$ 0.25
1	J2	CONN-D9M	$ 0.55	$ 0.55
1	J3	SIL-100-06	$ 0.12	$ 0.12
1	J4	Conector Controlador	$ 1.00	$ 1.00
1	JP1	JUMPER	$ 0.12	$ 0.12
1	JP2	JUMPER2	$ 0.12	$ 0.12
1	PB1	BUTTON2	$ 0.25	$ 0.25
2	X1,X2	CRYSTAL	$ 0.75	$ 1.50
Precio Por Controlador Inteligente RTC				**$ 21.01**

A. 3.3 presupuesto de materiales para controlador de propósito general

Tabla A.4 Presupuesto de materiales para acondicionamiento de señales del controlador de propósito general.

CANTIDAD	NOMBRE	VALOR o ID	PRECIO UNITARIO	SUBTOTAL
Resistor				
1	R1	100R	$ 0.15	$ 0.15
8	R2-R9	390k	$ 0.15	$ 1.20
Capacitor				
8	C1,C4,C7,C10,C13, C16,C19,C22	2u2	$ 0.18	$ 1.44
16	C2,C3,C5,C6,C8,C9,C11,C12,C14,C 15, C17,C18,C20,C21,C23,C24	100n	$ 0.18	$ 2.88
Circuito Integrado				
2	U1,U2	ULN2803	$ 2.50	$ 5.00
2	U3,U4	LM324	$ 2.00	$ 4.00
8	U5-U12	HCPL3700	$ 3.00	$ 24.00
2	U13,U14	CNY74-4	$ 3.50	$ 7.00
Diodos				
2	D1,D2	1N5256B	$ 0.40	$ 0.80
Miscelaneos				
1	DS1	DIPSW_8B	$ 0.80	$ 0.80
5	J1,J10,J11,J13,J14	CONN-DIL20	$ 1.00	$ 5.00
8	J2-J9	TBLOCK-M3	$ 0.50	$ 4.00
1	J12	CONN-SIL8	$ 0.35	$ 0.35
33		TBLOCK-I3	$ 0.50	$ 16.50
8	RLY1-RLY8	DSP1-L2-DC5V	$ 4.50	$ 36.00
6	RN1-RN6	RX8	$ 1.00	$ 6.00
1	RP1	RESPACK-8	$ 0.70	$ 0.70
Costo Acondicionamiento de señales				**$ 115.82**

Tabla A.5 Presupuesto de materiales Controlador Propósito General

CANTIDAD	NOMBRE	VALOR O ID	PRECIO UNITARIO	SUBTOTAL
Resistor				
2	R1,R2	3k9	$ 0.15	$ 0.30
1	R3	120R	$ 0.15	$ 0.15
1	R4	10k	$ 0.15	$ 0.15
4	R5-R8	4k7	$ 0.15	$ 0.60
2	R9,R10	330R	$ 0.15	$ 0.30
2	R11,R12	1K	$ 0.15	$ 0.30
Capacitor				
2	C1,C2	22p	$ 0.18	$ 0.36
2	C3,C4	33p	$ 0.18	$ 0.36
1	C5	0.1u	$ 0.22	$ 0.22
2	C6,C7	4u7	$ 0.22	$ 0.44
1	C8	10n	$ 0.18	$ 0.18
Integrados				
3	U1,U2,U11	MCP23008	$ 2.50	$ 7.50
1	U3	PIC18F258	$ 8.00	$ 8.00
2	U4,U5	MCP23016	$ 2.00	$ 4.00
2	U6,U7	MAX127	$ 12.00	$ 24.00
1	U8	MCP2551	$ 1.50	$ 1.50
1	U9	MAX521	$ 12.00	$ 12.00
Diodos				
3	D1,D3,D4	LED	$ 0.35	$ 1.05
Miscelanios				
4	DSW1-DSW4	DIPSW_8	$ 0.80	$ 3.20
5	J1-J5	CONN-DIL20	$ 1.00	$ 5.00
1	J6	CONN-D9F	$ 0.55	$ 0.55
1	J7	CONN-D9M	$ 0.55	$ 0.55
1	J8	CONN-H20	$ 1.00	$ 1.00
1	J9	PUERTO	$ 0.50	$ 0.50
1	J10	TBLOCK-M3	$ 0.50	$ 0.50
1	JP1	JUMPER	$ 0.12	$ 0.12
1	PB1	BUTTON2	$ 0.25	$ 0.25
4	RN1-RN4	RX8	$ 1.00	$ 4.00
2	X1,X2	CRYSTAL	$ 0.75	$ 1.50
Costo por Controlador de Propósito General				$ 78.58