Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Electrónica

Instituto Costarricense de Electricidad

ICE

“Implementación de un traductor de protocolo Modbus para una unidad terminal remota de un sistema SCADA”

Informe de Proyecto de Graduación para optar por el Grado de Bachiller en Ingeniería Electrónica

Fernando Lizana Moreno

Cartago, 2001

Resumen

Los sistemas de control y adquisición de datos son conocidos por sus siglas en inglés como sistemas SCADA. Se utilizan para supervisar y controlar la operación de diversos procesos físicos. Básicamente, el sistema SCADA se compone por una serie de dispositivos llamados unidades terminales remotas, las cuales sirven como interfaz con los procesos físicos controlados, y una computadora central, que se encarga de enviar mensajes con las diferentes instrucciones de control y supervisión que las unidades terminales remotas deben realizar.

Para establecer la comunicación entre la computadora central y las unidades terminales remotas de un sistema SCADA, se pueden utilizar diferentes protocolos. Un protocolo estandarizado y utilizado por varios fabricantes es el llamado protocolo Modbus.

El Instituto Costarricense de Electricidad posee un sistema SCADA para la supervisión de las redes de distribución con que brinda el servicio de energía eléctrica a sus clientes.

Dicho sistema se está actualizando, mediante el rediseño de todas sus partes. Dentro del diseño de la unidad terminal remota, se quiere dotarla de un protocolo de comunicación estandarizado, por lo que se ha recurrido al Modbus.

El presente informe documenta la realización de un dispositivo traductor, que permita la comunicación entre la computadora central y la unidad terminal remota de un sistema SCADA, mediante la utilización del protocolo Modbus.

Palabras claves: protocolo Modbus, unidades terminales remotas, sistemas SCADA, distribución eléctrica.

Abstract

The systems of control and data acquisition are known by their initials as SCADA systems. They are used to supervise and control the operation of diverse physical processes. Basically, the SCADA system is composed by a series of devices called remote terminal units, which serve as interfase with the controlled physical processes, and a central computer that takes charge of sending messages with the different control and supervision instructions that the remote terminal units should carry out.

To establish the communication between the central computer and the remote terminal units of a SCADA system, different protocols can be used. A standardized protocol used by several makers, is the call Modbus protocol.

The Instituto Costarricense de Electricidad possesses a SCADA system for the supervision of the distribution nets in which they offers the electric power service to they clients.

This system is modernizing, through the redesign of all of its parts. Inside the design of the remote terminal unit. For that, it is wanted to endow it of a standardized communication protocol, for what has been appealed the Modbus.

This report documents the realization of a translating device that allows the communication between the central computer and the remote terminal unit of a SCADA system, using the Modbus protocol.

Key words: Modbus protocol, remote terminal units, SCADA systems, electric distribution.

ÍNDICE GENERAL

Página

ÍNDICE DE FIGURAS

Página

ÍNDICE DE TABLAS

Página

CAPÍTULO 1

INTRODUCCIÓN

1.1 Descripción de la empresa

A continuación se brinda una descripción general de la empresa donde se está desarrollando el proyecto.

1.1.1 Descripción general

El Instituto Costarricense de Electricidad (ICE) es una empresa costarricense, la cual fue creada como institución autónoma en el año 1949 y fue concebida desde su origen como el ente rector y principal ejecutor del desarrollo y administración de la industria eléctrica nacional.

Con la creación del ICE, el país avanzó notablemente en el desarrollo del sector eléctrico, en la segunda mitad del siglo XX. En los primeros 10 años se triplicó la capacidad instalada nacional y hoy en día la misma es superior al millón de kilovatios, o sea 30 veces mayor que la que se tenía en 1950. Asimismo, el servicio es confiable, de alta calidad, de cobertura nacional, sustentado en el uso racional de los recursos energéticos nacionales y soportado en políticas y planes de expansión a largo plazo.

El ICE actualmente brinda servicios en las áreas de energía y telecomunicaciones. Para brindar un mejor servicio al cliente, se encuentra dividido en dos grandes sectores: ICE-Energía e ICE-Telecomunicaciones.

ICE-Energía se encuentra subdividido en 6 Unidades Estratégicas de Negocios (UEN), las cuales son:

Producción de electricidad

Transporte de electricidad

Servicio al cliente

Proyectos y servicios asociados

Centro nacional de control de energía

Centro nacional de planificación de energía

1.1.2 Descripción del departamento donde se está realizando el proyecto

El proyecto se realizó para el Centro de Servicio Investigación y Desarrollo, el cual pertenece a la UEN Proyectos y Servicios Asociados, del ICE Energía.

La Unidad Estratégica de Negocios Proyectos y Servicios Asociados es un área estratégica del ICE Energía, en la cual se llevan a cabo proyectos que le permiten al ICE continuar siendo líder en el desarrollo del Sistema Eléctrico Nacional y Regional, proyectando su capacidad dentro y fuera de las fronteras de Costa Rica, con un compromiso con el medio ambiente.

Los proyectos incluyen el diseño y construcción de hardware electrónico y también el desarrollo de aplicaciones de software en diferentes plataformas de desarrollo (Unix, Dos, Windows).

Este Centro de Servicio cuenta con los siguientes laboratorios: electrónica de corrosión, circuitos impresos y sistemas de potencia.

1.2 Definición del problema y su importancia

La Unidad Estratégica de Negocios Proyectos y Servicios Asociados, del ICE Energía, requiere montar un sistema para supervisar y controlar las líneas de distribución eléctrica, en las regiones que se encuentran electrificadas, con el objetivo de poder controlar la calidad del servicio brindado a los usuarios.

Para esto, el Centro de Servicio Investigación y Desarrollo está implementando un Sistema de Control y Adquisición de Datos, conocido por su nombre en inglés como System of Control and Data Acquisition, y por sus siglas como sistema SCADA, para la supervisión y control de la red de distribución de energía eléctrica, a la vez que actualiza la parte existente. Cabe mencionar que en las regiones de Cañas, Naranjo y Barranca, ya opera un sistema de este tipo).

Un sistema SCADA es básicamente un sistema de monitoreo y control de Unidades Terminales Remotas (UTR), por medio de una computadora central[1]. Las Unidades Terminales Remotas consisten en interfaces entre los procesos físicos que se requieren controlar y la computadora central.

Existen diversos protocolos para establecer las comunicaciones entre las UTR y la computadora central. Uno de estos es el protocolo Modbus[2].

---

[1] En el apéndice 1 se brinda una descripción más detallada sobre los sistemas SCADA

[2] Se ha adjuntado una explicación más detallada acerca del funcionamiento de dicho protocolo en el apéndice 2.

El proyecto SCADA que se está desarrollando en el Instituto Costarricense de Electricidad pretende actualizar la tecnología de un sistema que ya cumplió con su vida útil. Actualmente se ha diseñado y fabricado en serie un módem para las regiones de Cañas y Naranjo y está en la etapa de diseño la unidad terminal remota que se utilizará, y se espera esté finalizada para mediados de este año. Así mismo, se ha elaborado el software necesario para realizar las operaciones de control y monitoreo de los diferentes equipos de distribución.

Dentro del diseño de la unidad terminal remota, se pretende dotarla de un protocolo estandarizado, que sea utilizado por distintos fabricantes, para lo cual se ha recurrido al Modbus. Para realizar lo anterior, es necesario diseñar e implementar un dispositivo que permita a las Unidades Terminales Remotas la comunicación con el Computador Central, el cual se ubica en el Centro Local de Operación de Redes (CLOR), mediante la utilización del protocolo Modbus.

El problema del cual se ocupa el presente trabajo es el diseño e implementación de un dispositivo traductor de protocolo Modbus, que sirva de interfaz entre la Computadora Central y las Unidades Terminales Remotas del sistema SCADA.

La Computadora Central enviará señales a las UTR utilizando el protocolo Modbus. El traductor debe ser capaz de comunicarle al UTR las instrucciones recibidas. Así mismo, el traductor debe ser capaz de enviar las señales generadas por la UTR a través del módem, utilizando el protocolo mencionado.

Además, en vista de que las Unidades Terminales Remotas no se han terminado de diseñar, se requiere de la implementación de un programa de prueba, desarrollado en un computador, mediante el cual se compruebe el óptimo funcionamiento del Dispositivo Traductor a desarrollar.

La ventaja más grande que tiene el hecho de que los equipos utilizados tengan la capacidad de comunicarse mediante el protocolo Modbus es la facilidad que tendrían para interconectarse con equipos desarrollados por otros fabricantes, logrando así que el sistema SCADA implementado sea compatible con otros equipos que se pueden agregar en un futuro para el mejoramiento del mismo.

1.3 Objetivos

A continuación se muestran los objetivos que se plantearon para el desarrollo del presente proyecto

1.3.1 Objetivo general

Implementar un dispositivo traductor de protocolo Modbus para una unidad terminal remota UTR de un sistema SCADA.

1.3.2 Objetivos específicos

a. Investigar el funcionamiento y modo de aplicación del protocolo Modbus.

b. Investigar el funcionamiento y programación del microcontrolador Basic Stamp.

c. Diseñar el circuito del traductor de protocolo Modbus para la unidad terminal remota.

d. Programar en lenguaje Delphi una aplicación de prueba del dispositivo traductor de protocolo Modbus.

e. Montar el circuito del traductor de protocolo Modbus para la unidad terminal remota.

f. Probar el circuito del traductor de protocolo Modbus para la unidad terminal remota.

g. Diseñar el software para el traductor de protocolo Modbus para la unidad terminal remota.

h. Realizar pruebas finales a todo el proyecto mediante la utilización del software de aplicación creado.

i. Escribir un informe técnico del proyecto para la empresa y el informe de proyecto de graduación para el Instituto Tecnológico de Costa Rica.

CAPÍTULO 2

ANTECEDENTES

2.1 Estudio del problema a resolver

A continuación se presenta un análisis de los aspectos técnicos del problema que se solucionó.

2.1.1 Sistema SCADA

El dispositivo traductor que se desarrolló es parte de una unidad terminal remota de un sistema SCADA. Funciona como una interfaz entre el módem y el CPU de la UTR.

El módem se encarga de la comunicación física entre la computadora central y la UTR. El CPU se encarga de leer los estados de las diferentes variables del sistema y escribirlos en la memoria RAM de la UTR, así como de forzar las variables a los valores requeridos por la computadora central. La figura 2.1 presenta un esquema general de la composición de la UTR con la que se está trabajando.

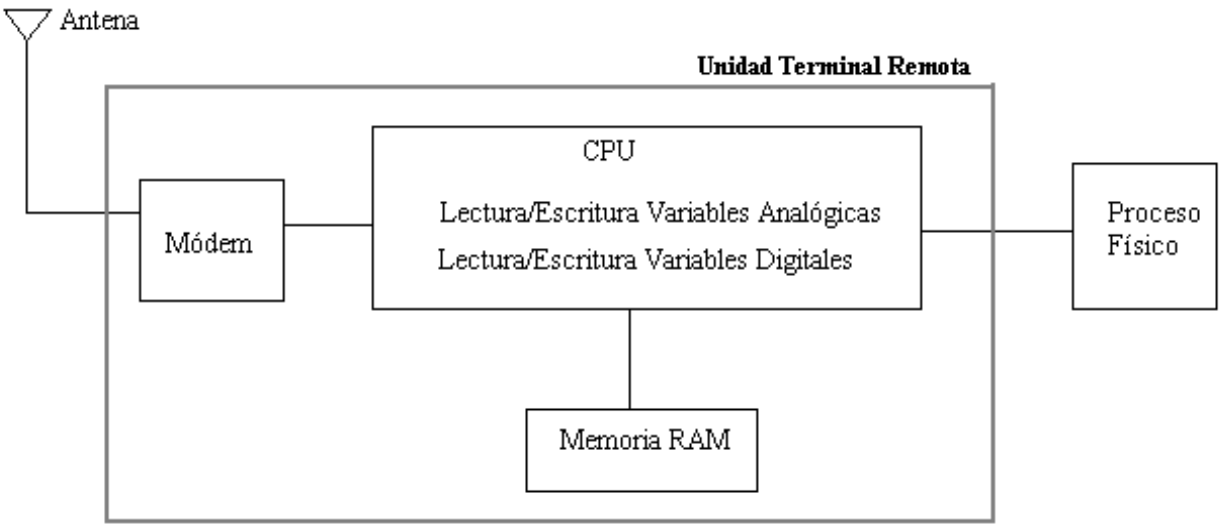

Figura 2.1 Esquema de la composición de una unidad terminal remota.

El dispositivo traductor que se desarrolló recibe los datos del módem a través de un puerto serie RS-232. Dichos datos están escritos en protocolo Modbus. Luego, dicho dispositivo debe ser capaz de realizar las acciones solicitadas por la computadora central (principalmente las acciones de lectura y escritura). Para realizar las acciones de lectura, el dispositivo traductor debe tener acceso a la memoria RAM de la UTR. Para realizar las acciones de escritura, el traductor debe indicarle al CPU la dirección y el valor de la escritura.

En vista de que en el sistema SCADA que se está implementando operarán varias UTR, cada una debe tener una dirección distinta. Entonces, el dispositivo traductor debe reconocer la dirección de la UTR en la que se encuentra, para lo que se realiza la lectura del espacio de selección de dirección de la UTR.

La figura 2.2 indica cómo debe operar dentro de la unidad terminal remota el dispositivo traductor que se diseñó. Las funciones del dispositivo traductor no son únicamente de interpretación de mensajes, sino que debe también realizar lecturas y construir mensajes de respuesta adecuados.

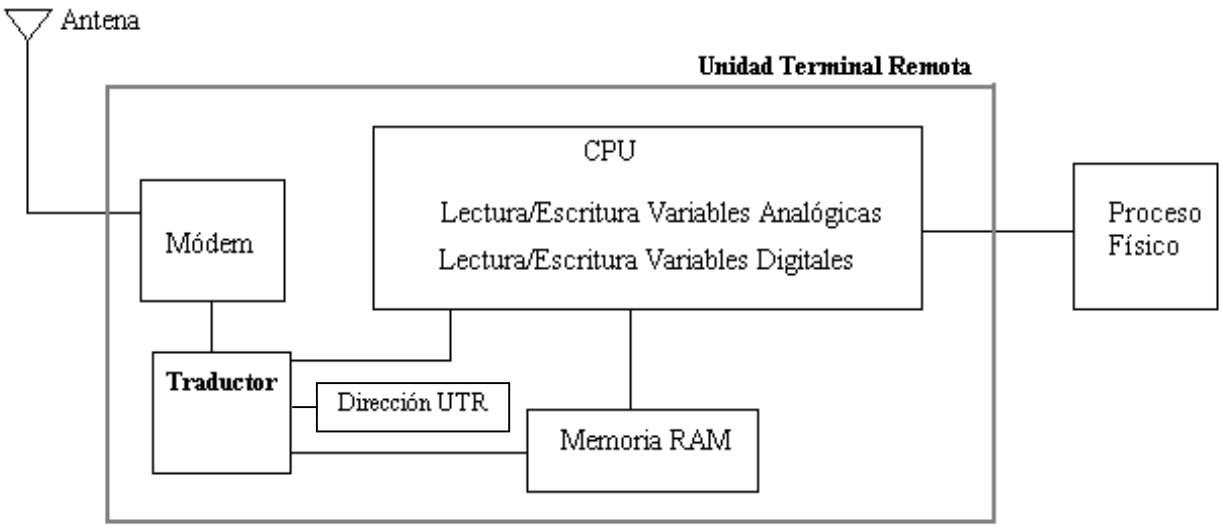

Figura 2.2 Esquema de operación del dispositivo traductor.

La figura 2.2 sirve de guía para la comprensión del hardware del dispositivo traductor, al mostrar las conexiones que debe tener dentro de la unidad terminal remota.

2.1.2 Protocolo Modbus

El protocolo Modbus está constituido básicamente en un sistema de mensajes mediante la estructura Maestro-Esclavo. Los esclavos se comunican con el maestro únicamente si éste se los solicita. Esto presupone que el dispositivo traductor que se desarrolló no debe preocuparse por construir mensajes, a menos que haya recibido uno previamente.

El conjunto de instrucciones implementadas en el protocolo Modbus es reducido, lo que lo convierte en un protocolo simple de utilizar. Las tablas A2.2 y A2.3 del apéndice 2 muestran dicho conjunto, con una breve descripción de las instrucciones.

Como se aprecia en dicha tabla, las instrucciones 1, 2, 3, 4, 5, 6, 15 y 16 se utilizan para realizar lecturas y escrituras de señales digitales y analógicas. El proyecto que se realizó tiene como objetivo la implementación de dichas instrucciones, dejando abierta la posibilidad de implementar las otras en un futuro.

Los mensajes en protocolo Modbus se componen por diferentes campos de datos. Primero se coloca la dirección del dispositivo con el que se quiere establecer comunicación. Luego se coloca la instrucción que se pretende que dicho dispositivo realice. Cada mensaje tiene sus características específicas3, dependiendo de la función que se requiera realizar. Al final de cada mensaje, se colocan siempre dos caracteres de verificación de error, utilizando el chequeo de redundancia cíclica.

El dispositivo traductor desarrollado debe entonces estar en constante monitoreo del puerto serie. Cuando reciba una señal debe verificar que se recibió el

mensaje sin errores. Hecha esta verificación, debe procesar el mensaje, para luego construir y enviar un mensaje de respuesta.

El dispositivo debe tener la capacidad de verificar la realización óptima de los requerimientos de la computadora central. En caso de que se dé alguna falla, el dispositivo traductor debe construir un mensaje de error (definido por el protocolo Modbus) y enviarlo al módem, para que éste lo comunique a la computadora central.

2.1.3 Programas de aplicación en lenguaje Delphi

Para la prueba del dispositivo traductor se desarrollaron dos programas de aplicación en lenguaje Dephi.

El primer programa se implementó para simular la Computadora Central del sistema SCADA. Dicho programa envía y recibe mensajes de protocolo Modbus a través del puerto serial de una computadora, hacia el dispositivo traductor.

---

[3] En el apéndice 2 se facilita información sobre la estructura de los diferentes mensaje Modbus

El segundo programa tuvo la función de simular la CPU de la unidad terminal remota. Al igual que el programa mencionado anteriormente, la comunicación con el dispositivo traductor se lleva a cabo a través del puerto serial de una computadora.

2.2 Requerimientos de la empresa

Al implementarse un sistema SCADA para supervisión y control de la red de distribución de energía eléctrica, en el Centro de Servicio Investigación y Desarrollo, del ICE Energía, se están desarrollando las UTR que irán colocadas en los diferentes puntos de supervisión. Sin embargo se hace necesario la implementación de un traductor que permita a las UTR comunicarse utilizando un protocolo actualizado. Se ha elegido inicialmente el protocolo Modbus para la implementación de dicho traductor, quedando abierta la posibilidad de ampliar la capacidad de comunicación de las UTR a otros protocolos utilizados actualmente.

La necesidad de contar con un sistema SCADA moderno hizo que la parte de distribución eléctrica del ICE tomara la decisión de actualizar sus equipos, para ello contactó al Centro de Servicio de Investigación y Desarrollo para la realización de estas tareas. Con este fin, se implementaron aplicaciones en Windows (versiones anteriores en DOS) y se reemplazó algunos de los equipos de comunicaciones implementando un módem con características propias de los sistemas. Adicionalmente se implementa actualmente una UTR con la finalidad de darle mayor cobertura al sistema y también con el objeto de reemplazar equipo obsoleto.

El dispositivo traductor que se desarrolló en el presente proyecto cumplirá la función de dotar al sistema SCADA del ICE de un protocolo de comunicación moderno y compatible con una amplia gama de equipos de diversos fabricantes, lo que permitirá una adecuada compatibilidad y una eventual ampliación del sistema.

2.3 Solución propuesta

La empresa ya contaba con una solución proyectada para enfrentar el problema planteado, la cual consistió en la utilización de un microcontrolador Basic Stamp 2sx, del fabricante Parallax Inc. para implementar un dispositivo traductor de protocolo Modbus para la unidad terminal remota que se está implementando.

A continuación se brinda una descripción de la propuesta de solución que fue dada para el desarrollo del proyecto.

2.3.1 Microcontrolador Basic Stamp 2sx

La decisión de utilizar este microcontrolador fue tomada, principalmente, por la facilidad de adquisición del mismo para la empresa. El costo de dicho microcontrolador es de 45 000 colones, sin embargo, aunque existe en el mercado microcontroladores más baratos, el Basic Stamp 2sx tiene una serie de características que lo presentaron como el ideal para el desarrollo del proyecto, entre las cuales se pueden destacar las siguientes:

a. Posee una memoria EEPROM muy amplia (8 bancos de 2 Kilobytes cada uno), lo que le permite almacenar un programa extenso (16 Kilobytes en total), sin necesidad de incluir memoria ROM adicional.

b. Posee 32 bytes de memoria RAM.

c. Tiene además 64 bytes de memoria RAM alternativa, lo cual aumenta la disponibilidad de memoria volátil para el desarrollo de programas.

d. Se programa en lenguaje PBASIC, el cual es simple y eficiente.

e. Posee incorporada una interfaz RS-232, la cual es necesaria para el desarrollo del proyecto.

f. Requiere de muy poca circuitería externa para un óptimo funcionamiento.

g. Al tener integrado la mayoría de componentes necesarios (como memoria ROM, RAM, interfaz RS-232), sale más barato de utilizar que otros microcontroladores menos costosos, pero sin las anteriores facilidades.

En la figura 2.3 se aprecia un diagrama del microcontrolador que se utilizó. Como puede observarse en dicha figura y en la tabla 2.1, este dispositivo posee las señales necesarias para manejar una interfaz RS-232, además de 16 puertos utilizables como señales de entrada/salida.

El Hardware del proyecto consistió principalmente en la interconexión del microcontrolador Basic Stamp 2sx a la unidad terminal remota y a un módem, mediante el cual se comunica con el Computador Central del sistema SCADA.

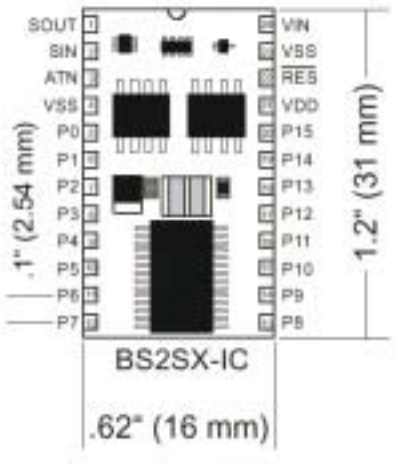

Figura 2.3 Diagrama del Microcontrolador Basic Stamp 2sx de Parallax Inc.

Tabla 2.1 Pines del microcontrolador Basic Stamp 2sx.

Número de Pin	Nombre del Pin	Descripción
1	SOUT	Salida serial (Interfaz RS-232).
2	SIN	Entrada serial (Interfaz RS-232)
3	ATN	Atención (Interfaz RS-232). Se conecta a la señal DTR de un puerto serial para comunicaciones.
4	VSS	Tierra
5-20	P0-P15	Puertos de Entrada/Salida
21	VDD	Entrada de energía (5 Voltios).
22	RES	Señal de Reinicio
23	VSS	Tierra
24	VIN	Entrada de energía no regulada. Acepta tensiones de 5,5 a 12 voltios CD, e internamente es regulada a 5 voltios

2.3.2 Unidad terminal remota

El Dispositivo Traductor que se diseñó se conecta con el CPU de una unidad terminal remota que controla una serie de procesos físicos. En la figura 2.4 puede observarse el diagrama de la CPU de la UTR que se utilizó.

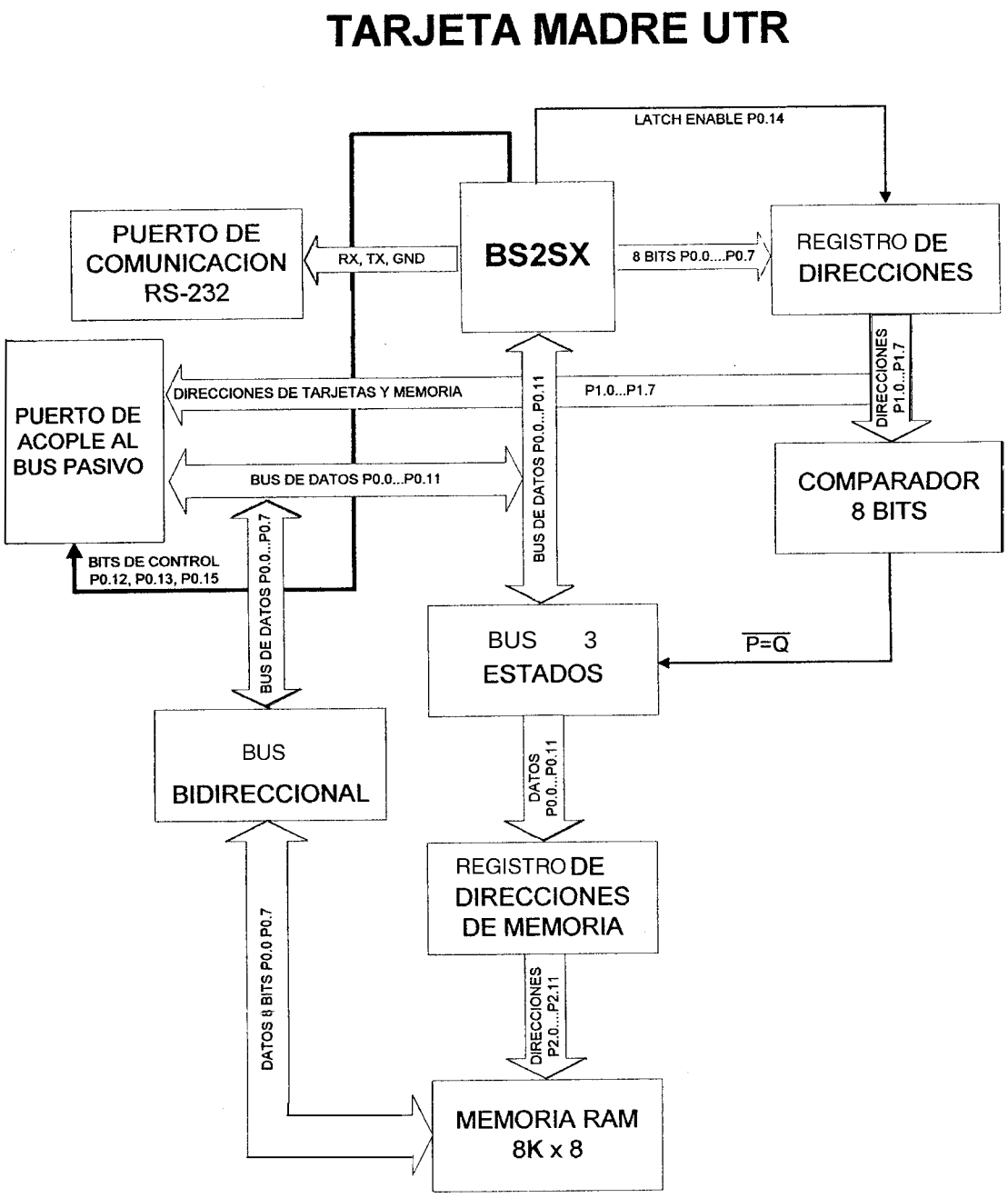

Figura 2.4 Diagrama de la CPU de la Unidad Terminal Remota para la cual se desarrolló el Dispositivo Traductor.

Para comunicar el Dispositivo Traductor con la CPU de la unidad terminal remota se utilizaron tres puertos, dos de los cuales se emplean a modo de puerto serial entre los dispositivos mencionados. El tercer puerto se utiliza como una señal de "requerimiento de comunicación", y es mediante ésta señal que se da inicio a las comunicaciones entre el Dispositivo Traductor y la CPU de la unidad terminal remota.

El microcontrolador Basic Stamp 2sx permite utilizar cualquiera de sus 16 puertos para comunicaciones seriales, mediante las instrucciones SERIN y SEROUT (Ver apéndice 3), lo que resulta una importante simplificación del trabajo a realizar, ya que para comunicar el dispositivo traductor con el CPU de la UTR se requirió únicamente de una instrucción de microcódigo.

2.3.3 Programas de pruebas desarrollado en lenguaje Delphi

Se desarrollaron dos programas en lenguaje Delphi, para probar el desempeño del Dispositivo Traductor mediante la conexión a puertos seriales de dos computadoras.

Para esta parte del proyecto, se utilizó el software Delphi 5. La escogencia de este lenguaje de programación fue justificada, principalmente, por el hecho de que la empresa posee la licencia de utilización de dicho programa. Además, es una herramienta poderosa para el desarrollo de aplicaciones en plataforma Windows.

2.3.4 Descripción de la solución que se implementó

Básicamente, la solución consistió en diseñar un dispositivo, el cual se debe colocar entre la unidad terminal remota y el módem encargado de comunicarse con la computadora central o algún otro dispositivo que utilice el protocolo Modbus, como se indica en la figura 2.5.

Se implementó un sistema mínimo utilizando el microcontrolador Basic Stamp 2sx, el cual es capaz de permitir a la unidad terminal remota comunicarse mediante el protocolo Modbus. Para esto se diseñó la circuitería necesaria para acoplar el traductor con el módem y con la UTR.

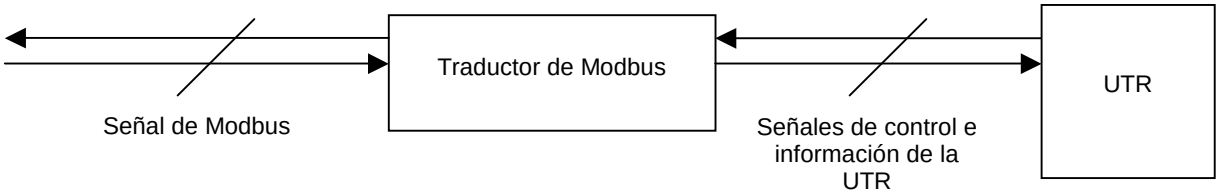

Figura 2.5 Diagrama de la solución propuesta para el problema formulado

Seguidamente se diseñó el programa que contiene el microcontrolador Basic Stamp 2sx.

La implementación de la CPU de las unidades terminales remotas no se encontraba completamente terminada mientras se desarrollaba el presente proyecto, por lo que se definió con anterioridad un lenguaje, el cual va a ser utilizado para comunicar al traductor con la CPU. Las pruebas finales del proyecto no se realizaron conectando directamente el traductor a una UTR, por lo que fue necesario el desarrollo del software de aplicación para probar el óptimo funcionamiento del traductor. Se hace necesario especificar lo anterior para aclarar que el desarrollo del presente proyecto no estuvo sujeto al avance del desarrollo de las unidades terminales remotas.

CAPÍTULO 3

PROCEDIMIENTO METODOLÓGICO

La metodología que se siguió para el desarrollo del proyecto fue la siguiente.

Primero se procedió a investigar sobre el funcionamiento del protocolo Modbus, y sobre el funcionamiento y la programación del microcontrolador Basic Stamp 2sx.

Para esta parte del proyecto, se utilizó información brindada por la empresa, tales como libros y manuales, así como información obtenida de Internet. El objetivo fue determinar adecuadamente el tipo de información que recibirá el dispositivo traductor y la estructura de la misma. Además se requirió aprender a utilizar de forma óptima el microcontrolador Basic Stamp 2sx, para poder llevar a cabo el diseño del circuito y del programa que contenía.

Una vez terminada la etapa de investigación, se procedió al diseño del circuito del dispositivo traductor. Esto correspondió a la parte de diseño del Hardware del proyecto. Para esto se requirió de los manuales de los componentes a utilizar e información varia proporcionada por la empresa. Además, se requirió de una computadora para la documentación del diseño.

Una vez terminado el diseño del circuito, se procedió a pedir a los proveedores el envío de los componentes necesarios.

Mientras se esperaba la llegada de los componentes requeridos, se procedió a diseñar el software que se colocó en la memoria EEPROM del microcontrolador Basic Stamp 2sx. Para llevar a cabo este objetivo se requirió de una computadora.

Antes que llegaran los componentes, y simultáneamente con la etapa anterior, se inició el desarrollo de los programas de aplicación en lenguaje Delphi. Debido a que esta etapa no era crítica en el cronograma del proyecto, se fue realizando paralelamente al desarrollo de las otras actividades.

Una vez que los componentes llegaron, se procedió a montar el circuito del traductor.

Luego, se procedió a realizar pruebas al hardware y al software del dispositivo traductor, con el fin de corregir los errores que se cometieron en las etapas anteriores.

Una vez que todas las etapas anteriores concluyeron, se procedió a realizar pruebas finales a todo el proyecto: el hardware y el software del dispositivo traductor y los programas de aplicación.

CAPÍTULO 4

DESCRIPCIÓN DEL HARDWARE UTILIZADO

El hardware diseñado consistió básicamente en un microprocesador Basic Stamp 2sx, y una memoria SRAM de 64 Kbits, con la circuitería necesaria para su lectura. Debió incluirse además una serie de dispositivos que permitieran escribir manualmente en la memoria RAM de la unidad terminal remota, con el fin de insertar los datos deseados en la misma.

La lista de componentes requeridos para la implementación del hardware se adjunta en el presente trabajo en el apéndice 4.

La figura 4.1 presenta el diagrama del circuito diseñado. Se puede observar en esta figura que se utilizó una serie de interruptores para poder realizar la escritura manual de la memoria SRAM.

El hardware que se presenta en la figura 4.1 debe comunicarse por medio de 2 puertos seriales con las otras partes de la unidad terminal remota a la que pertenece. Para esto, se utilizan los pines SERIN, SEROUT, P14 y P15 del microcontrolador.

Mediante los pines SERIN y SEROUT el microcontrolador se comunicará con el módem de la Unidad Terminal Remota (UTR). Mediante los puertos P14 y P15 se comunica con el CPU de la UTR. La velocidad utilizada en ambos casos es de 9600 baudios.

El puerto 13 del microcontrolador, correspondiente al pin P13, se utiliza para enviar al CPU la solicitud de comunicación.

Se incluyen 2 conjuntos de 8 interruptores cada uno, los cuales se utilizan para definir la dirección que tendrá el dispositivo (la unidad terminal remota) dentro del sistema SCADA, y para elegir el modo de operación que se desea. En el proyecto que se desarrolló, únicamente hay un modo de operación disponible, sin embargo, queda abierta la posibilidad de implementar otros.

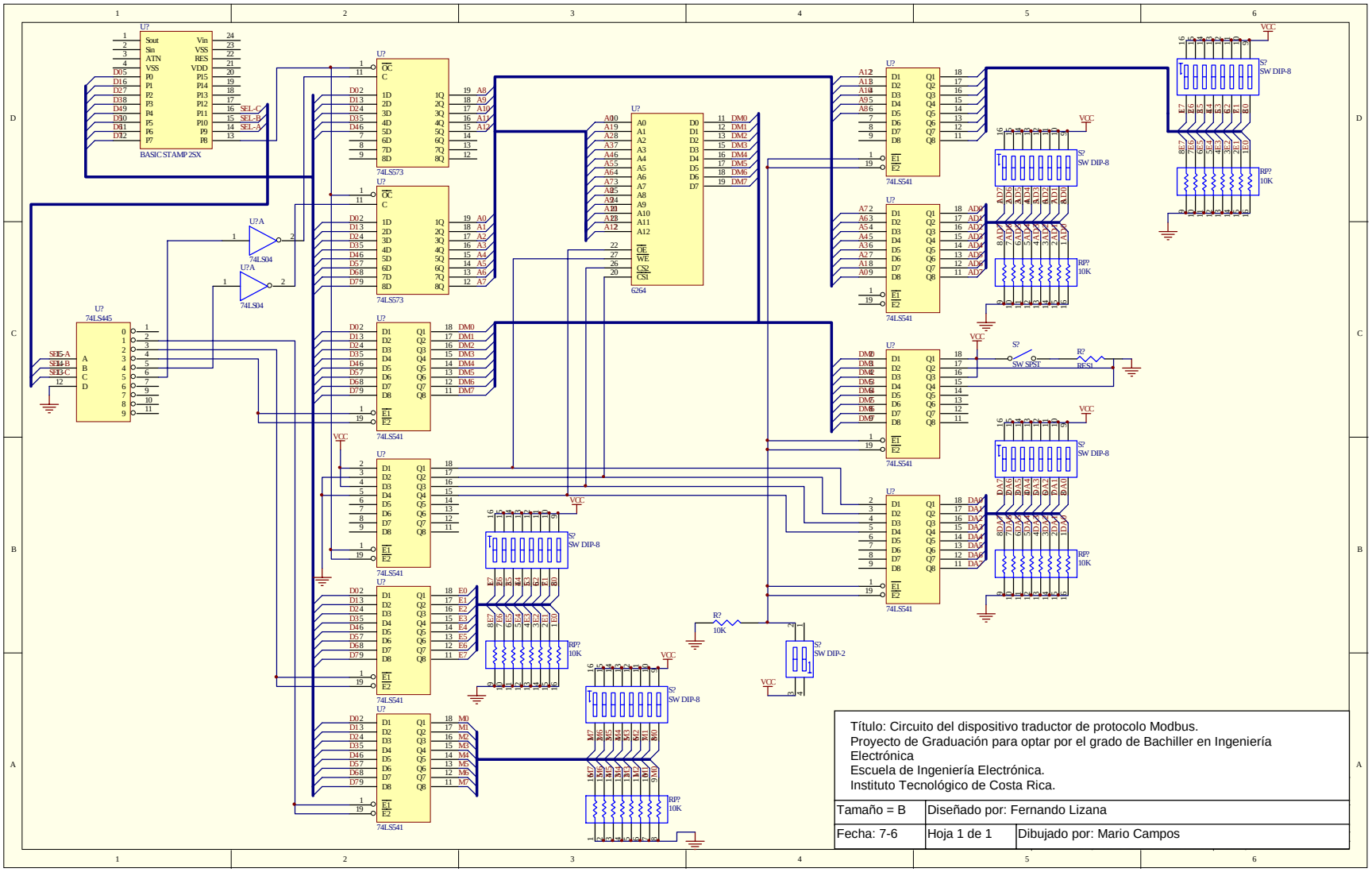

Figura 4.1 Diagrama eléctrico del circuito del traductor de protocolo Modbus.

En la figura 4.2 se puede observar la apariencia del circuito implementado. Se observa a la derecha de dicha figura el conjunto de interruptores que se colocaron para realizar la escritura manual de la memoria RAM. También se puede apreciar la presencia de un botón pulsador, mediante el cual se envía la señal de escritura WR a la memoria.

De acuerdo con el objetivo general del proyecto, se presenta en la figura 4.2 una foto del montaje del circuito.

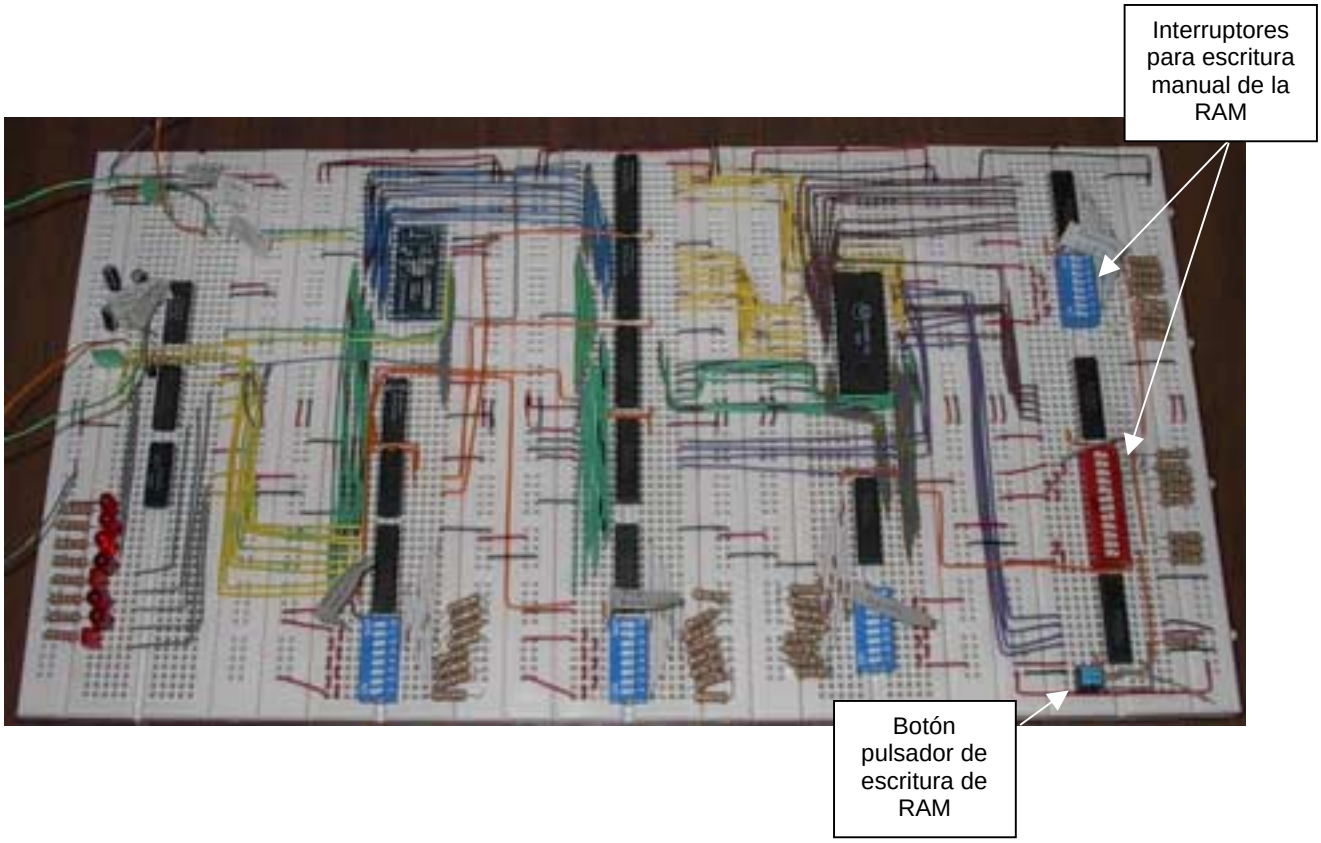

Figura 4.2 Fotografía del circuito del traductor de protocolo Modbus que se implementó, realizado en una tabla de prototipo.

CAPÍTULO 5

DESCRIPCIÓN DEL SOFTWARE DEL SISTEMA

5.1 Software del microcontrolador Basic Stamp 2sx

El software implementado en el microcontrolador Basic Stamp 2sx debió cumplir las especificaciones del protocolo Modbus, que se describen en el apéndice 2.

La figura 5.1 muestra el funcionamiento general del programa desarrollado. Primero se realiza una rutina de inicialización, luego el programa entra en estado de espera de señales Modbus. Si algún mensaje es recibido se procede a verificar que la dirección de unidad terminal remota es la correspondiente al dispositivo. Una vez verificada la dirección, se procede a leer el espacio de función del mensaje Modbus recibido, y dependiendo de este valor, se salta a las diferentes rutinas (lectura o escritura). Al inicio de cada rutina, se revisa que el valor de los caracteres de Chequeo Cíclico de Error (CRC por sus siglas en inglés) sea el correcto, lo cual indicaría que el mensaje recibido no contiene errores. Una vez terminada la rutina correspondiente a la función requerida, se procede a enviar el mensaje de respuesta, para luego saltar al inicio del programa, para que la UTR espere el siguiente mensaje.

5.1.1 Rutina de inicialización

La rutina de inicialización lee la dirección asignada a la unidad terminal remota, y el modo de operación que debe tener. Para esto, lo que se hace es que se obtienen los datos escritos en los conjuntos de interruptores colocados para dichos fines[4].

Estos datos son almacenados en variables, las cuales se utilizarán más adelante en el programa.

La figura 5.2 muestra el diagrama de flujos de la rutina de inicialización, mediante el cual se puede entender claramente su funcionamiento.

[4] En el capítulo 4 se describe el funcionamiento de estos interruptores.

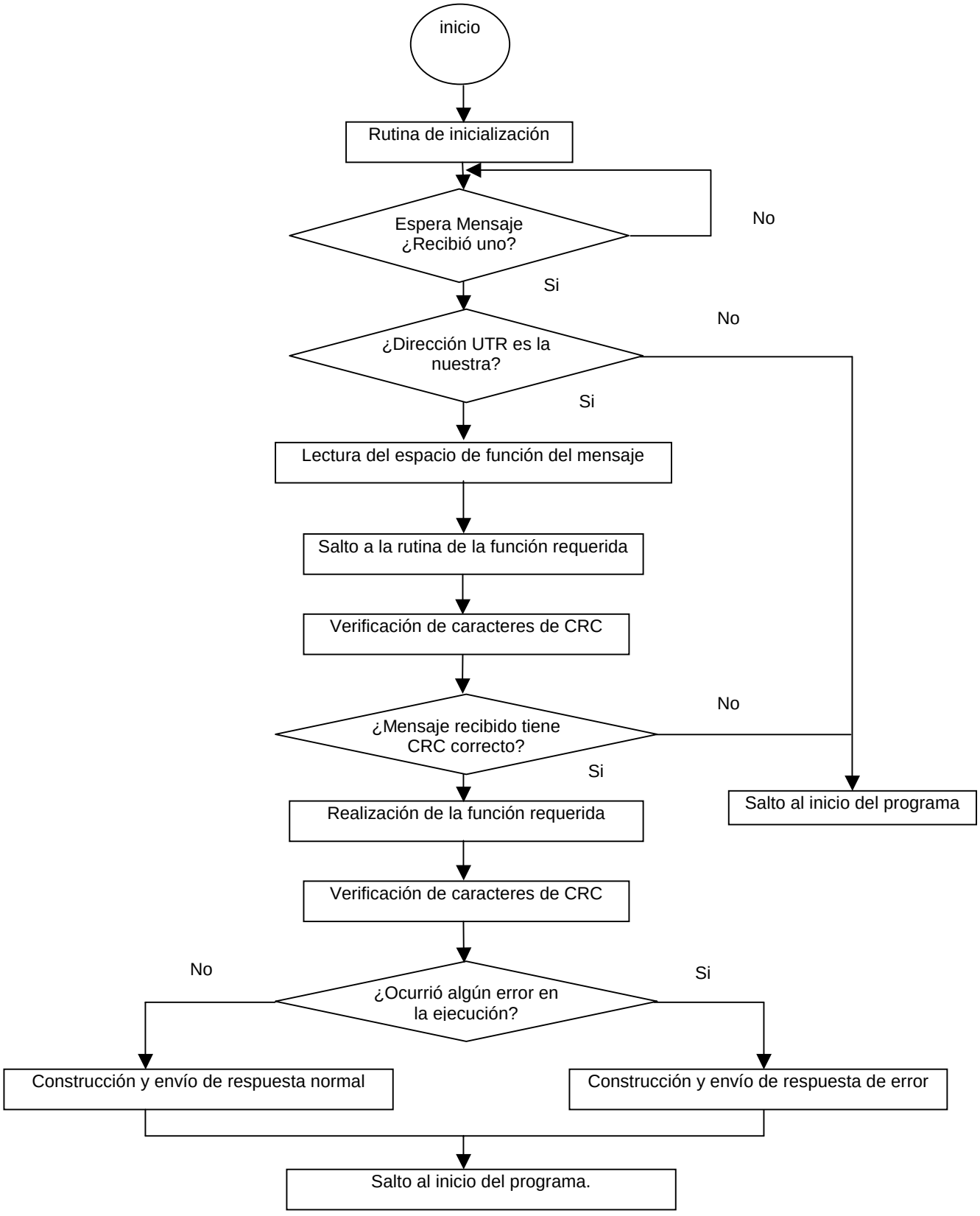

Figura 5.1 Diagrama de Flujo General del programa del Microcontrolador Basic Stamp 2sx.

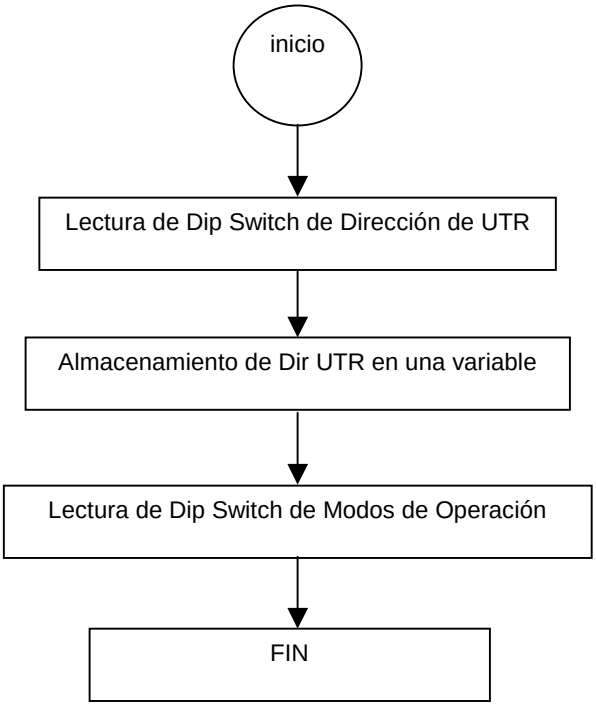

Figura 5.2 Diagrama de Flujo de la rutina de inicialización del programa del microcontrolador.

5.1.2 Rutinas de función

El protocolo Modbus define una serie de 24 instrucciones o funciones. En el apéndice 2, en la tabla A2.1 se muestra el lugar en el que se coloca la función de un mensaje Modbus. En el presente proyecto se implementaron únicamente 8 funciones. A continuación se describen las rutinas de ejecución correspondientes a cada una de estas funciones.

Lectura de bobinas (señales digitales)

La rutina de lectura de bobinas corresponde a la función 1 del protocolo Modbus.

En el desarrollo del proyecto se definió que el espacio de memoria donde se colocarán las bobinas sería el comprendido entre las posiciones 1000 y 2000 decimal.

En la figura 5.3 puede observarse el diagrama de flujo de esta rutina. Primero, se procede a enviar una señal de requerimiento de comunicación a la CPU de la unidad terminal remota. Una vez que se ha recibido respuesta, se le envía un mensaje a la CPU en el que se le indica que se realizará una lectura de memoria. Luego de obtener una respuesta positiva de la CPU, se procede a calcular la dirección de la bobina que se requiere leer. Finalmente, se realiza la lectura, con lo cual se concluye esta subrutina.

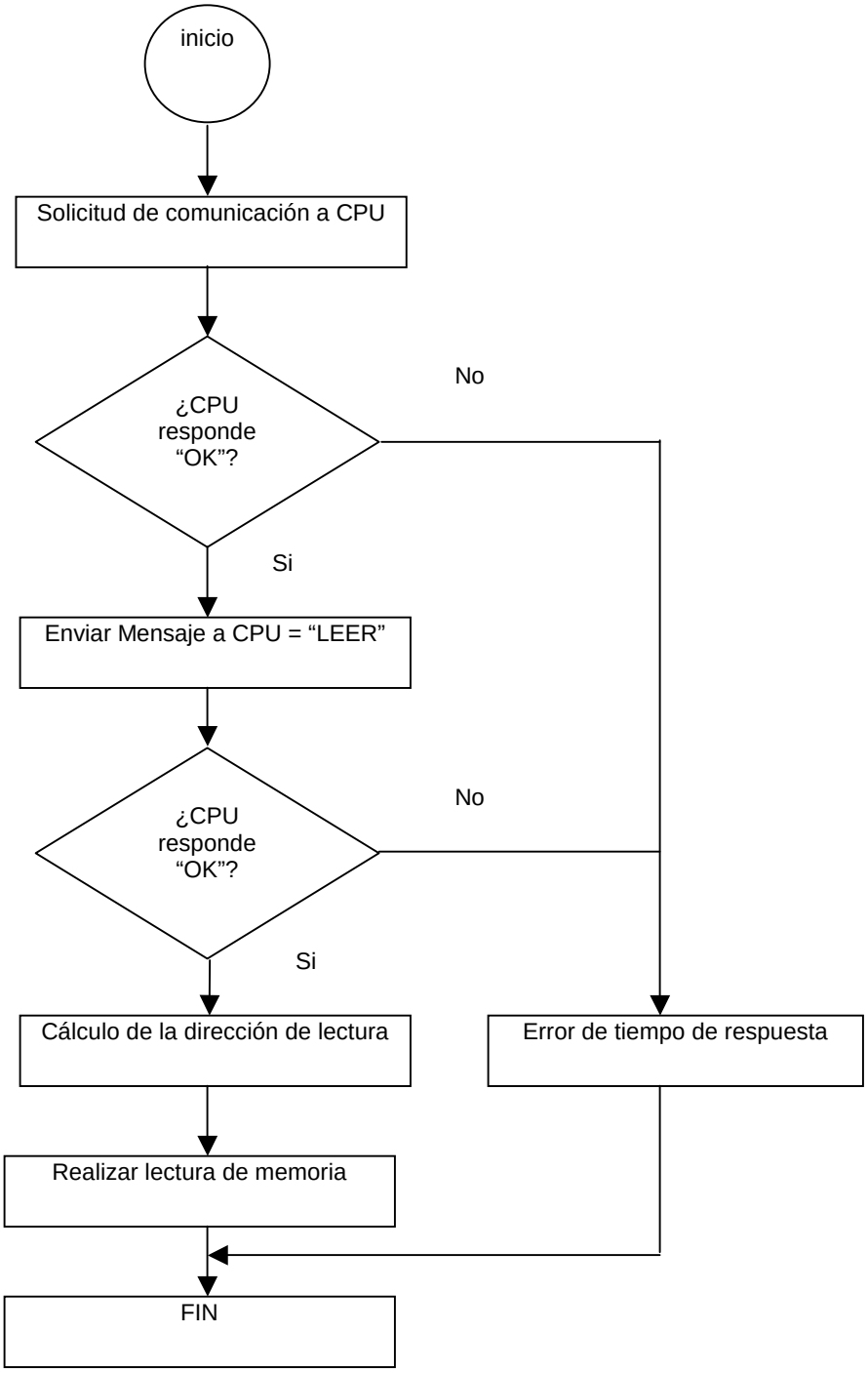

Figura 5.3 Diagrama de flujo de las rutinas de lectura del microcontrolador

Lectura de entradas discretas (señales digitales)

La rutina de lectura de entradas discretas corresponde a la función 2 del protocolo Modbus.

Su ejecución es idéntica a la función anterior, por lo que no se incluye otro diagrama de flujos con su explicación.

La diferencia que se presenta con la función anterior, es que las entradas discretas se colocarán entre las direcciones 2000 y 3000 (decimal) de la memoria RAM de la UTR. Además, las entradas discretas son de sólo lectura, mientras que los valores de las bobinas puede ser forzado.

Lectura de registros de retención (señales analógicas)

Esta rutina corresponde a la función 3 del protocolo Modbus, lectura de registros de retención (holding registers, por su nombre en inglés).

El espacio de memoria donde se colocarán los datos contenidos en estos registros será a partir de la dirección 3000 decimal.

El diagrama de flujo de esta subrutina es idéntico al de la función 1 (lectura de bobinas), mostrado en la figura 5.3.

Lectura de registros de entrada (señales analógicas)

La función cuatro de Modbus corresponde a la lectura de registros de entrada. Estos registros se diferencian de los registros de retención en el hecho de que son únicamente de lectura.

El diagrama de flujo que aparece en la figura 5.3 describe el funcionamiento de esta subrutina.

Escritura de una bobina (señal digital)

La función Modbus número 5 corresponde a la escritura de una señal digital.

Como se puede apreciar en la figura 5.4, en esta subrutina primero se procede a solicitarle a la CPU de la UTR que se establezca comunicación. Una vez que se ha recibido respuesta, se indica a la CPU que lo que se requiere es realizar la escritura de una señal digital. Una vez que se ha recibido confirmación de la CPU, se procede a enviarle los datos de la escritura (dirección y estado de la bobina). Una vez terminado esto, se esperará una señal de confirmación de la CPU. Con esto termina la ejecución de la presente subrutina.

Escritura de un registro de retención (señal analógica)

La función Modbus número 6 corresponde a la escritura de un registro de retención.

El modo de funcionamiento se describe en la figura 5.5. Primero se envía a la CPU una señal de requerimiento de comunicación. Una vez que la CPU responde, se le envía un mensaje indicando que se realizará una escritura de un registro de retención. Luego de que la CPU confirme que está preparada, se procede a enviar los datos de escritura, los cuales consisten en la dirección y el valor del registro. Si la CPU pudo escribir correctamente los datos, entonces enviará al traductor una señal indicadora. Con esta acción se terminaría la ejecución de esta subrutina.

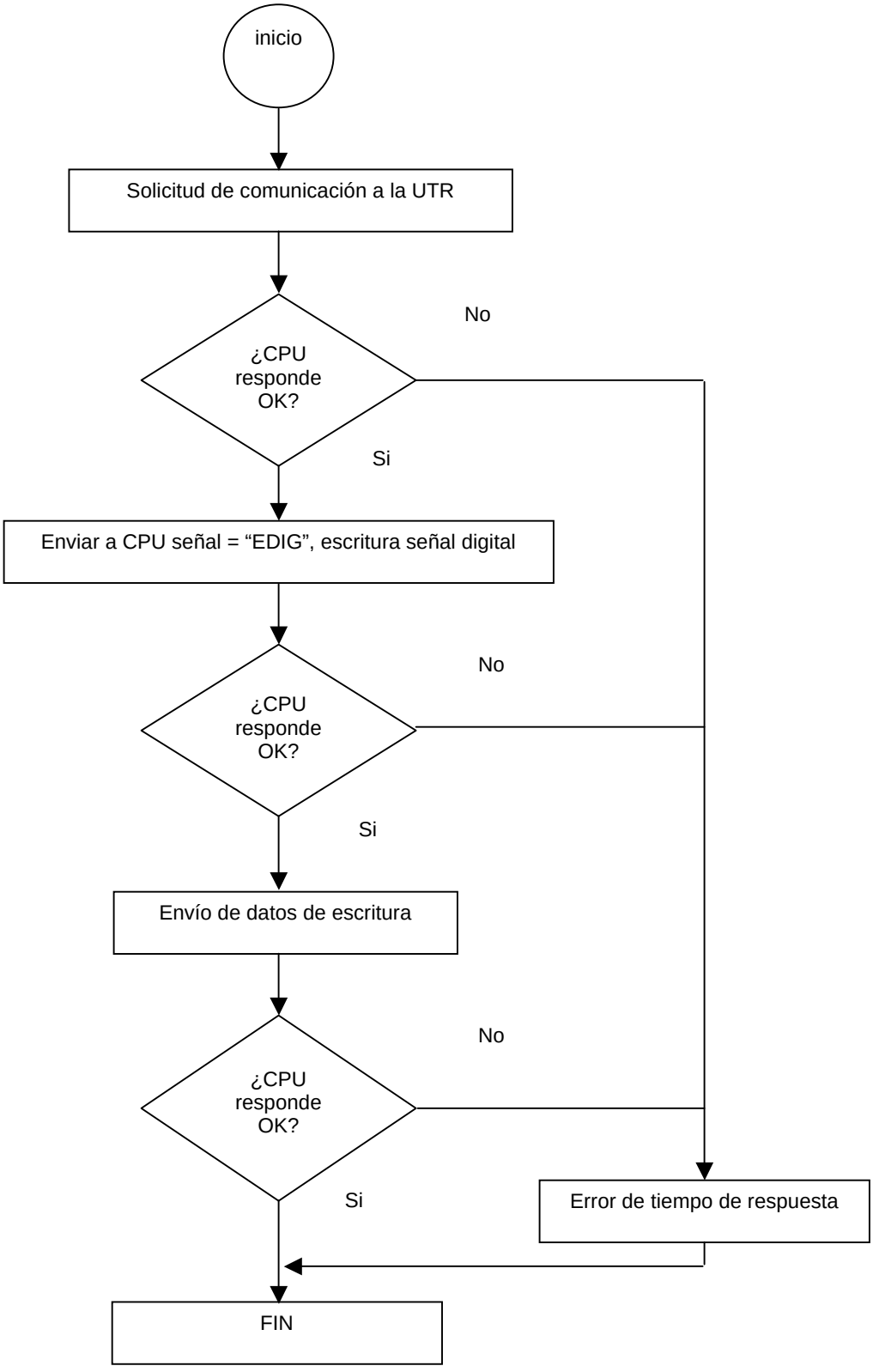

Figura 5.4 Diagrama de flujo de la rutina de escritura de una bobina.

Escritura de varias bobinas (señales digitales)

La función Modbus número 15 corresponde a la escritura de varias bobinas.

En la figura 5.6 puede apreciarse el diagrama de flujo que describe el funcionamiento de esta rutina.

Inicialmente, se coloca en una variable el número de bobinas que se escribirán. Luego se procede a solicitarle comunicación al CPU. Seguidamente se envía una señal al CPU en el que se le indique que se realizará una escritura de una señal digital. Si la CPU responde correctamente, entonces se enviará los datos de escritura, conformados por la dirección y el estado de la bobina. Si la CPU envía una señal que indique que se realizó con éxito la escritura, entonces se procederá a decrementar en uno la variable que almacena el número de bobinas que se deben escribir, y se saltará al inicio de la subrutina para continuar con la escritura de la siguiente bobina. La presente subrutina finaliza una vez que se ha realizado la escritura de todas las bobinas requeridas.

Escritura de varios registros de retención (señales analógicas)

La escritura de varias señales analógicas corresponde a la función 16 del protocolo Modbus.

El funcionamiento de esta rutina es descrito por el diagrama de flujo de la figura 5.7.

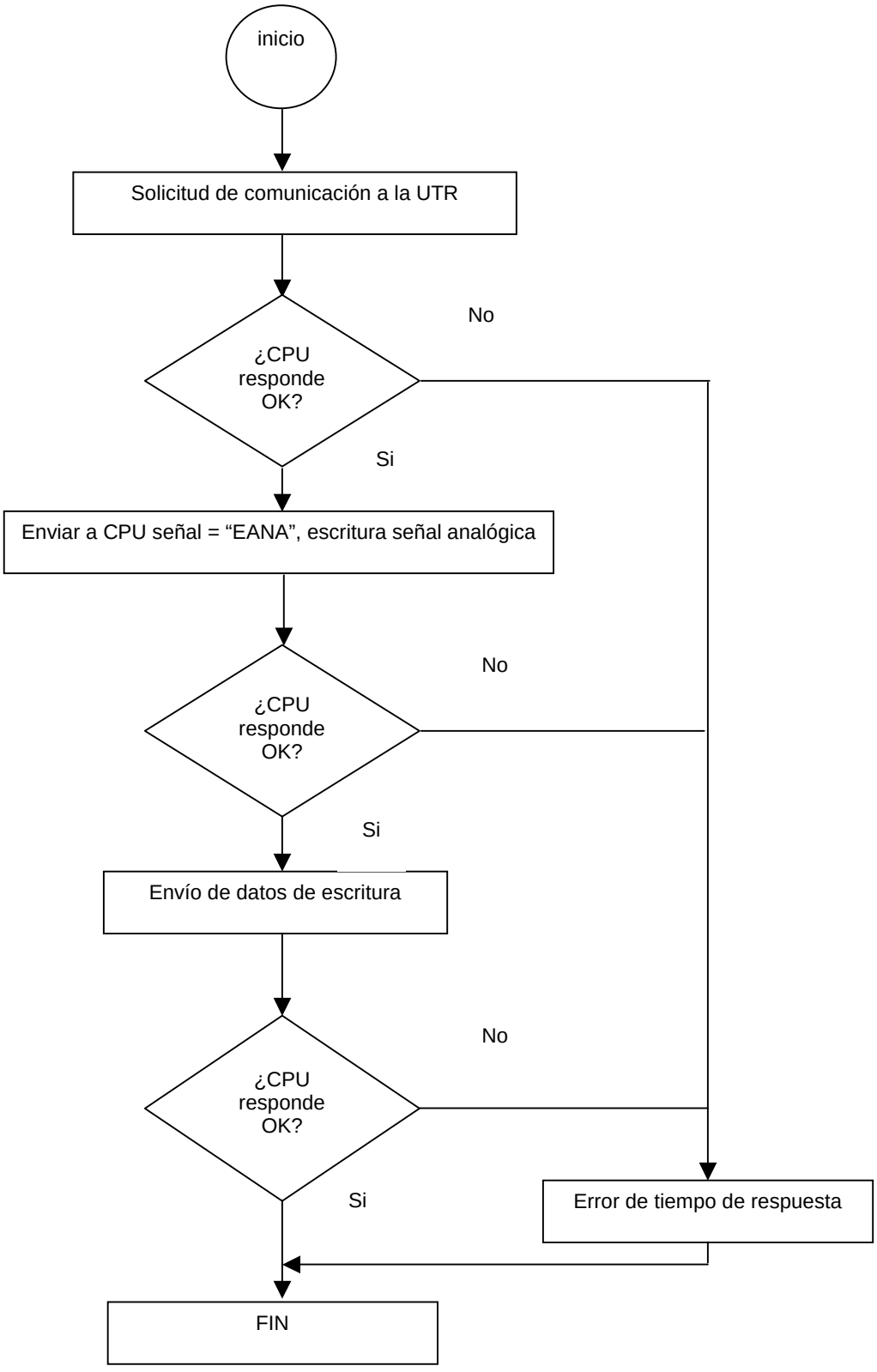

Figura 5.5 Diagrama de flujo de la rutina de escritura de un registro de retención.

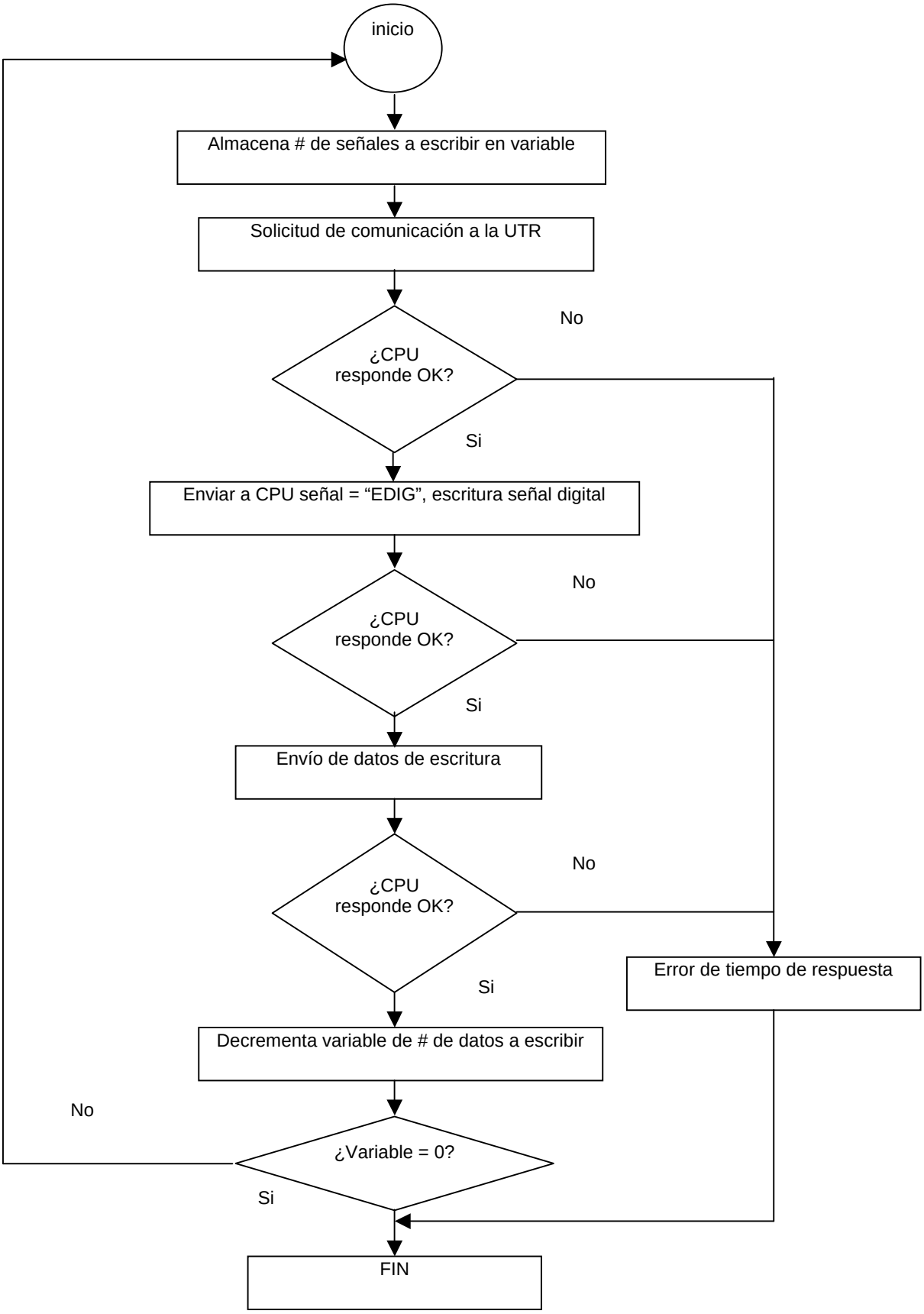

Figura 5.6 Diagrama de flujo de la rutina de escritura de varias señales digitales

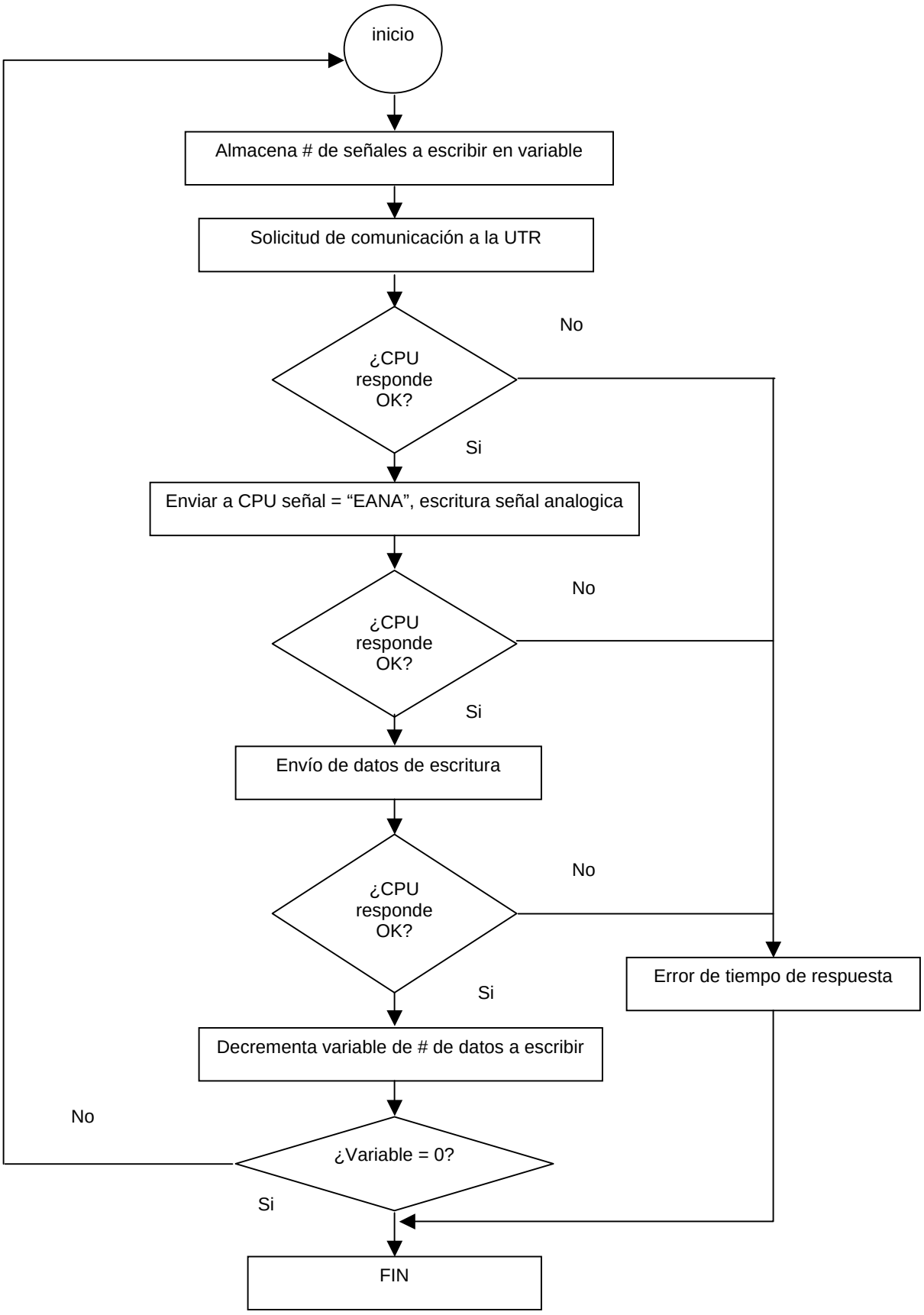

Figura 5.7 Diagrama de flujo de la rutina de escritura de varias señales analógicas.

5.1.3 Rutina de verificación de error (Chequeo de Redundancia Cíclica)

El protocolo Modbus establece 2 modos diferentes de organización de los datos a transmitir: el modo ASCII y el modo UTR. Para cada uno de los modos de operación del protocolo, se establece un método diferente para el chequeo de los errores en las comunicaciones. Como se aprecia en el apéndice 2 del presente informe, el protocolo Modbus indica que en modo UTR, el chequeo de error en las comunicaciones debe realizarse mediante la técnica de chequeo de redundancia cíclica (CRC).

Para realizar el chequeo de redundancia cíclica, se utilizó un algoritmo facilitado en la página de Internet de Modicon. En la bibliografía del presente informe se facilita la dirección del sitio consultado. Dicho algoritmo consiste en la utilización de dos tablas para realizar rápidamente el cálculo de los caracteres de chequeo de error.

La figura 5.8 muestra el diagrama de flujo que describe el funcionamiento del algoritmo de cálculo de los caracteres de chequeo de error.

Primero se coloca el mensaje al que se le chequeará el CRC en un arreglo de datos, llamado “mensaje”, luego se colocan 2 variables en 255 (FF hexadecimal). Estas variables son llamadas CRC_HI y CRC_LO. Seguidamente, la variable “i” es inicializada en cero. Después se realiza el ciclo mostrado en la figura 5.8. Al terminar dicho ciclo, las variables CRC_HI y CRC_LO tendrán los valores de chequeo de redundancia cíclica correspondientes al mensaje.

La tabla 5.1 proporciona los valores a colocar en las tablas del algoritmo de chequeo de redundancia cíclica.

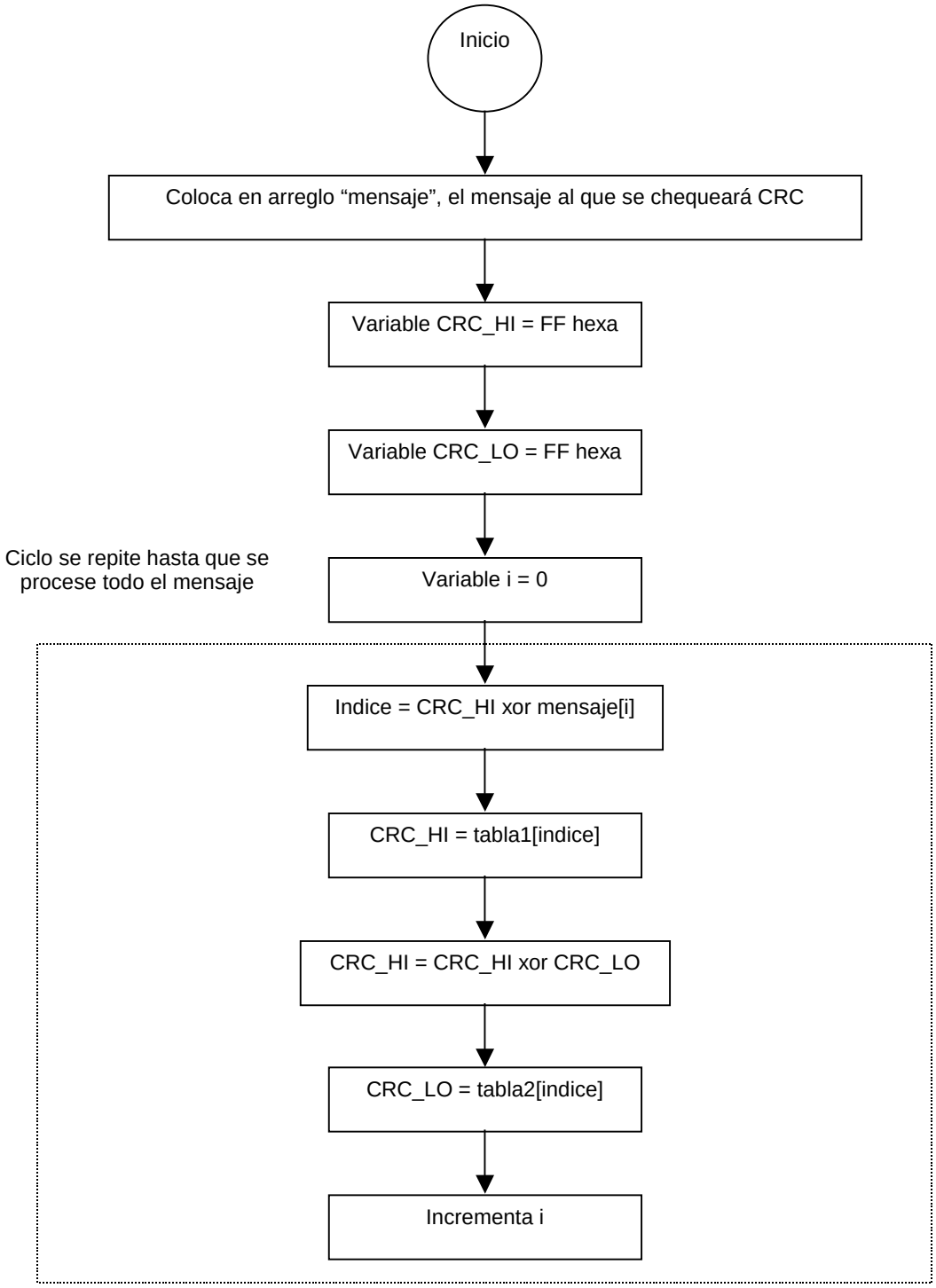

Figura 5.8 Diagrama de flujo de la rutina de chequeo de error (chequeo de redundancia cíclica).

Tabla 5.1 Tablas de comparación para el algoritmo de cálculo de chequeo de redundancia cíclica (los valores están en hexadecimal).

Número de tabla	Contenido de la tabla
Tabla 1	00,C1,81,40,01,C0,80,41,01,C0,80,41,00,C1,81,40,01,C0,80,41,00,C1,81,40,00,C1,81,40,01,C0,80,41,01,C0,80,41,00,C1,81,40,00,C1,81,40,01,C0,80,41,01,C0,80,41,00,C1,81,40,01,C0,80,41,00,C1,81,40,00,C1,81,40,01,C0,80,41,00,C1,81,40,01,C0,80,41,01,C0,80,41,00,C1,81,40,00,C1,81,40,01,C0,80,41,00,C1,81,40,00,C1,81,40,01,C0,80,41,01,C0,80,41,00,C1,81,40,00,C1,81,40,01,C0,80,41,00,C1,81,40,01,C0,80,41,01,C0,80,41,00,C1,81,40,01,C0,80,41,00,C1,81,40,00,C1,81,40,01,C0,80,41,01,C0,80,41,00,C1,81,40,01,C0,80,41,00,C1,81,40,00,C1,81,40,01,C0,80,41,01,C0,80,41,00,C1,81,40,01,C0,80,41,00,C1,81,40,00,C1,81,40,01,C0,80,41,01,C0,80,41,00,C1,81,40,01,C0,80,41,00,C1,81,40
Tabla 2	00,C0,C1,01,C3,03,02,C2,C6,06,07,C7,05,C5,C4,04,CC,0C,0D,CD,0F,CF,CE,0E,0A,CA,CB,0B,C9,09,08,C8,D8,18,19,D9,1B,DB,DA,1A,1E,DE,DF,1F,DD,1D,1C,DC,14,D4,D5,15,D7,17,16,D6,D2,12,13,D3,11,D1,D0,10,F0,30,31,F1,33,F3,F2,32,36,F6,F7,37,F5,35,34,F4,3C,FC,FD,3D,FF,3F,3E,FE,FA,3A,3B,FB,39,F9,F8,38,28,E8,E9,29,EB,2B,2A,EA,EE,2E,2F,EF,2D,ED,EC,2C,E4,24,25,E5,27,E7,E6,26,22,E2,E3,23,E1,21,20,E0,A0,60,61,A1,63,A3,A2,62,66,A6,A7,67,A5,65,64,A4,6C,AC,AD,6D,AF,6F,6E,AE,AA,6A,6B,AB,69,A9,A8,68,78,B8,B9,79,BB,7B,7A,BA,BE,7E,7F,BF,7D,BD,BC,7C,B4,74,75,B5,77,B7,B6,76,72,B2,B3,73,B1,71,70,B0,50,90,91,51,93,53,52,92,96,56,57,97,55,95,94,54,9C,5C,5D,9D,5F,9F,9E,5E,5A,9A,9B,5B,99,59,58,98,88,48,49,89,4B,8B,8A,4A,4E,8E,8F,4F,8D,4D,4C,8C,44,84,85,45,87,47,46,86,82,42,43,83,41,81,80,40

5.2 Software de aplicación en lenguaje Delphi

El software de aplicación se dividió en dos partes, ya que se requirió simular mediante computadoras los dos dispositivos con que se comunica el traductor de protocolo Modbus: el CPU de la unidad terminal remota y la computadora central. Esta última se comunica con el traductor a través de un módem.

5.2.1 Software de aplicación 1: Simulador de la Computadora Central

Este programa se hizo para simular a la Computadora Central, la cual debe ser capaz de enviar los requerimientos de lectura y escritura en protocolo Modbus, y recibir los mensajes de respuesta.

La figura 5.9 muestra el diagrama de flujo del funcionamiento general del programa de aplicación.

Primero se inicializa el puerto serial de la computadora, mediante el cual se enviará los mensaje al traductor. El puerto debe transmitir los datos a una velocidad de 9600 baudios[5], con 8 bits[6] de información, sin bit de paridad, y debe enviar dos bits de parada (stop bits, por su nombre en inglés), para asegurar el óptimo funcionamiento del sistema. El bit de paridad consiste en una señal binaria que se utiliza para verificar posibles errores en datos almacenados en arreglos de bits. Un bit de parada es una señal binaria que indica la finalización de una transmisión de un paquete de datos en una comunicación serial.

---

[5] El baudio es la unidad con la que se mide la velocidad de transmisión de datos a través de un puerto serial. 1 baudio = 1 bit / segundo

[6] Un bit es una señal binaria, la cual puede tener 2 estados únicamente. En lenguaje lógico, se establece que los valores de una señal binaria pueden ser uno o cero.

Luego, se despliega en pantalla las opciones que existen para enviar mensajes Modbus: lectura de bobinas, lectura de entradas discretas, lectura de registros de entrada, lectura de registros de retención, escritura de bobinas, y escritura de registros de retención.

Seguidamente, se salta a las distintas rutinas correspondientes a las diferentes instrucciones Modbus. En cada una se procede a solicitar al usuario del programa que digite los valores de la acción que desea realizar: dirección de lectura/escritura, y valor de escritura, en el caso de que se vaya a realizar esta última acción.

Después de esto, se envía el mensaje Modbus a través del puerto serial de la computadora. Se espera un tiempo de 2 segundos y luego se procede a recibir el mensaje de respuesta que debió enviar el traductor.

Finalmente, se procede a desplegar en pantalla los datos enviados y los recibidos.

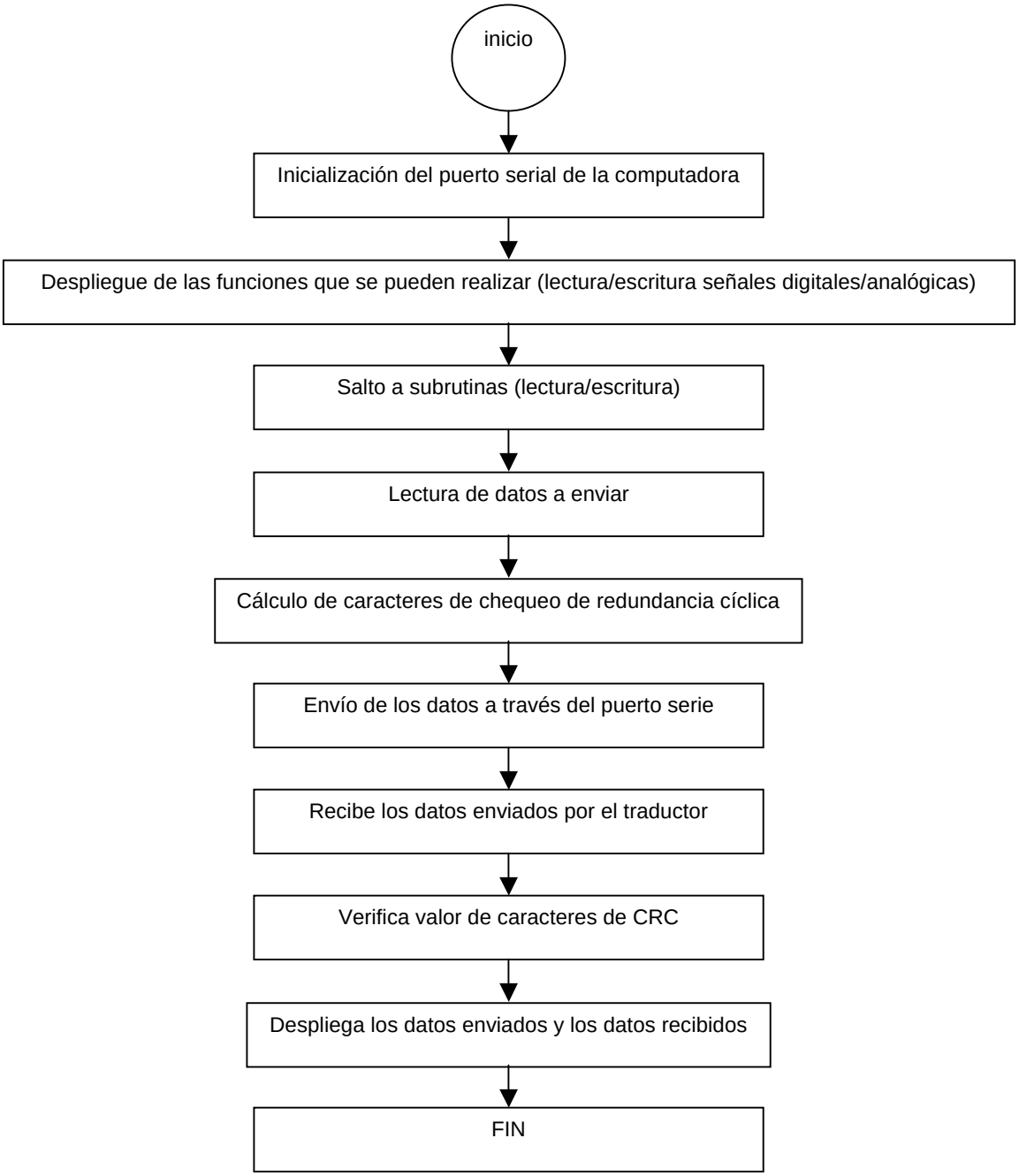

Figura 5.9 Diagrama de flujo del programa de aplicación simulador de la Computadora Central.

5.2.2 Software de aplicación 2: Simulador de la CPU

El segundo software de aplicación debió simular a la CPU de la unidad terminal remota.

La comunicación entre el traductor y la CPU se lleva a cabo a través de tres señales. Dos corresponden a las señales de transmisión serial, y una tercera corresponde a la señal de requerimiento de comunicación, la cual se envía mediante la utilización del puerto 13 del microcontrolador.

Con el fin de simular la señal de requerimiento de comunicación, en el hardware se conectó un LED[7] a la salida del puerto 13 del microcontrolador. Cuando este diodo se enciende, el operador del programa debe señalar con el ratón de la computadora, en el botón que aparece en la pantalla de la misma, para iniciar así el proceso de comunicación entre el traductor y el programa simulador de la CPU.

La figura 5.10 muestra el diagrama de flujo del programa de aplicación simulador de la CPU de la unidad terminal remota.

[7] LED: diodo emisor de luz, por sus siglas en inglés.

El programa primero inicializa el puerto serial de la computadora que se está utilizando. Luego, despliega en la pantalla un botón y entra en estado de espera. Entonces, cuando el usuario del programa seleccione dicho botón con el ratón de la computadora, el programa enviará a través del puerto serie una señal conteniendo los valores ASCII de las letras “OK”. Esto se hace debido a que a través del puerto serial de la computadora no se puede chequear el estado de la señal de requerimiento de comunicación, enviada a través del puerto 13 del microcontrolador. Seguidamente, el programa recibe la respuesta enviada por el traductor. Si dicha respuesta es una indicación de lectura de memoria, representada por los caracteres ASCII de las letras “LEER”, entonces se envía una señal de “OK” y el programa finaliza. Si la respuesta es una señal de escritura analógica o digital, representadas por los caracteres ASCII “EDIG” o ”EANA” respectivamente, entonces se procede a enviar la señal de “OK” y luego recibir los datos de escritura.

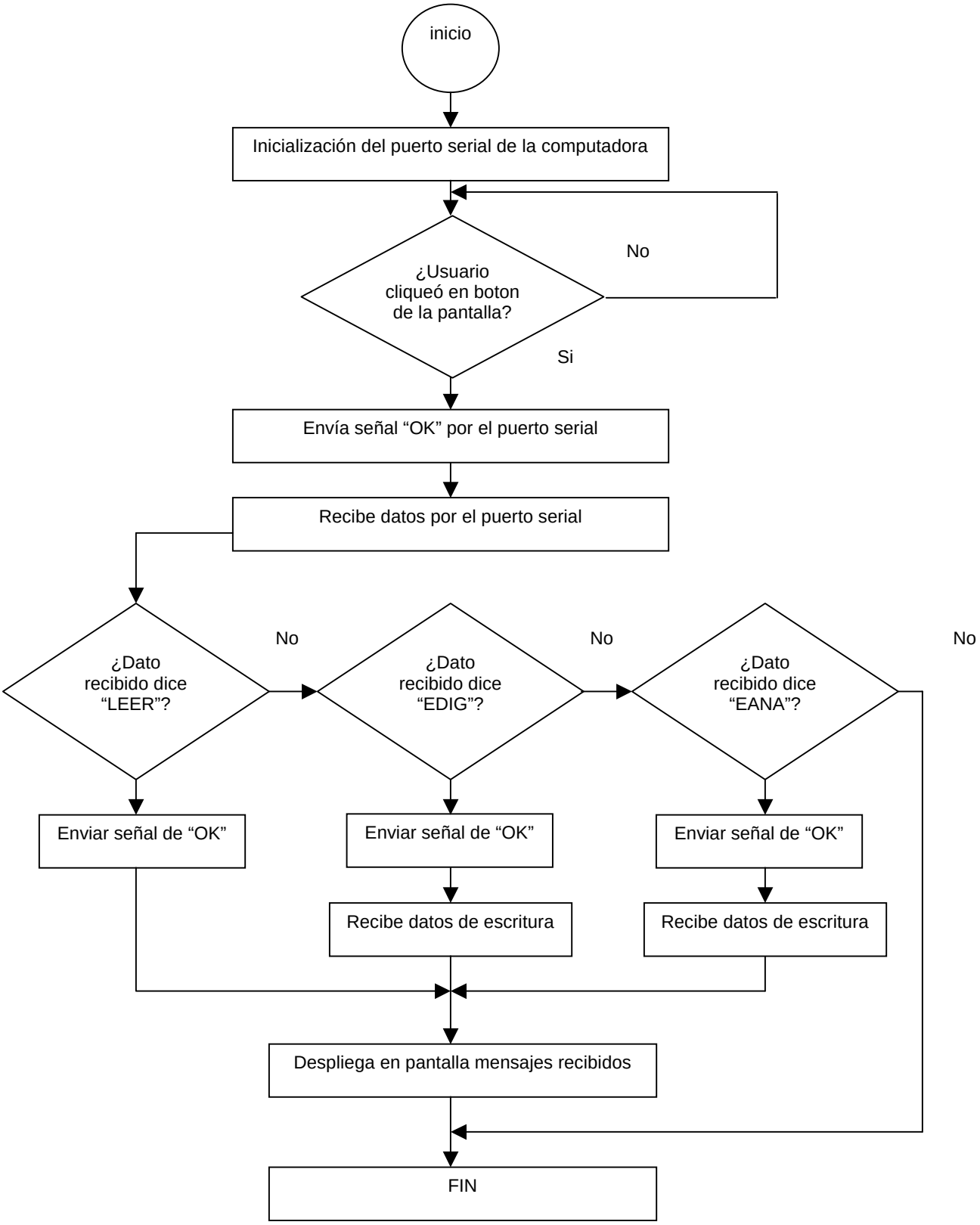

Figura 5.10 Diagrama de flujo del programa de aplicación simulador de la CPU.

CAPÍTULO 6

ANÁLISIS Y RESULTADOS

6.1 Explicación del diseño

El proyecto realizado se describe en todas sus partes en las siguientes secciones.

6.1.1 Hardware del dispositivo traductor

El hardware del dispositivo traductor que se diseñó es gobernado por el microcontrolador Basic Stamp 2sx, como se puede apreciar en la figura 4.1.

Se requirió que el dispositivo traductor se comunicara con la computadora central del sistema SCADA, a una velocidad de 9600 baudios, a través de un módem. Para resolver este requerimiento se emplearon los pines SIN y SOUT del microcontrolador, los cuales pueden conectarse directamente a un puerto RS-232 sin necesidad de hardware adicional.

El dispositivo diseñado debió comunicarse también con la CPU de la unidad terminal remota. Para realizar dicha comunicación se utilizaron tres líneas, una para enviar una señal de “requerimiento de comunicación” y las dos restantes para enviar y recibir datos en forma serial. Para esto se utilizaron los puertos 13,14 y 15 del microcontrolador. Para la comunicación serial, fue necesario emplear un acople mediante el dispositivo TC232CPE, ya que los puertos 14 y 15 del microcontrolador no pueden conectarse directamente a un puerto serial RS-232.

Para implementar las rutinas de lectura del protocolo Modbus, fue necesario que el dispositivo traductor diseñado tuviera acceso a la memoria RAM de la CPU de la unidad terminal remota.

La memoria RAM del sistema se encuentra normalmente gobernada por la CPU. Ésta le inserta constantemente los datos leídos de las variables físicas que controla y supervisa.

Para poder realizar una lectura de memoria se requirió emplear registros con tercer estado, para no generar conflictos en los buses de dirección y datos de la memoria RAM.

Otro requerimiento del hardware del proyecto fue la necesidad de incluir dos conjuntos de interruptores. Uno para escribir manualmente la dirección de la unidad terminal remota a la que pertenece el traductor y otro para que el usuario escribiera los modos de operación que se desean para el sistema.

El circuito implementado posee buses con tercer estado para realizar la lectura de la dirección de la unidad terminal remota y los modos de operación del sistema.

El microcontrolador utilizado fue el Basic Stamp 2sx, el cual tiene 16 puertos que se pueden utilizar como entradas o salidas. En el desarrollo del hardware, fue necesario manejar una gran cantidad de dispositivos, tales como la memoria RAM del sistema y los buses de lectura de dirección de unidad terminal remota y modos de operación. Se requirió manejar esos dispositivos con los puertos mencionados anteriormente. Como puede observarse en la tabla 6.1, la cantidad de señales que se requirió para utilizar los diferentes dispositivos que formaron parte del hardware del sistema es mayor que la cantidad de puertos del microcontrolador, por lo que tuvo que idearse una estrategia para poder manejarlas con los puertos disponibles. La solución que se encontró para esto, fue la utilización de un decodificador de binario a decimal 74LS445. Como se observa en la figura 4.1, mediante el empleo de tres señales enviadas por el microcontrolador, se pudieron accesar todos los dispositivos necesarios a través del decodificador mencionado.

Se utilizaron los puertos 0 al 7 del microcontrolador como bus de datos, mediante el cual se ingresan y obtienen datos de los distintos registros, buses e interruptores del hardware.

El puerto 8 se reservó para la activación-desactivación de los registros que contienen la dirección de memoria a accesar.

Los puertos 9 a 11 del microcontrolador se utilizaron para direccionar el decodificador, el cual se utiliza para activar los diferentes dispositivos de interfaz con la memoria RAM del sistema, y con los arreglos de interruptores de dirección de unidad terminal remota y modos de operación.

Tabla 6.1 Señales requeridas para el control de los dispositivos del hardware del dispositivo traductor

Nombre del dispositivo	Número de señales requeridas
Memoria RAM de la unidad terminal remota	13 señales de dirección 8 señales de datos 4 señales de control
Bus habilitador de interruptores de dirección de unidad terminal remota	8 señales de datos 1 señal de control
Bus habilitador de interruptores de modos de operación	8 señales de datos 1 señal de control
Puerto serie de comunicación con la computadora central	2 señales seriales
Puerto serie de comunicación con la CPU de la unidad terminal remota	2 señales seriales 1 señal de “requerimiento de comunicación”

El puerto 13 del microcontrolador se utilizó como la señal de “requerimiento de comunicación” que el dispositivo traductor envía a la CPU de la unidad terminal remota.

Los puertos 14 y 15 se utilizaron como interfaz de comunicación serial entre el dispositivo traductor y la CPU de la unidad terminal remota.

La tabla 2.1 indica que los pines SIN y SOUT del microcontrolador utilizado funcionan como entrada y salida serial, respectivamente. En la figura 2.3 puede apreciarse la ubicación de dichos pines. En el proyecto desarrollado, estas señales se utilizaron para comunicar al dispositivo traductor diseñado con la computadora central del sistema SCADA.

Para manejar las direcciones de memoria RAM, se utilizaron dos registros 74LS573, los cuales tienen tercer estado[8]. Fue necesario utilizar dos registros porque el bus de datos que se definió tiene 8 señales, y la memoria RAM tiene 13 señales de dirección. Por este motivo, para escribir una dirección nueva en la memoria RAM, se requirió de dos ciclos de escritura a registro. Esto hace que el acceso a la memoria sea mas lento que si se hubiera utilizado únicamente un registro. Sin embargo, la ventaja que existe es que se utilizan menos puertos del microcontrolador para realizar la operación de direccionamiento de memoria RAM.

Para accesar el bus de datos de la memoria RAM del sistema se utilizó un bus con tercer estado 74LS541.

La memoria RAM tiene cuatro señales de control, mediante las cuales se pueden realizar las operaciones de lectura y escritura de la misma. El dispositivo diseñado únicamente puede realizar lecturas a la memoria. Por lo anterior, se utilizó un bus 74LS541 con valores prefijados a la entrada para activar las señales de lectura de la memoria.

[8] Tercer estado: estado de alta impedancia que presentan las salidas de algunos circuitos integrados.

Como se puede observar en la figura 4.1, los distintos registros y buses de tercer estado se activan mediante las señales emitidas por el decodificador 74LS445. Dependiendo de la salida que tenga activa dicho decodificador, se activan o desactivan la lectura de memoria RAM, el direccionamiento de la memoria RAM, la lectura de dirección de unidad terminal remota y la lectura de modos de operación. Como se mencionó anteriormente, el microcontrolador controla al decodificador mediante los puertos 9, 10 y 11. Debe recalcarse que si todas las entradas al decodificador se encuentran en cero, entonces todos los dispositivos con acceso al bus de datos y direcciones del sistema se encontrarán en estado de alta impedancia.

Finalmente, para realizar pruebas de funcionamiento del proyecto, se requirió implementar un circuito en tabla de prototipo para escritura manual de la memoria RAM, para así poder verificar que los datos leídos de la misma correspondían a los datos escritos por el usuario. Para esto se empleó una serie de interruptores y buses de tercer estado, los cuales se encienden manualmente.

6.1.2 Software del dispositivo traductor

Para el diseño del software del dispositivo traductor, se siguieron las reglas definidas por el protocolo Modbus, las cuales se adjuntan en el apéndice 2 del presente informe.

Se requirió que el dispositivo traductor estuviese en constante monitoreo del puerto serial con el que se comunica con la computadora central, a la espera de recibir un mensaje válido. Si se recibe un mensaje erróneo o correspondiente a otra unidad terminal remota, entonces el traductor no realiza ninguna acción y debe continuar monitoreando el puerto serie.

Cada vez que se recibe un mensaje Modbus, el programa del traductor debe verificar que los caracteres de chequeo de redundancia cíclica estén correctos. De lo contrario, se asume que el mensaje recibido contiene un error, por lo que no se realiza ninguna acción y se entra en modo de espera de nuevos mensajes. La rutina de chequeo de redundancia cíclica utilizada es la proporcionada por el fabricante Modicon, cuya dirección electrónica se adjunta en la Bibliografía. Esta rutina se explicó detalladamente en el capítulo 5 del presente informe.

Si algún mensaje válido es recibido, se debe saltar a las subrutinas de lectura de memoria o escritura de variables, las cuales se han explicado detalladamente en el capítulo 5 del presente informe.

Las rutinas que se implementaron en el presente proyecto son las de lectura y escritura de señales digitales y analógicas.

Para realizar las rutinas de lectura, el dispositivo traductor solicita permiso a la CPU de la unidad terminal remota para accesar la memoria. Esto lo hace mediante la activación del puerto 13 del microcontrolador Basic Stamp 2sx. Si recibe respuesta a través del puerto de comunicación serial implementado con los puertos 14 y 15 del microcontrolador, entonces se procede a activar las señales de lectura de memoria RAM. Una vez terminada la lectura, se envía una señal de finalización a la CPU de la unidad terminal remota, para que esta sepa que puede comenzar a utilizar nuevamente la memoria RAM. Todo este procedimiento se hace para evitar que el dispositivo traductor y la CPU intenten utilizar al mismo tiempo la memoria del sistema.

Cabe recalcar que para realizar las rutinas de lectura de memoria RAM, los tiempos de acceso de la misma no representan ninguna limitación para el sistema, ya que el microcontrolador trabaja a una velocidad promedio de 10 000 instrucciones por segundo, mientras que el tiempo mínimo de lectura de la memoria RAM empleada es de 100 nanosegundos.

Para realizar las rutinas de escritura, el programa implementado envía primero una señal de “requerimiento de comunicación” a la CPU de la unidad terminal remota. Si se recibe respuesta adecuada, se procede a enviar uno a uno los datos de escritura, para que sea la CPU quien realice las escrituras necesarias.

En caso que se den errores en la ejecución de las rutinas, el programa implementado envía una señal de error a la computadora central. Como puede observarse en el apéndice 2, el protocolo Modbus define el modo en que tienen que estructurarse los diferentes mensajes de error que pueden ser enviados.

En general, el programa del dispositivo traductor debe comunicarse vía dos puertos seriales con la computadora central del sistema SCADA y con la CPU de la unidad terminal remota. La comunicación se realizó a una velocidad de 9600 bits por segundo, empleando las instrucciones SERIN y SEROUT del microcontrolador. Como se observa en la Tabla A3.2 del Apéndice 3 del presente informe, las instrucciones mencionadas anteriormente permiten la utilización de los puertos del microcontrolador como terminales de transmisión – recepción de un puerto serial.

6.1.3 Software de aplicación

Como se mencionó en el capítulo 5, el software de aplicación se conformó por dos programas diferentes, uno para simular la computadora central del sistema SCADA, y otro para simular la CPU de la unidad terminal remota.

Para ambos programas se requirió utilizar los puertos seriales de las computadoras en la que se corrieron las aplicaciones. Dichos programas se diseñaron para que fueran capaces de enviar y recibir todos los mensajes requeridos por el dispositivo traductor, siguiendo las normas del protocolo Modbus.

Un requerimiento importante que se tomó en cuenta a la hora de diseñar los programas de aplicación, fue la necesidad de desplegar las tramas enviadas y recibidas por los mismos, con el fin de verificar la óptima operación del dispositivo traductor.

El programa de aplicación simulador de la computadora central presenta las diferentes opciones que tiene el usuario para solicitar a la unidad terminal remota que realice una acción. La figura 6.1 muestra la pantalla de la computadora corriendo el programa de aplicación simulador de la computadora central. El usuario puede, mediante la selección de los botones que aparecen en pantalla, realizar cualquiera de las funciones Modbus requeridas. También se incluyó la posibilidad de que el usuario envíe una señal arbitraria a través del puerto serial, esto con el fin de verificar el óptimo funcionamiento de las rutinas de chequeo de error del dispositivo traductor.

La figura 6.2 muestra la pantalla del programa simulador de la computadora central, una vez que se ha seleccionado la opción de lectura de señales digitales. Una vez que el usuario del programa inserte los datos requeridos, el programa que se diseñó procede a calcular los caracteres de chequeo de redundancia cíclica, para luego enviar el mensaje Modbus a través del puerto serie de la computadora.

Luego de que el mensaje ha sido enviado, el programa de aplicación simulador de la computadora central entra en un estado de espera, el cual dura dos segundos. Este período sirve para darle tiempo al dispositivo traductor a que realice la función requerida por la computadora central, y envíe un mensaje de respuesta. Dicho mensaje se coloca en el bus de datos de entrada del puerto serial de la computadora en la que se está corriendo el programa de aplicación.

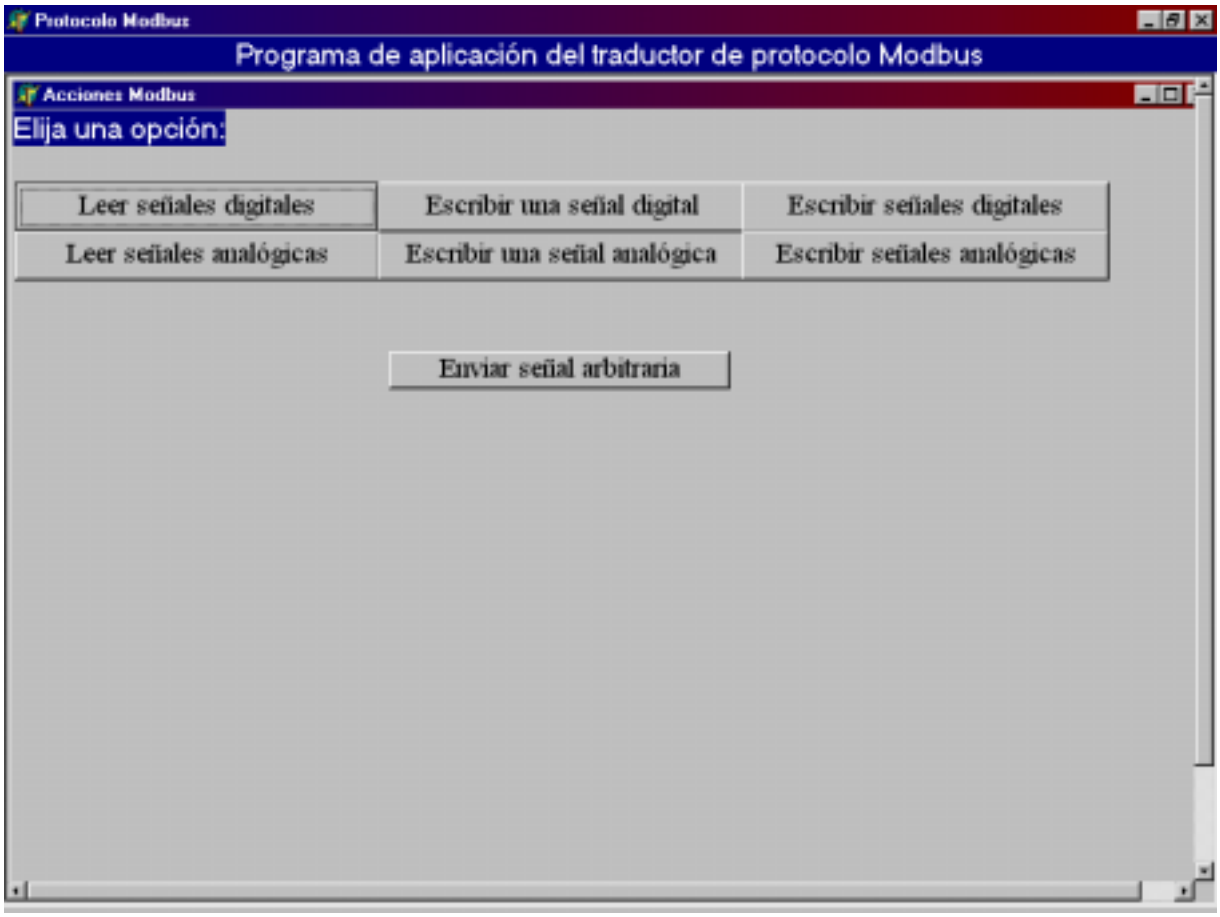

Figura 6.1 Pantalla de opciones del programa de aplicación simulador de la computadora central.

Una vez que ha terminado el lapso de espera asignado, se procede a leer el mensaje enviado por el dispositivo traductor. A dicho mensaje se le verifican los caracteres de chequeo de error.

Por último, se procede a desplegar los datos enviados y los datos recibidos, de la forma que se muestra en la figura 6.3. Los caracteres de los mensajes Modbus se despliegan uno por uno, en forma hexadecimal. Esto se hace para que sea más fácil su lectura. También se despliega un mensaje indicando si los datos recibidos tienen correctos los caracteres de verificación de error.

En la figura 6.3, la cual muestra la pantalla de despliegue de tramas enviadas y recibidas del programa de aplicación simulador de la computadora central del sistema SCADA, puede observarse que se reservaron 77 espacios para desplegar el mensaje recibido. Esto se hizo debido a que por las características del microcontrolador utilizado para el desarrollo del proyecto, el número máximo de datos que se puede enviar en un mensaje Modbus de respuesta es de 77.

Mediante la utilización de este programa se puede entonces enviar mensajes Modbus a través del puerto serial de una computadora, y se puede recibir respuesta. Así se verificó que el dispositivo traductor funcionara de la manera requerida.

Debe recalcarse que el presente programa simula de forma incompleta las funciones de la computadora central del sistema SCADA. Sin embargo, el objetivo del desarrollo del mismo fue la verificación del funcionamiento del dispositivo traductor, mediante el análisis de las tramas de los mensajes enviados y recibidos.

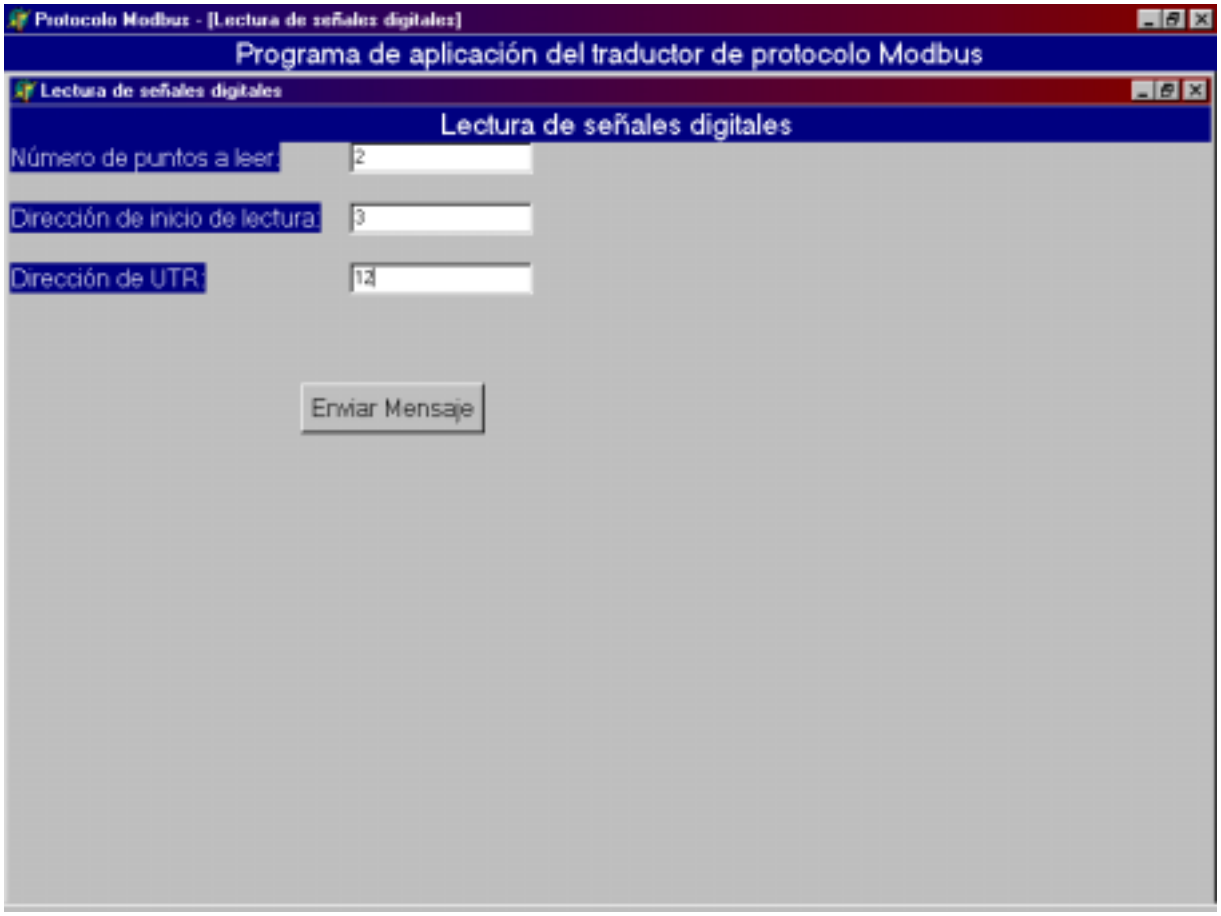

Figura 6.2 Pantalla de lectura de señales digitales del programa de aplicación simulador de la computadora central**.**

El segundo programa de aplicación consistió en la simulación de la CPU de la unidad terminal remota.

Las funciones que se requirieron implementar en dicho programa fueron las de respuesta frente a los requerimientos de comunicación del dispositivo traductor.

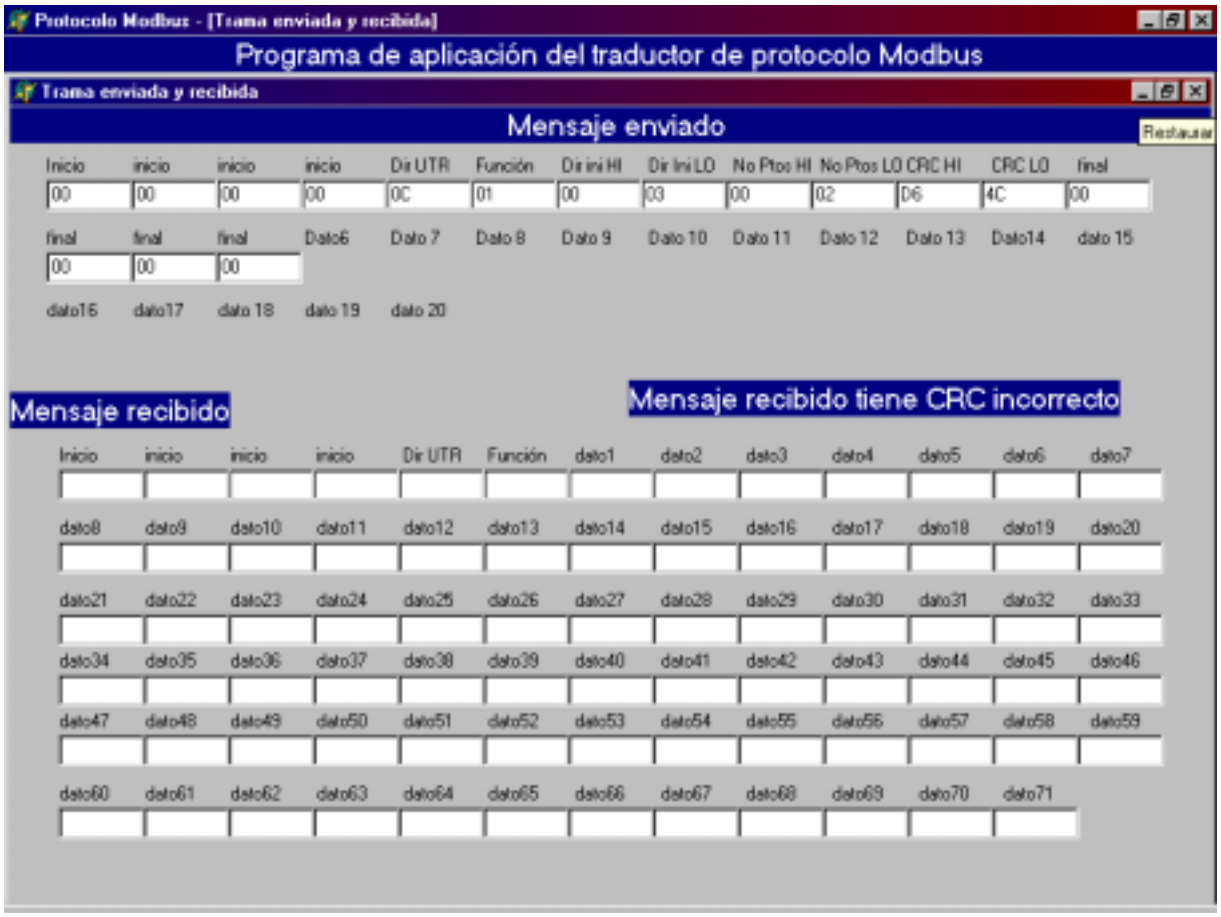

Figura 6.3 Pantalla de despliegue de tramas enviadas y recibidas del programa de aplicación simulador de la computadora central.

Se presentó un problema en el diseño de este programa, ya que el dispositivo traductor envía la señal de “requerimiento de comunicación” mediante la activación del puerto 13 del microcontrolador. Sin embargo, no se encontró una manera en que el programa de aplicación simulador de la CPU pudiera verificar el estado de dicho puerto, para así iniciar las comunicaciones con el dispositivo traductor. Para solucionar este problema, se recurrió a lo mostrado en la figura 6.4. En esta figura se observa que el programa despliega en la pantalla un ícono que dice “enviar señal de OK”. El programa se encuentra en estado de espera y no realiza ninguna acción hasta que el usuario selecciones dicho ícono, luego de lo cual se procede a enviar a través del puerto serial los caracteres ASCII correspondientes a las letras “OK”. El hardware del sistema tiene conectado un LED a la salida del puerto 13 del microcontrolador. Entonces, cuando el usuario del programa observe que dicho LED se enciende, debe seleccionar al ícono desplegado en la pantalla de la computadora. De esta forma se soluciona el problema de la detección de la señal de “requerimiento de comunicación”. El inconveniente que se presenta con esta solución es que se requiere de una persona que esté verificando el estado del LED del puerto 13 del microcontrolador, por lo que la acción de respuesta del programa simulador de la CPU de la unidad terminal remota no es automática.

Luego de que ha sido enviado el primer mensaje hacia el dispositivo traductor, se establece una comunicación con el mismo, en la que se indica cuál es la acción que se va a realizar: lectura de memoria, escritura de señales digitales o escritura de señales analógicas.

El objetivo del desarrollo de este programa fue el despliegue de las señales enviadas y recibidas por la CPU de la unidad terminal remota, para verificar el óptimo funcionamiento del dispositivo traductor.

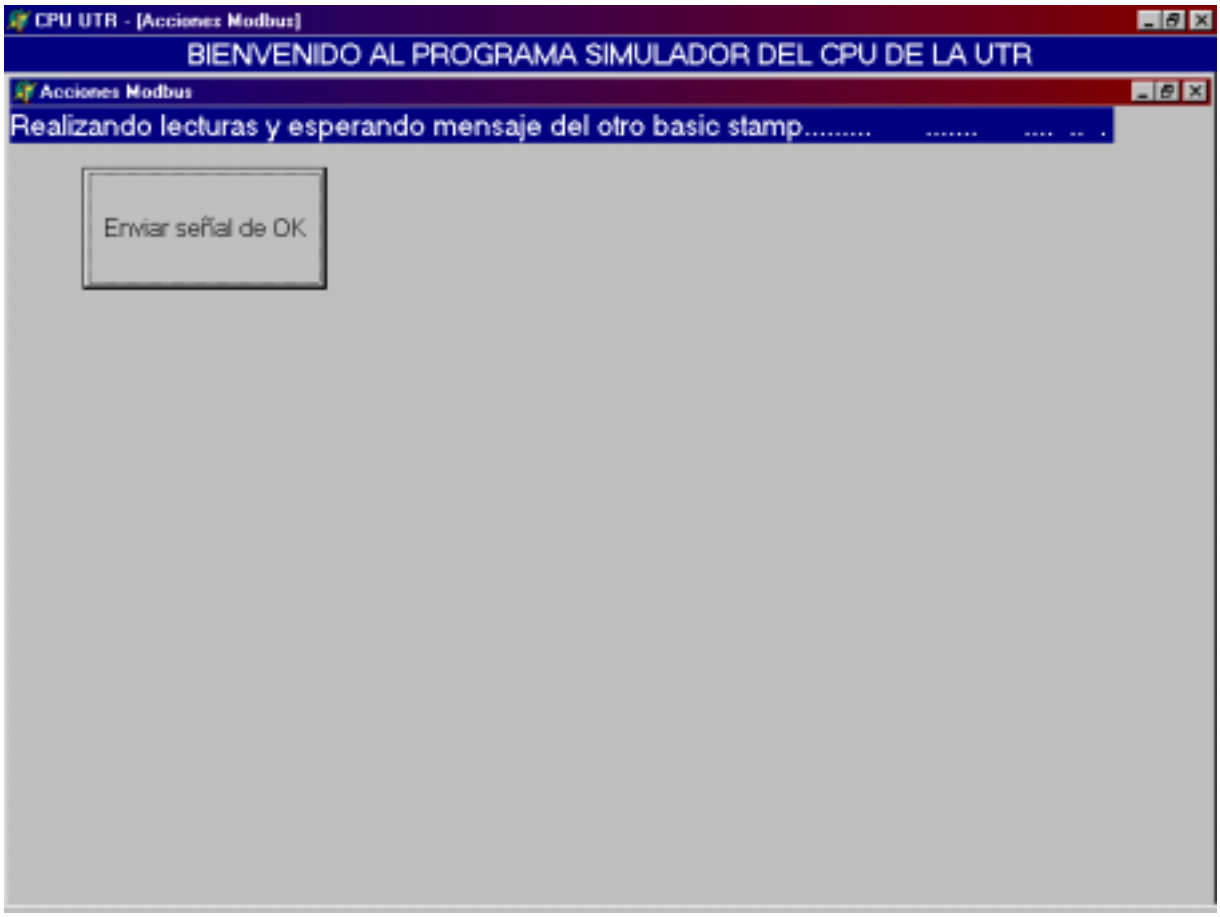

Figura 6.4 Pantalla del programa de aplicación simulador de la CPU de la unidad terminal remota.

6.2 Alcances y limitaciones

A continuación se exponen los alcances logrados y las limitaciones encontradas en la realización de las distintas etapas del proyecto.

Con respecto al hardware del sistema, los alcances obtenidos fueron el dispositivo traductor implementado en tablas para prototipo, con capacidad de comunicación mediante puertos seriales a una velocidad de 9600 baudios. Una limitación que se encontró en la realización del hardware del sistema fue la memoria RAM interna del microcontrolador. Las funciones de lectura que el protocolo Modbus establece, permiten que se lean varias señales con una sola instrucción. Sin embargo, si se solicita al dispositivo traductor que realice varias lecturas seguidas, los datos de cada lectura deben almacenarse en la memoria RAM del microcontrolador antes de enviarlo por el puerto serial hacia la computadora central. El microcontrolador utilizado, el Basic Stamp 2sx, tiene 32 bytes de memoria RAM, de los cuales únicamente 26 bytes se pueden utilizar como variables. Además, tiene 64 bytes adicionales de memoria RAM que no se pueden utilizar como variables. El resultado de esta limitación es que en las funciones de lectura, el dispositivo traductor puede enviar como máximo 504 señales digitales o 31 registros. Sin embargo, la limitación encontrada no es crítica, ya que permite la operación del sistema. Sin embargo, este podría ser mejorado en un futuro.

Al desarrollar el software del dispositivo traductor, los alcances obtenidos fueron la implementación de las funciones de protocolo Modbus definidas al iniciar el presente proyecto. Las limitaciones encontradas fueron principalmente por la necesidad de una memoria RAM más grande en el microcontrolador. Para el recibimiento de datos en el microcontrolador, se utilizó la instrucción SERIN. Sin embargo, dicha instrucción requiere de la utilización de variables para recibir datos. Esto provoca una limitación en la implementación de las instrucciones de escrituras múltiples del protocolo Modbus. Al poderse manejar únicamente 26 variables, entonces el número máximo de señales a escribir en una sola instrucción es de 112 señales digitales o 7 señales analógicas.

Una limitación que tuvo que tomarse en cuenta en el desarrollo de los programas de aplicación encargados de simular a la computadora central del sistema SCADA y a la CPU de la unidad terminal remota, fue la dificultad de sincronizar las comunicaciones seriales de las computadoras con el dispositivo traductor, ya que el microcontrolador Basic Stamp 2sx tiene problemas de este tipo al trabajar a velocidades iguales o superiores a los 9600 baudios. Para solucionar este problema, los puertos seriales de las computadoras fueron inicializados para transmitir datos con 2 bits de parada[9].

[9] Bits de parada: Señal binaria correspondiente a un uno lógico, que se ubica al final de cada dato transmitido vía puerto serial. Indica la finalización del envío de un dato.

CAPÍTULO 7

CONCLUSIONES Y RECOMENDACIONES

7.1 Conclusiones

Una vez concluido el presente proyecto, se llegó a las siguientes conclusiones:

a. El protocolo Modbus es un sistema que permite organizar las comunicaciones entre unidades terminales remotas y la computadora central de un sistema de control y adquisición de datos.

b. El proyecto realizado se comunica mediante dos puertos seriales, utilizando el microcontrolador Basic Stamp 2sx.

c. El lenguaje de programación Delphi permite el desarrollo de programas que utilicen el puerto serial de una computadora para comunicaciones externas.

d. El dispositivo traductor implementado permite la comunicación entre la computadora central de un sistema SCADA y sus unidades terminales remotas, mediante la utilización del protocolo Modbus.

7.2 Recomendaciones

Para futuras ampliaciones o consultas al proyecto desarrollado, se hacen las siguientes recomendaciones:

a. Para aumentar la capacidad del dispositivo traductor a la hora de realizar funciones de lectura o escritura de múltiples señales, se recomienda utilizar un microcontrolador con mas memoria RAM disponible con variables. De esta forma se lograría que el dispositivo pudiera leer o escribir una gran cantidad de señales con una sola instrucción de protocolo Modbus.

b. El microcontrolador utilizado tiene una velocidad promedio de operación de 10000 instrucciones por segundo. Para realizar comunicaciones a 9600 baudios puede presentar problemas de sincronización. Por esto se recomienda utilizar un microcontrolador más rápido, pero con la misma facilidad de poder realizar comunicaciones seriales.

c. Para mejorar los problemas mencionados anteriormente, se recomienda la utilización del microcontrolador Basic Stamp 2p. Es del mismo fabricante (Parallax Inc.) y viene en un empaque idéntico al microcontrolador utilizado, por lo que el hardware diseñado no sufriría ningún cambio. La diferencia se encuentra en el hecho de que el Basic Stamp 2p posee 64 bytes adicionales de memoria RAM. Además, su velocidad promedio de operación es de 20000 instrucciones por segundo, lo que representa el doble de la rapidez del Basic Stamp 2sx.

CAPÍTULO 8

BIBLIOGRAFÍA

A continuación se muestra la bibliografía y páginas de Internet que fueron consultadas en la realización del presente proyecto.

a. http://www.modicon.com

b. **Basic Stamp Manual**. Versión 2.0. Parallax Inc. 2000

c. Reisdorph, Kent. **Aprendiendo Borland Delphi 4 en 21 días**. Prentice Hall. México. 1999.

d. Tomasi, Wayne. **Sistemas de Comunicaciones Electrónicas**. Segunda edición. Prentice Hall. México. 1998.

APÉNDICES

Apéndice 1: Sistema SCADA

Un sistema SCADA es básicamente un sistema de monitoreo y control de Unidades Terminales Remotas (UTR), por medio de una computadora central

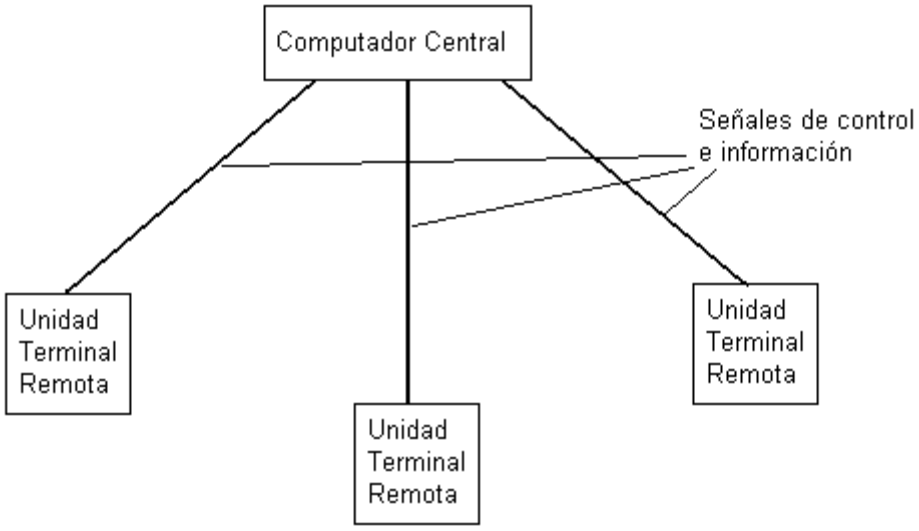

Figura A1.1 Diagrama de bloques que presenta el funcionamiento de un sistema SCADA.

La Figura A1.1 muestra un diagrama básico de un sistema SCADA, en el que una computadora central envía señales de control y recibe señales de información desde las Unidades Terminales Remotas.

Cuando las Unidades Terminales Remotas se encuentran físicamente muy alejadas del Computador Central, es necesario utilizar métodos inalámbricos de comunicación, para recibir instrucciones y enviar información, tal y como se muestra en la figura A1.2.

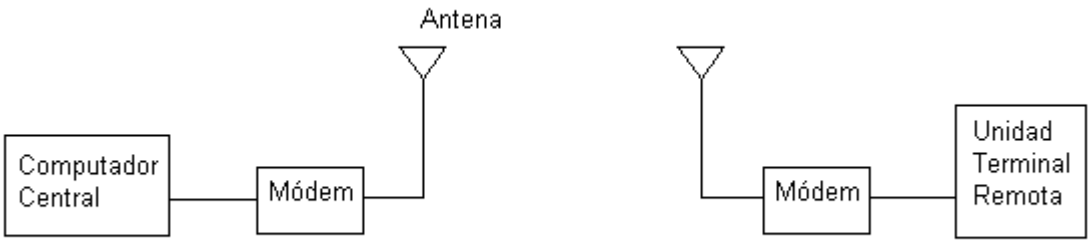

Figura A1.2 Diagrama de bloques que presenta la comunicación entre el Computador Central y una UTR físicamente muy alejada.

Las Unidades Terminales Remotas consisten en dispositivos que funcionan como interfaz entre los procesos físicos que se requieren monitorear y el Computador Central. Normalmente, las UTR pueden procesar entradas y salidas, analógicas ó digitales, con algunas diferencias de interfaz, las cuales dependen del diseño de la UTR. Tienen capacidad para recibir y ejecutar una serie de instrucciones, y para enviar información al Computador Central. La figura A1.3 muestra un ejemplo básico de aplicación de una unidad terminal remota.

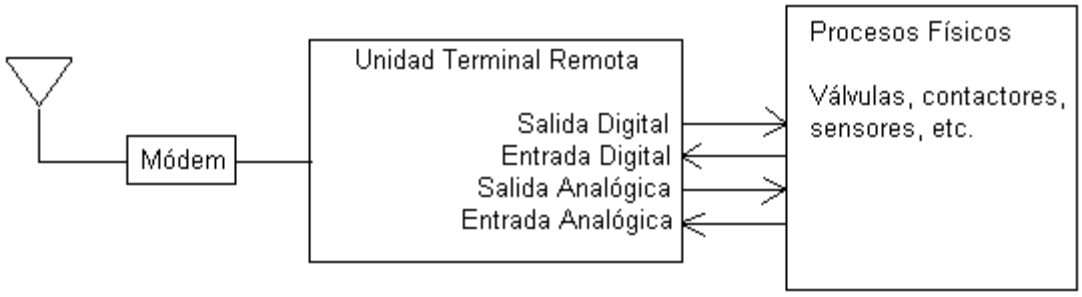

Figura A1.3 Diagrama de bloques que presenta un ejemplo del modo de aplicación de una unidad terminal remota.

En la figura A1.3 los procesos físicos que se pueden controlar mediante la UTR son muy variados, ya que cualquier señal analógica o digital puede ser leída o accionada por esta unidad. La cantidad de entradas y salidas que posee una UTR dependen del diseño de la misma.

Varios fabricantes ofrecen Unidades Terminales Remotas, con diferentes posibilidades de lectura y control de señales.

Para comunicarse con las Unidades Terminales Remotas, el Computador Central puede utilizar diferentes códigos. Sin embargo, existen protocolos estandarizados que facilitan la utilización de equipos de distintos fabricantes. Un protocolo que es muy utilizado en la actualidad es el llamado Modbus, desarrollado inicialmente como protocolo de comunicación entre Controladores Lógicos Programables.

Apéndice 2: Protocolo Modbus

El protocolo Modbus define la estructura de los mensajes que serán enviados por la computadora central a las diferentes Unidades Terminales Remotas. Describe el proceso que un controlador debe utilizar para solicitar acceso a otros dispositivos, cómo debe responder a los mensajes de los otros dispositivos y cómo deben ser detectados y reportados los errores.

El protocolo Modbus utiliza una técnica de Maestro-Esclavo cuando el controlador requiere comunicarse con los otros dispositivos. Los dispositivos esclavos responden enviando la información requerida o realizando las acciones indicadas en el mensaje. El controlador maestro puede enviar mensajes a los dispositivos esclavos individualmente o puede enviar mensajes generales. Los esclavos responderán únicamente a los mensajes enviados individualmente.

El protocolo Modbus establece el formato para los mensajes del dispositivo maestro indicando la dirección del dispositivo esclavo, un código definiendo la acción requerida, la información que debe ser enviada y una señal de detección de errores. El esclavo responderá utilizando también este protocolo. Si ocurriera un error en el recibimiento del mensaje, o si el dispositivo esclavo no puede realizar la acción requerida, entonces éste construirá un mensaje de error y lo enviará al dispositivo maestro.

Al enviar un mensaje, el código indica al dispositivo direccionado cuál es la acción que debe realizar. Los bytes de información contienen los datos adicionales que pueda requerir el esclavo para realizar la función. Por ejemplo, la función con código 03 solicitará al esclavo que lea los registros de memoria y que responda con su contenido. Los bytes de información deben indicar al esclavo en cuál registro comenzar con la lectura y cuántos registros debe leer. El espacio asignado para detección de errores provee un método para el esclavo para valorar la integridad del contenido del mensaje.

Al responder un mensaje, el esclavo enviará los datos requeridos por el maestro. Si ocurriera un error, el código de función es modificado para indicar que la respuesta es una respuesta de error, y los bytes de información contendrán un código que describirá el error.

Existen dos modos de transmisión serial de los datos, en el protocolo Modbus. Estos son el modo ASCII y el modo UTR (unidad terminal remota). En el presente proyecto, el modo de transmisión que se utilizó fue el UTR.

Sistema de Codificación:

Se codifica en hexadecimal.

Dos caracteres hexadecimales cada 8 bits del mensaje.

Bits por mensaje:

1 Start bit

8 data bits, LSB se envía primero

1 bit para paridad par/impar. (No se envía si no se utiliza paridad)

1 stop bit (2 stop bits si no se utiliza el bit de paridad)

Detección de error:

CRC (Cyclical Redundancy Check)

Estructura de los mensajes

A la hora de enviar un mensaje, el protocolo Modbus especifica un punto de inicio o señal de inicio y un punto final. Esto permite a los dispositivos esclavos saber cuándo inicia y cuándo termina un mensaje, para así leer correctamente los datos enviados.

En el modo UTR, el inicio de un mensaje está indicado por una señal nula equivalente a 3,5 caracteres (los caracteres son hexadecimales, es decir, son de 4 bits cada uno). Luego, se transmite la dirección del dispositivo con el que se requiere comunicación. Una vez terminado el mensaje, el protocolo indicará el final del mismo enviando un intervalo de 3,5 caracteres. Luego de esto, un nuevo mensaje puede ser enviado. La tabla A2.1 ilustra la estructura de un mensaje Modbus, enviado mediante la utilización del modo UTR.

Tabla A2.1 Estructura de un mensaje Modbus, utilizando el modo UTR

Inicio	Dirección	Función	Datos	Chequeo de Error (CRC)	Finalización
T1-T2-T3-T4	8 Bits	8 bits	n * 8 bits	16 bits	T1-T2-T3-T4

Direccionamiento de los dispositivos esclavos

Para direccionar los diferentes dispositivos esclavos, en los mensajes enviados por el maestro se reservan dos caracteres (8 bits). La dirección cero está reservada para el envío de mensajes generales. Los demás estarán ubicados en las direcciones que van desde el 1 hasta el 247 decimal. El dispositivo maestro accesa algún otro poniendo la ubicación en el espacio de dirección del mensaje. El esclavo responde colocando su dirección en el espacio reservado para esto en el mensaje, para indicar así al maestro cuál dispositivo está respondiendo.

Instrucciones Modbus

El tercer espacio de un mensaje Modbus está reservado para la función o instrucción que se solicita al esclavo que realice. Las tablas A2.2 y A.2.3 muestran las instrucciones implementadas en el protocolo Modbus y brinda una pequeña explicación de su significado.

Tabla A2.2 Funciones del protocolo Modbus.

Número	Nombre	Explicación
1	Lectura de estado de señales digitales	Solicita al esclavo la lectura del estado de un número de señales digitales. La dirección de las mismas es referenciada a 0x (Es decir, direcciones desde 00000 hasta 0FFFF hexadecimal)
2	Lectura de estado de señales digitales2	Solicita al esclavo la lectura del estado de un número de señales digitales. La dirección de las mismas es referenciada a 1x
3	Lectura de estado de señales analógicas	Solicita al esclavo la lectura del estado de un número de señales analógicas. La dirección de las mismas es referenciada a 4x
4	Lectura de estado de señales analógicas2	Solicita al esclavo la lectura del estado de un número de señales analógicas. La dirección de las mismas es referenciada a 3x
5	Escritura de una señal digital	Indica al esclavo el valor que debe colocar en determinada señal digital
6	Escritura de una señal analógica	Indica al esclavo el valor que debe colocar en determinada señal analógica
7	Lee estado de las señales de excepción	Lee estado de registro de estado (banderas del controlador)
8	Diagnóstico	Una serie de subfunciones de diagnóstico
9	Programa controlador 484	Utilizada para programar los controladores de este tipo
10	Poll 484	Utilizada para programar los controladores de este tipo
11	Obtención del contador de eventos de comunicación	Obtiene el dato del contador de eventos de comunicación del dispositivo esclavo
12	Obtención de registro de eventos de comunicación	Obtiene una serie de datos que registran el estado de las últimas comunicaciones establecidas.
13	Programación del controlador	Utilizada en la programación del dispositivo esclavo

Tabla A2.3 Funciones del protocolo Modbus (continuación).

Número	Nombre	Explicación
14	Poll controller	Utilizada en la programación del dispositivo esclavo
15	Escritura de varias señales digitales	Indica al esclavo los valores que debe colocar en diferentes señales digitales
16	Escritura de varias señales analógicas	Indica al esclavo los valores que debe colocar en diferentes señales analógicas
17	Reporte de identificación del esclavo	Regresa información específica al dispositivo esclavo
18	Programación 884/M84	Utilizada en la programación de dicho controlador
19	Reinicio de enlace de comunicación	Reinicia el enlace de comunicación
20	Lectura de referencia general	Lectura de los registros reservados para referencia general
21	Escritura de referencia general	Escritura de los registros reservados para referencia general
22	Escritura enmascarable de registros referenciados a 4x	Escribe los registros especificados, referenciados a 4x, mediante una combinación de las funciones lógicas AND y OR.
23	Lectura/Escritura de registros referenciados a 4x	Lee en registros indicados, y escribe en otros registros indicados.
24	Lectura de cola FIFO	Lee cola de registros referenciados a 4x

Mensajes de error

Cuando se presenta un error en la ejecución de una instrucción, el dispositivo esclavo debe construir una respuesta de error y enviarla al maestro.

El mensaje de error está conformado, como puede observarse en la tabla A2.4, por la dirección del dispositivo esclavo, seguida por la señal de identificación de error. Luego se coloca un espacio de descripción del tipo de error que se sucedió. Finalmente, se colocan los caracteres de chequeo de error.

La señal de identificación de error es la señal de la instrucción requerida, con el bit más significativo colocado en uno. Esto equivale a realizar la operación lógica "Y" entre la señal de instrucción y el número 128 decimal.

La tabla A2.5 proporcional los distintos valores que deben colocarse en la señal de descripción de error, dependiendo del evento ocurrido.

Tabla A2.4 Estructura de los mensajes de error en protocolo Modbus.

Dato 1	Dato 2	Dato 3	Dato 4
Dirección del dispositivo esclavo	Identificación de error	Descripción del tipo de error ocurrido	Chequeo de redundancia cíclica

Tabla A2.5 Valores de señal de descripción de error, dependiendo del tipo de evento ocurrido.

Valor de señal de descripción de error	Nombre	Significado
1	Función inválida	La función requerida no está disponible en el dispositivo esclavo
2	Dirección inválida	La dirección enviada no está disponible
3	Dato inválido	El dato recibido no es válido en el esclavo
4	Error de esclavo	Ocurrió un error irrecuperable en el esclavo
5	Reconocimiento	Se está procesando la función, pero el esclavo tardará mucho. Este mensaje se envía para prevenir un error de tiempo del maestro.
6	Esclavo ocupado	El esclavo está ocupado. El maestro deberá retransmitir los datos posteriormente.

Apéndice 3: Programación del microcontrolador Basic Stamp 2sx

En las tablas A3.1 y A3.2 se muestran las diferentes instrucciones de programación del microcontrolador Basic Stamp 2sx.

Tabla A3.1 Instrucciones de programación del Basic Stamp 2sx.

Instrucción	Funcionamiento
Branch	Realiza un salto dependiendo del valor de una variable definida
Button	Se utiliza para conectar un puerto a un pushbutton.
Count	Sirve para contar ciclos de cambio de estado en un puerto
Data	Escribe datos en al EEPROM del microcontrolador
Debug	Se utiliza como interfaz de pruebas con una computadora, a la hora de programar
Dtmfout	Genera tonos de teléfono (DTMF)
End	.Detiene la ejecución del programa del microcontrolador, y entra modo de bajo consumo.
For - Next - Step	Ciclo de repetición
Freqout	Genera una o dos ondas senoidales de salida
Get	Lee espacio de memoria RAM alterna
Gosub - Return	Realiza saltos y retornos de subrutina
Goto	Salto incondicional
High	Coloca en uno lógico algún puerto determinado
If - Then	Verifica condiciones de variables
Input	Hace que un puerto sea entrada
Lookdown	Se utiliza para comparar una variable, con valores establecidos en una lista
Lookup	Coloca en una variable un valor tomado de una lista, dependiendo de una variable índice.
Low	Pone en cero lógico un pin de salida

Tabla A3.2 Instrucciones de programación del Basic Stamp 2sx (continuación).

Instrucción	Funcionamiento
Nap	Hace que el microcontrolador entre en modo de bajo consumo por un período determinado
Output	Convierte en salida a un puerto determinado
Pause	Pausa la ejecución del programa un período determinado
Pulsin	Mide el ancho de un pulso de entrada
Pulsout	Genera un pulso con características determinadas
Put	Coloca datos en la memoria RAM alterna
Pwm	Genera salidas analógicas mediante modulación de ancho de pulso
Random	Genera un número pseudo-aleatorio
Rctime	Sirve para medir tiempos reducidos, como descargas de circuitos RC
Read	Lee información de la EEPROM del microcontrolador
Reverse	Cambia la dirección de un puerto (entrada - salida)
Run	Inicia un programa en las distintas tarjetas de memoria EEPROM
Serin	Utiliza un puerto como entrada de comunicaciones seriales asíncronas
Serout	Utiliza un puerto como salida de comunicaciones seriales asíncronas
Shiftin	Utiliza un puerto como entrada de comunicaciones seriales sincrónicas
Shiftout	Utiliza un puerto como entrada de comunicaciones seriales sincrónicas
Sleep	Apaga al microcontrolador un tiempo prolongado
Stop	Detiene la ejecución de un programa
Toggle	Cambia el estado de un puerto de salida
Write	Escribe en la memoria EEPROM
Xout	Se utiliza para enviar señales a través de la línea de alimentación

Apéndice 4: Lista de componentes electrónicos que fueron requeridos para la realización del proyecto

Tabla A4.1 Lista de componentes requeridos para el proyecto.

Componente	Cantidad
Microcontrolador Basic Stamp 2sx	1
Decodificador 74LS42	1
Latch 74LS573	2
Buffer 74LS541	8
Dip switch de 8 interruptores	5
TC232CPE	1
Inversores 7404	1
Interruptor tipo Push Button	1
Resistencias 10KΩ	46
Resistencias 220Ω	10
LEDS	10
Memoria SRAM MS6264	1
Protoboard	5