Universidad de las Ciencias Informáticas

Facultad #5

Módulo de comunicación de un firmware para un controlador lógico programable basado en hardware reconfigurable.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor:

Yarisnelis Hanoy Dorta Martínez.

Tutor:

Ing. Maikel Ramírez Despaigne.

Ciudad de la Habana, Junio del 2011.

"Año 52 de la Revolución".

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Yarisnelis H. Dorta Martínez

_____

Firma del Autor

Ing. Maikel Ramírez Despaigne

_____

Firma del Tutor

AGRADECIMIENTOS

- A la Revolución Cubana y a mi Comandante en Jefe Fidel Castro por permitir hacer realidad mi sueño.
- A la Universidad de las Ciencias Informáticas, por ser mi hogar y mi mundo durante cinco inolvidables años, donde nos formamos como hombres y mujeres del presente y del futuro.
- A mi tutor Maikel Ramírez Despaigne por la dedicación brindada para que este trabajo saliera adelante.
- A mi madre Dania Martínez por todo el sacrificio que ha hecho para que yo esté en esta universidad, y por haber hecho de mí la persona que soy. Así mismo hago extensivo el agradecimiento a mi hermano Yosmelis Dorta y en especial a mi padrastro Oscar Romero por quererme y tratarme como una hija, por cuidar y apoyar a mi mamá todos estos años.
- A mi padre Jesús Dorta y al resto de mi familia, por su apoyo incondicional.
- A Daylin Hernández (Pucca) por todas las cosas lindas que ha hecho por mí, sobre todo cuando estuve en Venezuela, lejos de mi familia, gracias Pucca por ser la hermana mayor que la vida me negó y estar siempre ahí cuando más te he necesitado.
- A Sindy Concepción (La polla), por ser mi amiga incondicional, y mi compañera de fiestas durante muchos años. Igualmente agradezco a Oraida y a Sayli por acogerme en su casa como un miembro más de la familia.
- A mi hermanita Lourdes Carrazana por soportar mis pesadeces durante estos años y apoyarme incondicionalmente, aunque no lo crea yo también la quiero mucho y solo espero que nuestra amistad dure para siempre.
- A mi mamita Ania Mestre por quererme, apoyarme y considerarme como una hija, le doy gracias a Dios por ponerte en mi camino y solo espero que te sientas orgullosa de mí.
- A las secretarias más lindas y buenas de la Universidad, Mileydis Oliva y Leyvis Teruel por estar ahí cuando más las necesitaba.
- A mi decana Mayra Durán por ser como una madre para todos en la facultad.

AGRADECIMIENTOS

- A mis compañeros de aula y en especial a mis compañeras de apartamento Danae Cámara, Sulemy Cobas, Eileen Montero, Daylen Elena, Daylen Espinosa, Lianet Pozo, Annia María y Orlys Valero.
- A Yarisleydis Barzaga y Yanisleydis Enríquez por ser tan especiales conmigo, siempre las recordaré, a la china por sus locuras y los buenos momentos que pasamos juntas y a la gordi por todos sus consejos para que fuera una mejor persona.
- A Ronni Rivas por darme su amor, apoyo y comprensión durante los dos años que estuvimos juntos, gracias por cada minuto que pasamos juntos, nunca los olvidaré.
- A Alain por todo el apoyo brindado durante estos 5 años.
- A mis amigos Roseli Lemes, Yanet Ramos, Mailen Berenger, Yordan Gallardo, Sandra Rangel, Adriel Ramos, Beatriz, Rosaura y Frank Rodríguez, por todo su apoyo.
- A Rubén Gómez Johnson, Adrián Carlos, Javier Lorie y especialmente a Antonio Cedeño (toni) por ayudarme cada vez que tenía una duda de la tesis.
- A todas aquellas personas que en la UCI, me brindaron su amistad y ayuda siempre que la necesité.
- También a todos aquellos que fueron siempre fuente de fuerza y alegría y me impulsaron a seguir adelante hasta alcanzar la meta.
- A todos los que alguna vez me preguntaron: ¿y la tesis como va?
  A todos, muchas gracias**…**

Yarisnelis Hanoy Dorta Martínez.

DEDICATORIA

Especialmente a mi madre por todo su apoyo y su cariño durante estos largos años, por depositar toda su confianza en mí y permitirme realizar mi sueño, gracias mami por sacrificarte estos 22 años, sólo le pido a la vida que me deje disfrutarte muchos años más y me permita compensarte, siendo la hija que tú te mereces, te quiero muchísimo.

A mi padrastro por acompañar a mi mamá todos estos años, quererla y apoyarla incondicionalmente, sin importar lo que pase en el futuro quiero que sepas que eres como un padre para mí.

A mi hermano por apoyarme y cuidar de mi mami en estos años en los que he estado lejos.

A mi padre, familiares y amigos por su apoyo incondicional.

A Ania por ser como una madre para mí todos estos años, gracias mamita por todas las cosas que hiciste por mí, por cada consejo que me diste, solo espero poder compensarte, esta tesis también es para ti.

RESUMEN

En la presente tesis se desarrolló el módulo de comunicación para el firmware de un controlador lógico programable basado en hardware reconfigurable con el procesador softcore embebido MicroBlaze, reutilizando módulos IP para hardware reconfigurable, y con las funcionalidades necesarias en el control industrial actual.

El módulo se implementó en el kit de desarrollo spartan 3e, utilizando las herramientas ISE 12.1 y EDK 12.1 de la empresa Xilinx. Se utilizó como lenguaje de modelado UML y sigue los pasos que propone el Proceso Unificado de Rational (RUP), además se utilizó como herramienta CASE para generar los artefactos que dicha metodología propone Visual Paradigm. El lenguaje C se utilizó para el desarrollo del sistema.

Palabras Clave: controlador lógico programable, módulo de comunicación, firmware, hardware reconfigurable.

Índice de figuras

Índice de tablas

Introducción

Los controladores lógicos programables son componentes indispensables en los sistemas de automatización: en la industria, automatización del sector hotelero, etc.

En Cuba se han desarrollado algunos autómatas programables como el *Nova* creado por el ICID (Instituto Central de Investigación Digital) y el *ErosPLC* del grupo de Serconi, ambos basados en microcontroladores.

Existe en el mundo un auge creciente del uso del hardware reconfigurable en el control industrial, sistemas operativos embebidos, potentes herramientas de diseño asistido por computadora (CAD), para el diseño del flujo hardware y software con dispositivos lógicos programables, específicamente las matrices de compuertas programables (FPGAs), módulos de propiedad intelectual (o módulos IP) como: puertos serie, puertos ethernet, etc.

Se está desarrollando un controlador lógico programable (PLC) reconfigurable utilizando los campos de matrices programables (FPGA), con un procesador softcore embebido, reutilizando módulos IP para hardware reconfigurable, y con las funcionalidades necesarias en el control industrial actual. Un elemento fundamental en un PLC es el firmware que se encarga de manejar el funcionamiento interno del PLC y de interpretar un programa en lenguaje PLC. Un firmware está compuesto por varios módulos, uno de ellos es el módulo de comunicación el cual debe ser desarrollado en esta investigación, teniendo en cuenta que no existe dicho módulo y la importancia del mismo, se tiene como **situación problémica:**

- Dificultad para controlar un proceso entre varios PLCs.
- No transferencia de archivos.
- No se podrá reportar alarmas.
- No será posible el diagnóstico de forma remota.

Por lo anteriormente expuesto, se deriva el siguiente **problema científico**: ¿Cómo proveer a un PLC basado en hardware reconfigurable de un mecanismo de comunicación?

Para ello el **objeto de estudio** de esta investigación es: los firmwares para los controladores lógicos programables y el **campo de acción** es el módulo de comunicación de un firmware para el controlador lógico programable basado en hardware reconfigurable.

En conformidad con el problema, nuestro **objetivo** es:

Desarrollar el módulo de comunicación de un firmware para un controlador lógico programable basado en hardware reconfigurable.

Objetivos específicos.

I. Caracterizar el firmware del controlador lógico programable-reconfigurable.

- Tarea1: Estudio de la arquitectura hardware y funcionalidades de los controladores lógicos programables actuales.
- Tarea2: Revisión bibliográfica de los firmwares de los controladores lógicos programables actuales.
- Tarea 3: Procesamiento y evaluación de la información obtenida.
- Tarea4: Especificación de las funcionalidades del firmware.

II. Determinar una arquitectura general que responda a las funcionalidades que se determinaron.

- Tarea 1: Diseño teórico de la arquitectura de software del módulo de comunicación para el controlador lógico programable basado en hardware reconfigurable.
- Tarea 2: Selección de las herramientas para el desarrollo del módulo de comunicación del controlador lógico programable basado en hardware reconfigurable.

III. Analizar las características del sistema y realizar el diseño del mismo.

- Tarea 1: Construcción del Diagrama de Dominio.
- Tarea 2: Levantamiento de los requisitos del sistema.
- Tarea 3: Realización del Modelo de Casos de Uso del Sistema.

- Tarea 4: Construcción del Diagrama de Clases del Diseño.

IV. Implementar el módulo de comunicación del firmware para el controlador lógico programable basado en hardware reconfigurable.

- Tarea 1: Familiarización y dominio de las herramientas seleccionadas.
- Tarea 2: Implementación del módulo de comunicación para protocolos modbus.
- Tarea 3: Realización de pruebas a la aplicación.

Idea a defender:
Con el desarrollo del módulo de comunicación se podrá establecer la comunicación serie entre el PLC y cualquier dispositivo que implemente Modbus.

Se utilizaron los siguientes métodos:

Teóricos:

***Histórico lógico*:** este método es utilizado para analizar los conocimientos teóricos existentes del protocolo Modbus, específicamente en su variante RTU, su situación en el mercado y tendencias actuales.

Inductivo Deductivo: este método se utiliza al estudiar toda la información encontrada y decidir las metodologías que se utilizarán para el desarrollo del sistema.

Empíricos

Consulta a especialistas: Se empleó para comprobar la necesidad y la funcionalidad práctica del módulo de comunicación que se propone en la presente investigación.

Los *aportes esperados* de esta investigación son:

Aporte Teórico: Diseño de una arquitectura software para desarrollar el módulo de comunicación de un firmware para un controlador lógico programable basado en hardware reconfigurable.

***Aporte Práctico*:** Implementación del módulo de comunicación de un firmware para un controlador lógico programable basado en hardware reconfigurable.

Capítulo 1: Fundamentación Teórica.

1.1 Introducción.

En este capítulo se abordan las características de los PLCs, su estructura interna, sus diversas funcionalidades en el control industrial y se hace mención a los diferentes tipos de PLCs que existen actualmente. También se realiza un breve acercamiento a lo que es un firmware y el papel que juega este dentro del PLC que se está desarrollando. Además se definen las interfaces de comunicación serie y TCP / IP, y se estudian los aspectos principales que caracterizan al protocolo Modbus, tipos de datos, formato de la trama, entre otros.

1.2 Controlador Lógico Programable.

Un controlador lógico programable (PLC) es una forma especial de controlador basado en procesador que usa una memoria programable para almacenar instrucciones e implementar funciones, tales como: lógica, secuenciamiento, temporizaciones, conteo y aritmética, para el control de máquinas y procesos, y son diseñados para operar por ingenieros con quizás un conocimiento limitado de computadores y lenguajes de computación (2).

Los PLCs son optimizados para tareas de control y el entorno industrial (2). Por tanto, son:

1. Robustos y diseñados para resistir vibraciones, temperatura, humedad y ruido.
2. Tienen la interfaces para las entradas y las salidas dentro del controlador.
3. Son fáciles de programar y tienen un lenguaje de programación fácil de comprender el cual es principalmente concerniente con operaciones lógicas y de conmutación.

Los PLC´s de manera general se componen esencialmente de tres bloques, los cuales son (4):

- La Sección de Entradas.
- La Unidad Central de Procesos (CPU).

- La Sección de Salidas.

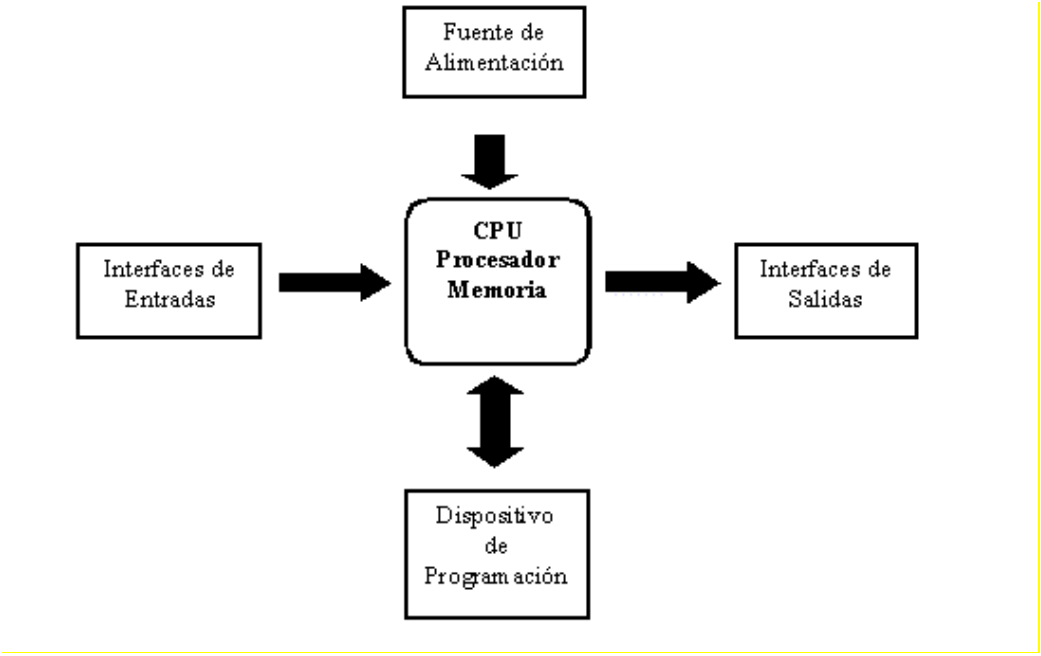

Figura 1 Componentes principales de un PLC.

La unidad central de proceso es, por decirlo así, la inteligencia del sistema, ya que mediante la interpretación de las instrucciones del programa de usuario y en función de los valores de las entradas, activa las salidas deseadas.

La sección de entradas, mediante la interfaz, adapta y codifica de forma comprensible por la CPU las señales procedentes de los dispositivos de entrada o captadores, como pulsadores, finales de carrera, sensores, etc.

La sección de salidas, mediante la interfaz, trabaja de forma inversa a la sección de entradas, es decir, decodifica las señales procedentes de la CPU, las amplifica y las envía a los dispositivos de salida o actuadores, como lámparas, relés, contactares, arrancadores, electroválvulas, etc.

Generalmente vamos a disponer de dos tipos de E/S:

- Digital: Estas se basan en el principio de todo o nada, es decir o no conducen señal alguna o poseen un nivel mínimo de tensión. Estas E/S se manejan a nivel de bit dentro del programa de usuario.
- Analógica: Estas pueden poseer cualquier valor dentro de un rango determinado especificado por el fabricante. Estas señales se manejan a nivel de byte o palabra (8/16 bits) dentro del programa de usuario.

Con las partes descritas podemos decir que tenemos un PLC, pero para que sea operativo son necesarios otros elementos tales como:

• La unidad de alimentación.

• La unidad o consola de programación si no se programa desde la PC.

El mercado de los PLC puede segmentarse en 5 grupos (2):

1. Micro PLCs.
2. PLCs pequeños (Small).
3. PLCs medianos (Medium).
4. PLCs grandes (Large).
5. PLCs muy grandes (Very Large).

Cada uno de estos con características diferentes, y su funcionalidad es mayor de una generación a otra, por lo que su uso siempre va a depender del problema que se quiera resolver.

1.2.1. Ciclo scan del PLC.

Antes de la ejecución del programa de usuario, la CPU consulta los estados de las entradas físicas y carga con ellos la memoria imagen de entradas. Durante la ejecución del programa de usuario, la CPU realiza los cálculos a partir de los datos de la memoria imagen y del estado de los temporizadores, contadores y relés internos. El resultado de estos cálculos queda depositado en la memoria imagen de salidas. Finalizada la ejecución, la CPU transfiere a las interfaces de salida los estados de las señales contenidos en la memoria imagen de salidas, quedando el sistema preparado para comenzar un nuevo ciclo.

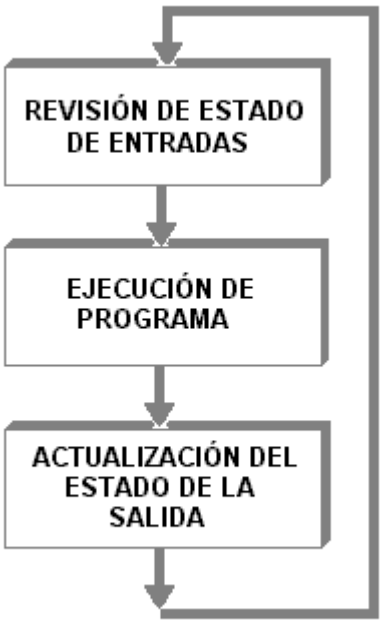

Figura 2 Ciclo Scan del PLC.

1.3 Hardware reconfigurable en el control industrial.

Los PLCs son importantes dispositivos de control industrial basados en tecnología de microprocesador. Características del PLC, tales como, alta generalidad y flexibilidad, fácil de programar y usar lo hacen ampliamente utilizado en la industria, incluyendo el control de automatización industrial (2).

Las técnicas de manufacturas han evolucionado con el tiempo, desde la producción masiva de ayer a la automatización flexible de hoy y a la automatización reconfigurable de mañana. Los ingenieros de control están incrementando el uso de tecnologías embebidas, tales como FPGA en combinación con las PCs y PLCs para solucionar aplicaciones de control complejas que mejoran los tiempos de respuestas del sistema y proporcionan confiabilidad adicional y determinismo (2). Los FPGAs son chips de silicio reprogramables, al utilizar bloques de lógica pre-construidos y recursos para ruteo programables, se pueden configurar para implementar funcionalidades personalizadas en hardware sin tener que utilizar una tablilla de prototipos o un cautín. Sólo se deberán desarrollar tareas de cómputo digital en software y compilarlas en un archivo de configuración o *bitstream* que contenga información de cómo deben conectarse los componentes (22).

El FPGA es una tecnología que emerge rápido y proporciona a los ingenieros de control ultra rápida velocidad de control, implementación de contadores personalizados, emulación de protocolos digitales, control discreto, y señales de E/S analógicas y digitales mezcladas en un solo dispositivo (2).

El crecimiento de esquemas complejos de control obliga a ingenieros e investigadores a explorar nuevas arquitecturas. El hardware reconfigurable, específicamente los FPGAs, han surgido como una alternativa a la demanda de aplicaciones (2).

1.4. Control Lógico Programable en hardware reconfigurable.

Por ser el FPGA de naturaleza puramente digital, la mayor parte de las funcionalidades del PLC van estar embebidas en él, excepto la interfaces para E/S de las señales analógicas. En el PLC convencional las funcionalidades (conteo, temporización, módulo PID, etc) se desarrollan a nivel de software y de manera secuencial. En el PLC que se está desarrollando las funcionalidades se realizan en hardware aprovechando las potencialidades de paralelismo y velocidad que este ofrece (2).

Se quiere obtener un PLC reconfigurable que se pueda adaptar a los requerimientos de cada aplicación en particular (cada aplicación puede diferir en la cantidad y tipo de módulos de comunicación, módulo PID, contadores, temporizadores, E/S analógicas y digitales).

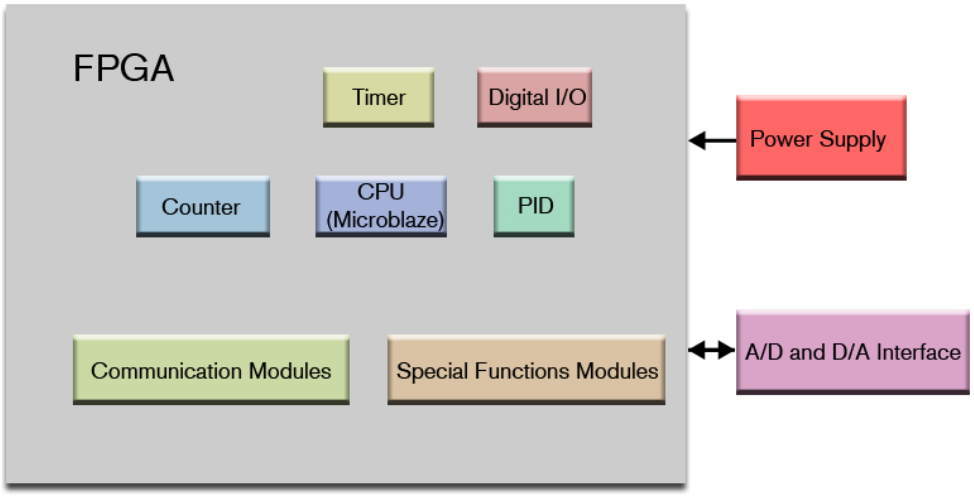

Figura 3 PLC basado en hardware reconfigurable.

1.5. Introducción al Firmware.

Un firmware o programación en firme, es un bloque de instrucciones de programa para propósitos específicos, grabado en una memoria del tipo Read Only Memory (ROM), que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo. Funcionalmente, el firmware es el intermediario (interfaz) entre las órdenes externas que recibe el dispositivo y su electrónica, ya que es el encargado de controlar a esta última para ejecutar correctamente dichas órdenes. Por lo general, la finalidad del firmware es la de control de las operaciones: recibe instrucciones y las redirecciona según las necesidades (11). El firmware viene incorporado con el dispositivo, por lo que es en cierto punto, hardware y software al mismo tiempo.

Encontramos firmware en memorias ROM de los sistemas de diversos dispositivos periféricos, como en monitores de video, unidades de disco, impresoras, etc., pero también en los propios microprocesadores, chips de memoria principal y en general en cualquier circuito integrado.

El programa BIOS de un ordenador es un firmware cuyo propósito es activar una máquina desde su encendido y preparar el entorno para la instalación de un Sistema Operativo complejo, así como responder a otros eventos externos (botones de pulsación humana) y al intercambio de órdenes entre distintos componentes del ordenador.

En un microprocesador el firmware es el que recibe las instrucciones de los programas y las ejecuta en la compleja circuitería del mismo, emitiendo órdenes a otros dispositivos del sistema.

En los PLC tradicionales, el firmware se almacena en memoria ROM y es el responsable del funcionamiento interno del mismo, además es el encargado de interpretar el programa de usuario realizado en un lenguaje de programación de PLC. Parte de ese firmware lo compone el código del módulo de comunicación utilizando el protocolo Modbus en su variante serie RTU que es el objetivo de esta tesis.

1.6. Interfaces de comunicación. Serial y TCP/ IP.

Hoy en día las computadoras han avanzado bastante desde que se inventó la primera, y con ellas han avanzado los dispositivos de almacenamiento. Debido al avance tecnológico se crearon puertos, que sirven para recibir y enviar datos de la computadora a periféricos que estén conectados a ella, estos se llaman puertos de comunicación y actualmente se conoce una gran gama de ellos. Los puertos de comunicación, como su nombre indica, son una serie de puertos que sirven para comunicar nuestro ordenador con periféricos u otros ordenadores. Se trata en definitiva de dispositivos **I/O** (Input/Output, o Entrada/Salida). Entre los diferentes puertos de comunicación tenemos (9): puerto serie y puerto TCP/ IP.

1.6.1. Puerto Serie.

Un puerto serie o puerto serial es una interfaz de comunicaciones de datos digitales, frecuentemente utilizado por computadoras y periféricos, donde la información es transmitida bit a bit enviando un solo bit a la vez, en contraste con el puerto paralelo que envía varios bits simultáneamente. A lo largo de la mayor parte de la historia de las computadoras, la transferencia de datos a través de los puertos serie ha sido generalizada. Se ha usado y sigue usándose para conectar las computadoras a dispositivos como terminales o módems. Los mouse, teclados, y otros periféricos también se conectaban de esta forma. El término "puerto serie" normalmente identifica el hardware más o menos conforme al estándar RS-232, diseñado para interactuar con un módem o con un dispositivo de comunicación similar (8).

Los dispositivos de redes, como los enrutadores y conmutadores, a menudo tienen puertos serie para modificar su configuración. Los puertos serie se usan frecuentemente en estas áreas porque son sencillos, baratos y permiten la interoperabilidad entre dispositivos. La desventaja es que la configuración de las conexiones serie requiere, en la mayoría de los casos, un conocimiento avanzado por parte del usuario y el uso de comandos complejos si la implementación no es adecuada.

La comunicación serial se lleva a cabo asincrónicamente, es decir que no es necesaria una señal (o *reloj*) de sincronización: los datos pueden enviarse en intervalos aleatorios. A su vez, el periférico debe poder distinguir los caracteres (un carácter tiene 8 bits de longitud) entre la sucesión de bits que se está enviando. Ésta es la razón por la cual en este tipo de transmisión, cada carácter se encuentra precedido por un bit de arranque y seguido por un bit de parada. Estos bits de control, necesarios para la transmisión serial, desperdician un 20% del ancho de banda (cada 10 bits enviados, 8 se utilizan para cifrar el carácter y 2 para la recepción).

Tipos de comunicación en serie:

- Simplex

En este caso el emisor y el receptor están perfectamente definidos y la comunicación es unidireccional. Este tipo de comunicaciones se emplean, usualmente, en redes de radiodifusión, donde los receptores no necesitan enviar ningún tipo de dato al transmisor.

- Duplex, half duplex o semi-duplex

En este caso ambos extremos del sistema de comunicación cumplen funciones de transmisor y receptor y los datos se desplazan en ambos sentidos pero no de manera simultánea. Este tipo de comunicación se utiliza habitualmente en la interacción entre terminales y una computadora central.

- Full Duplex

El sistema es similar al duplex, pero los datos se desplazan en ambos sentidos simultáneamente. Para que sea posible ambos emisores poseen diferentes frecuencias de transmisión o dos caminos de comunicación separados, mientras que la comunicación semi-duplex necesita normalmente uno solo. Para el intercambio de datos entre computadores este tipo de comunicaciones es más eficiente que las transmisiones semi-duplex.

1.6.2. TCP/ IP.

TCP/IP son las siglas de Protocolo de Control de Transmisión/Protocolo de Internet (en inglés *Transmission Control Protocol/Internet Protocol*), un sistema de protocolos que hacen posibles servicios Telnet, FTP, E-mail, y otros, entre ordenadores que no pertenecen a la misma red (9).

El **Protocolo de Control de Transmisión (TCP)** permite a dos anfitriones establecer una conexión e intercambiar datos. El TCP garantiza la entrega de datos, es decir, que los datos no se pierdan durante la transmisión y también garantiza que los paquetes sean entregados en el mismo orden en el cual fueron enviados.

El **Protocolo de Internet (IP)** utiliza direcciones que son series de cuatro números octetos (byte) con un formato de punto decimal, por ejemplo: 69.5.163.59.

La arquitectura del TCP/IP consta de cinco niveles o capas en las que se agrupan los protocolos, y que se relacionan con los niveles OSI de la siguiente manera:

- Aplicación: Se corresponde con los niveles OSI de aplicación, presentación y sesión. Aquí se incluyen protocolos destinados a proporcionar servicios, tales como correo electrónico (SMTP), transferencia de ficheros (FTP), conexión remota (TELNET) y otros más recientes como el protocolo HTTP (*Hypertext Transfer Protocol*).
- Transporte: Coincide con el nivel de transporte del modelo OSI. Los protocolos de este nivel, tales como TCP y UDP, se encargan de manejar los datos y proporcionar la fiabilidad necesaria en el transporte de los mismos.

- Internet: Es el nivel de red del modelo OSI. Incluye al protocolo IP, que se encarga de enviar los paquetes de información a sus destinos correspondientes. Es utilizado con esta finalidad por los protocolos del nivel de transporte.
- Físico: Análogo al nivel físico del OSI.
- Red: Es la interfaz de la red real. TCP/IP no especifica ningún protocolo concreto.

TCP/IP necesita funcionar sobre algún tipo de red o de medio físico que proporcione sus propios protocolos para el nivel de enlace de Internet. Por este motivo hay que tener en cuenta que los protocolos utilizados en este nivel pueden ser muy diversos y no forman parte del conjunto TCP/IP. Sin embargo, esto no debe ser problemático puesto que una de las funciones y ventajas principales de TCP/IP es proporcionar una abstracción del medio, de forma que sea posible el intercambio de información entre medios diferentes y tecnologías que inicialmente son incompatibles.

Dentro de un sistema TCP/IP los datos transmitidos se dividen en pequeños paquetes, y resaltan una serie de características.

- La tarea de IP es llevar los datos a granel (los paquetes) de un sitio a otro. Las computadoras que encuentran las vías para llevar los datos de una red a otra (denominadas enrutadores) utilizan IP para trasladar los datos. En resumen IP mueve los paquetes de datos a granel, mientras TCP se encarga del flujo y asegura que los datos estén correctos.
- Las líneas de comunicación se pueden compartir entre varios usuarios. Cualquier tipo de paquete puede transmitirse al mismo tiempo, y se ordenará y combinará cuando llegue a su destino. Compare esto con la manera en que se transmite una conversación telefónica, una vez que establece una conexión, se reservan algunos circuitos para usted, que no puede emplear en otra llamada, aunque deje esperando a su interlocutor por veinte minutos.
- Los datos no tienen que enviarse directamente entre dos computadoras. Cada paquete pasa de computadora en computadora hasta llegar a su destino. Éste, claro está, es el

secreto de cómo se pueden enviar datos y mensajes entre dos computadoras aunque no estén conectadas directamente entre sí. Lo que realmente sorprende es que sólo se necesitan algunos segundos para enviar un archivo de buen tamaño de una máquina a otra, aunque estén separadas por miles de kilómetros y pese a que los datos tienen que pasar por múltiples computadoras. Una de las razones de la rapidez es que, cuando algo anda mal, sólo es necesario volver a transmitir un paquete, no todo el mensaje.

- Los paquetes no necesitan seguir la misma trayectoria. La red puede llevar cada paquete de un lugar a otro y usar la conexión más idónea que esté disponible en ese instante. No todos los paquetes de los mensajes tienen que viajar, necesariamente, por la misma ruta, ni necesariamente tienen que llegar todos al mismo tiempo.
- La flexibilidad del sistema lo hace muy confiable. Si un enlace se pierde, el sistema usa otro. Cuando usted envía un mensaje, el TCP divide los datos en paquetes, ordena éstos en secuencia, agrega cierta información para control de errores y después los lanza hacia fuera, y los distribuye. En el otro extremo, el TCP recibe los paquetes, verifica si hay errores y los vuelve a combinar para convertirlos en los datos originales. De haber error en algún punto, el programa TCP destino envía un mensaje solicitando que se vuelvan a enviar determinados paquetes.

1.7. Módulos de comunicación.

Se usa un módulo para la comunicación serie y otro para la comunicación por ethernet, utilizando el protocolo Modbus RTU y Modbus TCP/IP respectivamente. Ambos módulos de comunicación están implementados dentro del FPGA, conectados a las interfaces físicas correspondientes a los puertos serie y ethernet que adecuan los niveles de voltaje. Estas interfaces se encuentran en la tarjeta spartan 3e (2).

1.7.1. Protocolo Modbus. ASC II y RTU.

Modbus es un protocolo de comunicaciones situado en el nivel 7 del Modelo OSI, basado en la arquitectura maestro/esclavo o cliente/servidor, diseñado en 1979 por Modicon para su gama de controladores lógicos programables (PLCs). Convertido en un protocolo de

comunicaciones estándar de facto en la industria es el que goza de mayor disponibilidad para la conexión de dispositivos electrónicos industriales. Las razones por las cuales el uso de Modbus es superior a otros protocolos de comunicaciones son (2):

1. es público.
2. su implementación es fácil y requiere poco desarrollo.
3. maneja bloques de datos sin suponer restricciones.

Modbus permite el control de una red de dispositivos, por ejemplo un sistema de medida de temperatura y humedad, y comunicar los resultados a un ordenador. Modbus también se usa para la conexión de un ordenador de supervisión con una unidad remota (RTU) en sistemas de supervisión de adquisición de datos (SCADA).

Tipos de datos en Modbus

Modbus diferencia cuatro tipos de datos y tiene funciones específicas para ellos con un direccionamiento independiente. Se diferencian entre sí, en si son de lectura o escritura y si son tipo bit o tipo Word (16 bits).

Tipo de dato	Lectura/Escritura	Ancho dato	Comentario
Entradas discretas (Discret Input)	Lectura	1 bit	Variables de un bit que el maestro puede leer.
Salidas discretas (Discret output)	Lectura /Escritura	1 bit	Variables de un bit que el maestro puede escribir y leer.
Registros de entrada (Input Register)	Lectura	palabra (16 bits)	Variables de 16 bits que el maestro puede leer.
Registros de salida **(**Holding Register)	Lectura / Escritura	palabra (16 bits)	Variables de 16 bits que el maestro puede escribir y leer.

Tabla 1 Tipos de datos en Modbus.

Aunque el direccionamiento de cada tipo de dato es independiente, el esclavo o servidor puede tener estas áreas solapadas de manera que podamos acceder al mismo dato de diferente forma.

Integración con el modelo OSI

La armonización de Modbus y el modelo OSI es cumplimentado en las capas de nivel físico, de enlace y de aplicación:

- Nivel Físico: Par trenzado, velocidad máxima de 19200 Baudios, RS232/RS485/422
- Nivel de enlace: Acceso a la red del tipo maestro/esclavo. Control de tramas vía CRC.
- Nivel de Aplicación: Lectura y escritura de variables (bits, palabras, entrada/salida). Modos de operación. Diagnósticos. Intercambio de información.

El enrutamiento que estaría representado en el nivel de Red es transmitido por direcciones en soporte físico.

La comunicación vía Modbus es usada para el intercambio de datos entre todos los dispositivos conectados al bus. El protocolo Modbus crea una estructura jerárquica (un maestro y varios esclavos). Un enlace multipunto conecta el maestro y los esclavos. El maestro manipula exclusivamente toda la comunicación y son posibles dos formas de diálogo:

- El maestro intercambia con un esclavo y espera por su repuesta.
- El maestro intercambia con todos los esclavos sin esperar respuesta alguna.

Principios de operación de Modbus

Los dispositivos que se comunican usando el protocolo Modbus usan la técnica Maestro - Esclavo, en la cual un dispositivo, en este caso el maestro, inicia la negociación enviando una solicitud. El dispositivo para el cual la solicitud fue direccionada, un esclavo, envía su respuesta al maestro (Encuesta-Respuesta). El maestro también puede enviar un mensaje general para todos los esclavos en cuyo caso estos últimos no le responden al maestro (Difusión).

Encuesta-Respuesta:

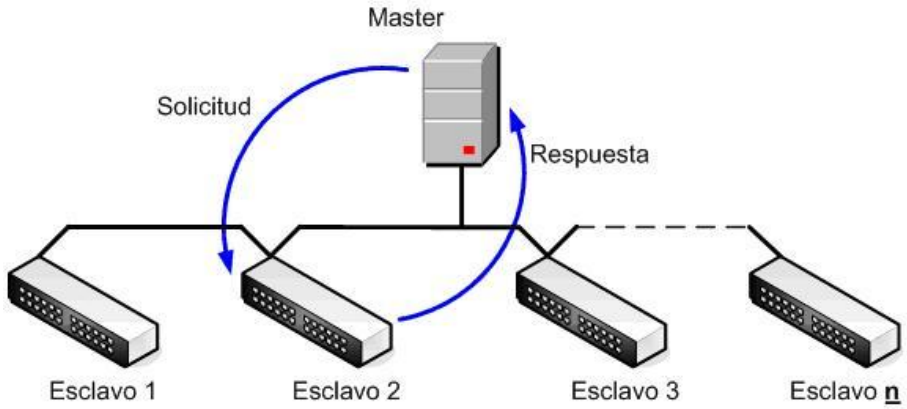

Figura 4 Encuesta – Respuesta.

Difusión:

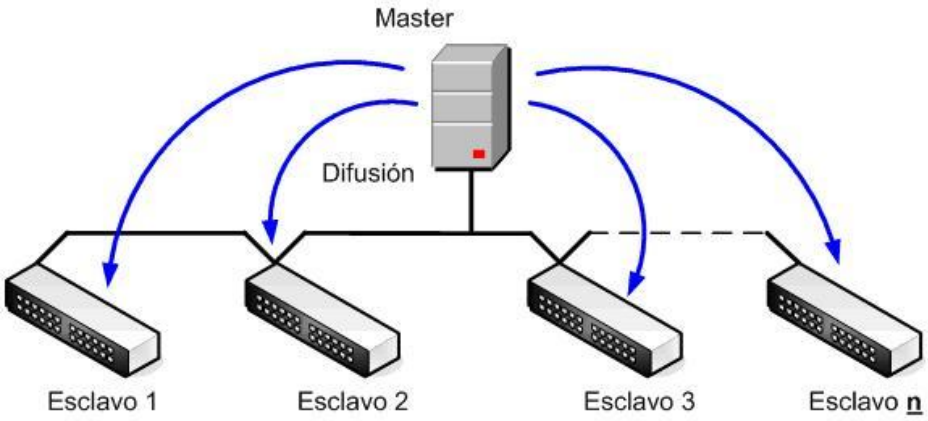

Figura 5 Difusión.

Características de la Encuesta:

La solicitud contiene un código que indica al esclavo adecuado que tipo de acción es solicitada. Los datos contienen información adicional la cual el esclavo necesita para ejecutar la función deseada. El campo de control de bytes le da la posibilidad al esclavo de verificar la integridad del contenido de la solicitud.

Características de la Respuesta:

Cuando un esclavo, siguiendo una negociación normal de la comunicación, envía una respuesta, el código de la función (Función Asociada) es un eco de lo que contenía la solicitud. Los datos son coleccionados por el esclavo, por ejemplo el valor de un registro o un estatus. Si ocurriera un error, el código de la función asociada es modificado para indicar que la respuesta es una respuesta de error. Los datos entonces contienen un código que permite el reconocimiento del tipo del error. El campo de control le permite al maestro la confirmación de la validez del mensaje.

Errores en Modbus

01: Función asociada desconocida

02: Dirección incorrecta.

03: Valor incorrecto.

04: Estación no lista para ejecutar la solicitud.

05: Acuse de recibo, la estación ha aceptado y está procesando la solicitud.

06: La estación está procesando y no está disponible.

07: No aceptación.

Otros códigos de error pueden existir y son específicos al producto conectado a la red.

Formato de la trama de encuesta/respuesta.

Encuesta:

No. – Esclavo 1 byte	Código –Función 1 byte	Datos n bytes	CRC 2 bytes

Respuesta Positiva:

No. – Esclavo 1 byte	Código –Función 1 byte	Datos -Procesados n bytes	CRC 2 bytes

Respuesta con Excepción:

No. – Esclavo 1 byte	Código –Función 1 byte	Código - Error n bytes	CRC 2 bytes

Figura 6 Formato de la trama Encuesta-Respuesta.

Tres tipos de codificaciones pueden ser usadas en la Red Modbus:

- Modbus ASC II
- Modbus TCP/IP
- Modbus RTU

Todos los dispositivos presentes en la Red tienen que ser configurados acorde al tipo de codificación, las cuales se explican a continuación:

1.7.1.1. Modbus ASC II.

En el modo ASCII todos los mensajes comienzan con el caracter *dos puntos* (:), y terminan con el caracter *cambio de línea-retorno del carro* (“CRLF”). Los caracteres enviados en los otros campos son hexadecimales (0-9, A-F). Los dispositivos en la red continuamente monitorean el arribo del caracter “:” y cuando esto ocurre, entonces cada dispositivo decodifica

el siguiente campo (Dirección) con el objetivo de discernir la dirección de destino y entonces tener en cuenta los siguientes caracteres si el esclavo es identificado. El final del mensaje es indicado por el carácter “CRLF” antecedido por los dos caracteres de control de errores que contienen el LRC (Longitudinal Reducing Check).

Inicio	Dirección	Función	Datos	LRC	Fin
1 caracter ":"	2 caracteres	2 caracteres	n caracteres	2 caracteres	2 caracteres

Figura 7 Formato de la trama Modbus ASC II.

1.7.1.2. Modbus TCP / IP.

Modbus TCP/IP es una variante o extensión del protocolo Modbus que permite utilizarlo sobre la capa de transporte TCP/IP (2). De este modo, Modbus-TCP se puede utilizar en Internet.

Modbus TCP/IP se ha convertido en un estándar industrial muy utilizado debido a su simplicidad, bajo coste, necesidades mínimas en cuanto a componentes hardware, y sobre todo a que se trata de un protocolo abierto. La combinación de una red física versátil y escalable como Ethernet con el estándar universal de interredes TCP/IP y una representación de datos independiente del fabricante, como Modbus, proporciona una red abierta y accesible para el intercambio de datos de proceso.

Modbus TCP/IP simplemente encapsula una trama Modbus en un segmento TCP. TCP proporciona un servicio orientado a conexión fiable, lo que significa que toda consulta espera una respuesta.

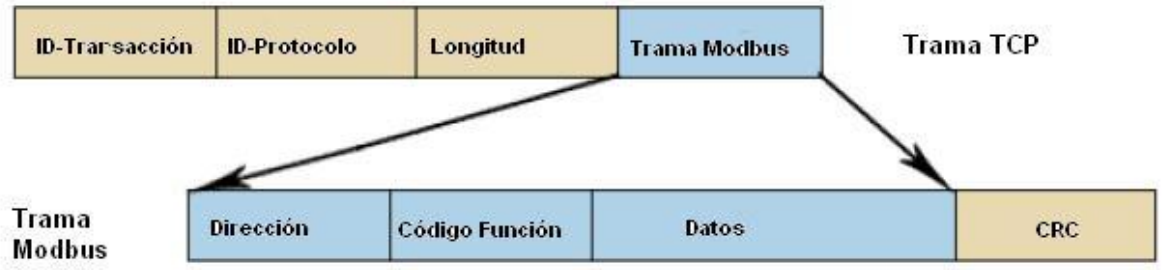

Figura 8 Encapsulamiento de la trama Modbus en TCP.

Esta técnica de consulta/respuesta encaja perfectamente con la naturaleza maestro/esclavo de Modbus, añadido a la ventaja del determinismo que las redes Ethernet conmutadas ofrecen a los usuarios en la industria. El empleo del protocolo abierto Modbus con TCP proporciona una solución para la gestión de unos pocos a decenas de miles de nodos.

1.7.1.3. Modbus RTU.

Como se enunció anteriormente en este trabajo se implementará el módulo de comunicación utilizando el protocolo Modbus en su variante serie RTU ya que es el modo más frecuentemente usado y es más eficiente que el modo ASCII (2). En el modo RTU el mensaje comienza con un período de silencio de al menos 3 caracteres y medio. Todos los dispositivos presentes en la red permanecen continuamente a la escucha, y decodifican el primer byte con el objetivo de determinar la dirección de destino, una vez que el dispositivo esclavo es identificado se prosigue a la recepción de los restantes campos de la trama. El siguiente byte que se recibe es el correspondiente al código de la función que se quiere ejecutar, en este caso se implementan 6 funciones Modbus correspondientes a la lectura y escritura de datos, más adelante se muestran dichas funciones con su código correspondiente. Luego se reciben los bytes de datos, los cuales contienen en primer lugar los parámetros necesarios para ejecutar la función indicada por el byte anterior, estos pueden ser direcciones del primer bit o byte, número de bits o palabras a leer o escribir, valor del bit o palabra en caso de escritura, etc. Los dos siguientes bytes corresponden a la palabra de control CRC.

Los caracteres son hexadecimales del tipo 0-9, A-F. Los datos contenidos en la trama tienen que contener todo el mensaje y deben ser enviados continuamente. La integridad del mensaje es indicada en el contenido del CRC (Código de Redundancia Cíclica).

Inicio	Dirección	Función	Datos	CRC	Fin
silencio	1 byte	1 byte	n bytes	2 bytes	silencio

Figura 9 Formato de la trama Modbus RTU.

La máxima longitud del mensaje es de 256 caracteres.

Funciones Modbus:

Modbus ofrece 24 funciones diferentes. Están caracterizadas por una función asociada a tamaño 1 byte. No todos los dispositivos admiten todas las funciones asociadas. A continuación se describen las que serán desarrolladas como parte de este trabajo.

- Lectura de n bits de salida.
  Código: 01 Esta función es usada para acceder a los bits de salida los cuales pueden ser escritos o leídos, y están definidos en la memoria del esclavo.
- Lectura de n bits de entrada.
  Código: 02 Esta función es idéntica a la anterior y tiene los mismos límites, es aplicable para los bits de entrada (El maestro puede leer pero no escribir).
- Lectura de n palabras de salida.
  Código: 03 Esta función es usada para acceder a las palabras de salida las cuales pueden ser escritas y leídas. Son definidas en la memoria del esclavo.
- Lectura de n palabras de entrada.
  Código: 04 Esta función, es idéntica a la anterior y tiene los mismos límites, es aplicable para las palabras de entrada (El maestro puede leer pero no escribir).
- Escribir 1 bit de salida.

Código: 05 Esta función es usada para fijar el estado de un bit de salida, el cual es definido en la memoria del esclavo como “0” o como “1” (Es de solo escritura).

- Escribir palabra de salida.
  Código: 06 Esta función, es idéntica a la anterior y tiene los mismos límites, es aplicable a las palabras de salida (Maestro puede escribir pero no leer).

1.8. Tipo de procesadores usados en sistemas embebidos.

Basado en el criterio de clasificación de reconfigurabilidad de la estructura interna del procesador, se pueden distinguir tres clases fundamentales (3):

- Hard Cores: Su estructura interna no puede ser modificada una vez fabricados.
- Firm Cores: Estos procesadores son reconfigurables, pero el diseñador no tiene acceso al código en lenguaje de descripción de hardware, en lugar de esto el procesador es configurado usando un entorno integrado de desarrollo.
- Soft Cores: Al diseñador le es posible acceder al código en lenguaje de descripción de hardware que define al procesador, pudiendo optimizarlo según sus necesidades.

1.8.1. Procesador MicroBlaze.

El procesador MicroBlaze es del tipo *Soft Core,* posee una arquitectura de 32 bits y ha sido desarrollado con unos requisitos muy rígidos respecto a la ocupación de recursos y prestaciones. La restricción en cuanto a los recursos a utilizar, fuerza a un diseño sencillo. Soporta hasta 87 instrucciones, cuenta con 32 registros de propósito general de 32 bits y 5 registros de propósito especial con igual cantidad de bits (3).

Los tipos de datos permisibles son:

- Byte: Ocho bits.
- Media palabra: Dieciséis bits.
- Palabra: Treinta y dos bits.

1.9. Herramientas, metodología y lenguaje de desarrollo.

1.9.1. UML (Lenguaje Unificado de Modelado).

El Lenguaje Unificado de Modelado (UML) se compone por diversos elementos gráficos que se combinan para conformar los diagramas. Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar los artefactos de un sistema, además puede utilizarse para escribir planos de software (20).

Es importante recalcar que UML no es una guía para realizar el análisis y el diseño orientado a objetos, es decir, no es un proceso, sino un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos. Un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementarlo.

El modelado de los sistemas con UML siguiendo los pasos del Proceso Unificado de Rational, que tiene como objetivo producir software de alta calidad, incluye actividades específicas y a su vez cada una de ellas contienen otras sub-actividades que sirven como una guía, de cómo deben ser las actividades desarrolladas y secuenciadas con el fin de obtener sistemas exitosos. Consecuentemente el desarrollo de los sistemas puede variar de desarrollador en proyecto o de empresa en empresa, adoptando siempre un Proceso de Desarrollo de Software.

1.9.2. Metodologías de Desarrollo de Software.

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Debido a la diversidad de propuestas nos enfrentamos a un problema en el momento de decidir qué metodología de desarrollo utilizar (21).

En un proceso de desarrollo de software la metodología define “Quién” debe hacer “Qué”, “Cuándo” y “Cómo” para alcanzar un objetivo y elevar la calidad del software que se produce en todas y cada una de sus fases de desarrollo. Seleccionar la metodología más apropiada que posibilite obtener óptimos resultados en el desarrollo de un software es una de las trabas principales de hoy en día (20).

El objetivo de una metodología de desarrollo es elevar la calidad del software (en todas las fases por las que pasa) a través de una mayor transparencia y control sobre el proceso. Es labor de la metodología de desarrollo hacer que esas medidas, para aumentar la calidad, sean reproducibles en cada desarrollo. A continuación se analizan las características de dos de las metodologías más usadas actualmente (21):

1.9.2.1. Proceso Unificado de Rational (RUP).

El software es un componente esencial de toda actividad basado en el uso de la informática, es por ello que la calidad en su desarrollo y mantenimiento resulta hoy en día uno de los principales objetivos estratégicos de las organizaciones.

El RUP es la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización (18).

El RUP provee un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su meta es asegurar la producción de software de muy alta calidad que satisfaga las necesidades de los usuarios finales, dentro de un calendario y presupuesto predecible.

El Proceso Unificado se basa en componentes, lo que significa que el sistema en construcción está hecho de componentes de software interconectados por medio de interfaces bien definidas. Este además usa el Lenguaje Unificado de Modelado (UML) en la preparación de todos los planos del sistema. De hecho, UML es una parte integral del Proceso Unificado, fueron desarrollados a la par. Los aspectos distintivos del Proceso Unificado están capturados en tres conceptos clave:

• **Dirigido por casos de uso:** Un caso de uso es una pieza en la funcionalidad del sistema que le da al usuario un resultado de valor. Los casos de uso capturan los requerimientos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso el cual describe la funcionalidad completa del sistema. Este modelo reemplaza la tradicional

especificación funcional del sistema. Los casos de uso son una herramienta para especificar los requerimientos del sistema, dirigen su diseño, implementación y pruebas, dirigen el proceso de desarrollo (18).

• **Centrado en la arquitectura:** El concepto de arquitectura de software involucra los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura surge de las necesidades de la empresa, tal y como las interpretan los usuarios y otros stakeholders, y tal y como están reflejadas en los casos de uso. La arquitectura es la vista del diseño completo con las características más importantes hechas más visibles y dejando los detalles de lado. Ya que lo importante depende en parte del criterio, el cual a su vez viene con la experiencia, el valor de la arquitectura depende del personal asignado a esta tarea (18).

• **Iterativo e incremental:** Desarrollar un producto de software comercial es una tarea enorme que puede continuar por varios meses o años. Es práctico dividir el trabajo en pequeños pedazos o mini-proyectos. Cada mini-proyecto es una iteración que finaliza en un incremento. Las iteraciones se refieren a pasos en el flujo de trabajo, los incrementos se refieren a crecimiento en el producto. Para ser más efectivo, las iteraciones deben estar controladas, esto es, deben ser seleccionadas y llevadas a cabo de una manera planeada (18).

En la Figura 10 se representa el RUP, donde se grafican los flujos de trabajo, las fases y la dinámica expresada en iteraciones y puntos de control.

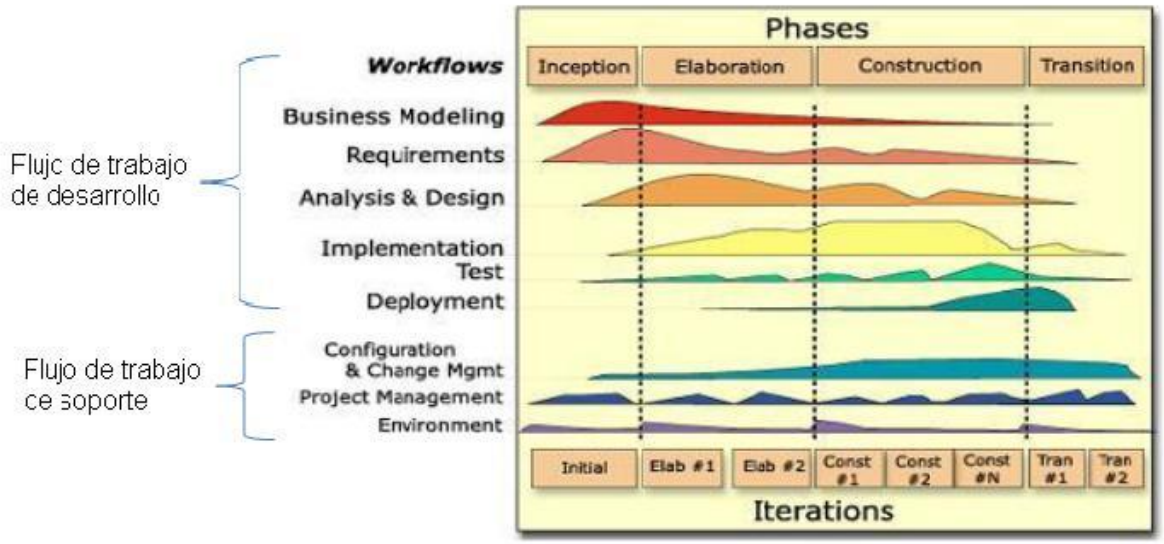

Figura 10 RUP en dos dimensiones.

1.9.2.2. Extreme Programming (XP).

Esta metodología se basa en la idea de que existen cuatro variables que guían el desarrollo de sistemas: Costo, Tiempo, Calidad y Alcance. La manera de encarar los desarrollos avalados por este modelo de desarrollo es permitir a las fuerzas externas (gerencia, clientes) manejar hasta tres de estas variables, quedando el control de la restante en manos del equipo de desarrollo. Este modelo hace visibles de manera más o menos continua estas cuatro variables. (19)

Características de XP

Es imposible prever todo antes de comenzar a programar; es imposible o si lo fuera es demasiado costoso e innecesario, ya que muchas veces se gasta demasiado tiempo y recursos en cambiar la documentación de la planificación para que se parezca al código. Para evitar esto, XP intenta implementar una forma de trabajo donde se adapte fácilmente a las circunstancias. Básicamente consiste en trabajar estrechamente con el cliente, haciendo pequeñas iteraciones (mini-entregas), cada dos semanas, donde no existe más documentación que el código en sí; cada versión contiene las modificaciones necesarias según el cliente vaya retroalimentando el sistema (por eso es necesaria la disponibilidad del

cliente durante todo el desarrollo). Para suplir la falta de requisitos, casos de uso, y demás herramientas; XP utiliza historias de usuarios, la historia de usuario es una frase corta que representa alguna función que realizará el sistema. Cada historia de usuario no puede demorar en desarrollarse más de una semana, si así lo requiriera, debe segmentarse (17).

Es requisito para XP definir un estándar en el tipo de codificación, esto hace que los programadores tengan definido ya el estilo de programación y no que cada uno programe a su estilo. Las pruebas en cada iteración son más que importante; de eso se trata este paradigma de programación, corregir mientras se programa. De esta forma se van cubriendo todos los baches que cada versión padezca. El código no es de nadie, todo el equipo puede manipular el código que existe, de esta forma cada pareja puede mejorar cada sección de código que utiliza, esto requiere de una prueba del mismo y la re-implementación en el sistema general. Cada dos semanas se entrega una versión al cliente, que lo verifica, realiza la retroalimentación y se continúa el desarrollo; este ciclo continúa hasta que el sistema cumpla con las expectativas del cliente, acto que concluirá el proyecto (19).

1.9.2.3. Selección de la metodología de desarrollo.

Luego de realizado el estudio de estas metodologías se determinó que la metodología a utilizar como base para el desarrollo de la solución propuesta es RUP debido a que es una de las más completas y abarcadoras, además de estar muy bien organizada y documentada. Si comparamos con respecto a XP esta presenta una documentación muy pobre y UML juega un papel prácticamente nulo. Además es el resultado de la evolución e integración de diferentes metodologías de desarrollo de software.

1.9.3. Herramientas CASE.

Las Herramientas CASE (*Computer Aided Software Engineering*,) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como, el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del

código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras (7).

Estas tienen como objetivos:

- Mejorar la productividad en el desarrollo y mantenimiento del software.
- Aumentar la calidad del software.
- Mejorar el tiempo y coste de desarrollo y mantenimiento de los sistemas informáticos.
- Mejorar la planificación de un proyecto.
- Aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos.
- Automatizar, desarrollo del software, documentación, generación de código, pruebas de errores y gestión del proyecto.
- Ayuda a la reutilización del software, portabilidad y estandarización de la documentación.
- Gestión global en todas las fases de desarrollo de software con una misma herramienta.
- Facilitar el uso de las distintas metodologías propias de la ingeniería del software.

1.9.3.1. Rational Rose.

Es la herramienta CASE que comercializan los desarrolladores de UML (Booch, Rumbaugh y Jacobson) y que soporta de forma completa la especificación del UML. Esta herramienta propone la utilización de cuatro tipos de modelos para realizar un diseño del sistema, utilizando una vista estática, otra dinámica, de los modelos del sistema, una lógica y otra física; que permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software. (10)
Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML.

Rational Rose proporciona mecanismos para realizar la denominada Ingeniería Inversa, es decir, a partir del código de un programa, se puede obtener información sobre su diseño. Cubre todo el ciclo de vida de un proyecto:

- Concepción y formalización de un proyecto.
- Construcción de los componentes.
- Transición a los usuarios y certificación de las distintas fases.

1.9.3.2. Visual Paradigm.

Visual Paradigm es un laureado producto que facilita a las organizaciones el diseño visual y el diagrama, integrar y desplegar sus aplicaciones empresariales de misión crítica y sus bases de datos subyacentes. La herramienta de desarrollo de software ayuda a su equipo a sobresalir todo el modelo de construcción y despliegue del software de proceso de desarrollo, y a aumentar al máximo la aceleración de ambos equipos y de los individuos (6).

Características:

- ✓ Producto de calidad.
- ✓ Soporta aplicaciones Web.
- ✓ Las imágenes y reportes generados, no son de muy buena calidad.
- ✓ Varios idiomas.
- ✓ Generación de código para Java y exportación como HTML.
- ✓ Fácil de instalar y actualizar.
- ✓ Compatibilidad entre ediciones.

Además Visual Paradigm ofrece:

- ✓ Entorno de creación de diagramas para UML 2.0
- ✓ Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.

- ✓ Capacidades de ingeniería directa (versión profesional) e inversa.
- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- ✓ Disponibilidad de múltiples versiones, para cada necesidad.
- ✓ Disponibilidad de integrarse en los principales IDEs.
- ✓ Disponibilidad en múltiples plataformas.

1.9.3.3. Selección de la herramienta CASE.

Después de analizar las características de estas dos herramientas, se determinó utilizar para el trabajo con la Ingeniería de Software, Visual Paradigm, ya que esta herramienta tiene la ventaja de poseer versiones multiplataforma, además es completamente compatible con la metodología que se decidió utilizar, brinda muchas facilidades en la generación de la documentación del software que se está desarrollando. Visual Paradigm posee licencia gratuita y comercial, es un producto de mucha calidad, configurable para el uso en varios idiomas, multilenguaje, muy fácil de instalar y actualizar además de brindar compatibilidad entre ediciones.

1.9.4. Herramienta para la implementación del firmware.

En la presente investigación se desarrollará un firmware, para un PLC sobre un procesador MicroBlaze, para ello es necesario utilizar el paquete de software EDK, el cual permite (3):

- • Seleccionar y configurar el hardware asociado al procesador.
- • Desarrollar aplicaciones de software en los lenguajes C y C++.
- • Simular y depurar la aplicación desarrollada.
- • Cargar el sistema en la FPGA para su ejecución.

Para trabajar en EDK, se necesita tener previamente instalado el entorno de desarrollo *ISE* (Entorno de Software Integrado), esto se debe a que EDK realiza llamadas al mismo para sintetizar el hardware del sistema que se está implementando.

1.9.5. Lenguaje de desarrollo.

La herramienta EDK utiliza para el desarrollo el lenguaje C.

C es un lenguaje de alto nivel, que permite programar con instrucciones de lenguaje de propósito general (15).

También C se define como un lenguaje de programación estructurado de propósito general; aunque en su diseño también primó el hecho de fuera especificado como un lenguaje de programación de sistemas, lo que proporciona una enorme cantidad de potencia y flexibilidad.

El estándar ANSI C formaliza construcciones no propuestas en la primera versión del lenguaje C, en especial asignación de estructuras y enumeraciones. Entre otras aportaciones, se definió esencialmente la biblioteca estándar de funciones otra de las grandes aportaciones.

En la actualidad, el lenguaje C sigue siendo uno de los más utilizados en la industria del software, así como en institutos tecnológicos, escuelas de ingeniería y universidades.

Prácticamente todos los fabricantes de sistemas operativos (tomando en cuenta a: UNIX, Linux, Solaris, Windows, entre otros.), soportan diferentes tipos de compiladores de lenguaje C.

Características del lenguaje C

Hay numerosas características que diferencian al lenguaje C de otros, y lo hacen eficiente, potente, eficaz, rápido, indispensable para todos los programas. Algunas son (15):

- Una nueva sintaxis para declarar funciones. Una declaración de función puede añadir una descripción de los argumentos de la función. Esta información adicional sirve para que los compiladores detecten más fácilmente lo errores causados por argumentos que no coinciden.
- Asignación de estructuras (registros) y enumeraciones.
- Pre-procesador más sofisticado.
- Una nueva definición de la biblioteca que acompaña a C. Entre otras funciones se incluyen: acceso al sistema operativo (por ejemplo, lectura / escritura de archivos), entrada y salida con formato, asignación dinámica de memoria, manejo de cadenas de caracteres.

- Una colección de cabeceras estándar que proporciona acceso uniforme a las declaraciones de funciones y tipos de datos.

Ventajas del lenguaje C

El lenguaje C tiene una gran cantidad de ventajas sobre otros lenguajes y constituyen precisamente la razón fundamental de que después de casi dos décadas de uso, C siga siendo uno de los lenguajes más populares, utilizados en empresas, organizaciones y fábricas de software de todo el mundo.

- C se caracteriza por su velocidad de ejecución. En los primeros días de la informática los problemas de tiempo de ejecución se resolvían escribiendo todo o parte de una aplicación en lenguaje ensamblador (muy al lenguaje de máquina).
- Debido a que existen muchos programas escritos en el lenguaje C, se han creado numerosas bibliotecas C para programadores profesionales que soportan gran variedad de aplicaciones.

1.10. Conclusiones Parciales.

Se caracterizaron los PLC´s y se realizó un estudio sobre la arquitectura hardware y funcionalidades de los controladores lógicos programables actuales. Se estudiaron los aspectos fundamentales del hardware reconfigurable en el control industrial actual.

Se analizaron las características del firmware y el papel que juega este dentro del PLC, además se estudiaron los aspectos principales del protocolo de comunicación Modbus. Se utilizará para la implementación la herramienta EDK, y dentro de esta el paquete XPS, esto implica que el lenguaje de desarrollo de esta investigación sea el lenguaje C. La metodología RUP se usará para guiar el proceso de desarrollo de software, se utilizará UML para el modelado y como herramienta CASE Visual Paradigm.

Capítulo 2: Características y Diseño del Sistema.

2.1. Introducción.

En este capítulo se analizan las características que va a presentar el sistema, realizando para ello el levantamiento de requisitos. Se capturan los tipos de objetos más importantes que existen o los eventos que suceden en el entorno donde estará el sistema, mostrándolos en el Modelo del Dominio. Además se realiza el diseño del sistema a través del Diagrama de Clases del Diseño.

2.2. Modelo del Dominio.

Un modelo del dominio captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema. Es una representación visual estática del entorno real objeto del proyecto, es decir, un diagrama con los objetos que existen (reales) relacionados con el proyecto que vamos a acometer y las relaciones que hay entre ellos. Pero no son clases de software (aunque algunos objetos del Modelo de Dominio pueden terminar siéndolo). (13)

Se llama "de Dominio" para distinguirlo del Modelo de Negocio (Model of Business) que en RUP es un concepto más amplio. El Modelo de Negocio incluye toda la organización, sus relaciones, sus procesos... todo. Sin embargo, el Modelo de Dominio se centra en una parte del negocio, la relacionada con el ámbito del proyecto. En este contexto el término "dominio" representa una parte del "negocio".

Se dice que es estática porque no representa la interacción en el tiempo de los objetos, sino que representa una visión "parada" de las clases y sus interacciones.

Por su parte, el Diccionario de Clases del Dominio describe textualmente las clases identificadas durante el modelado del dominio del problema. Este diccionario sirve como un glosario de términos y se muestra a continuación:

Trama Modbus: caracteriza a la trama de solicitud que es enviada al sistema.

Dispositivo: es el encargado de recibir la trama, procesar todos sus campos y luego enviar la respuesta correspondiente.

Trama Respuesta: caracteriza a la trama de respuesta que es enviada por el dispositivo una vez que se ha atendido la solicitud.

Campos: contiene todos los campos que componen tanto la trama de solicitud como la de respuesta y estos son: dirección del dispositivo, código de la función, datos, CRC.

2.2.1. Diagrama de Clases del Dominio.

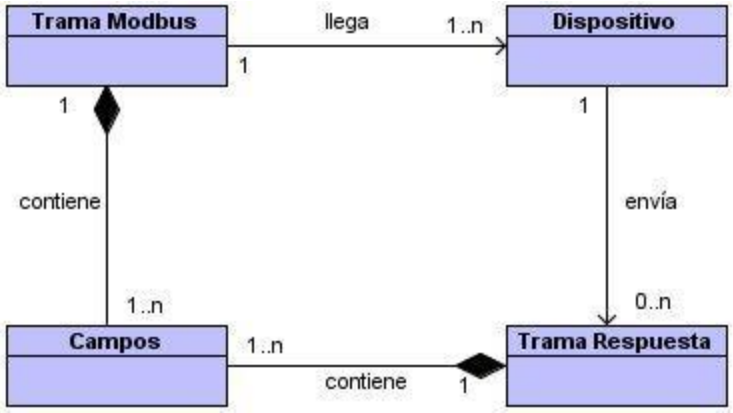

Figura 11 Diagrama de Clases del Dominio.

2.3. Especificación de Requerimientos del Sistema.

Un requerimiento es una condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo. Tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente (12).

Los requisitos se pueden clasificar en: funcionales y no funcionales.

2.3.1. Requerimientos funcionales.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir (12).

RF 1. Recibir una trama Modbus.

RF 1.1. Decodificar el primer byte de la trama que indica la dirección del esclavo.

RF 1.2. Identificar y ejecutar la función solicitada.

RF 1.3. Calcular y Aplicar CRC a la trama.

RF 1.4. Conformar la trama de respuesta.

RF 2. Enviar trama de Respuesta.

2.3.2. Requerimientos no funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación (12).

A continuación se detallan los requerimientos no funcionales del sistema:

1. Requisitos de Software.

RNF 1. Instalar el paquete EDK.

RNF 1.1. Instalar previamente el entorno de desarrollo *ISE.*

2. Requisitos de Hardware.

RNF 2.1 Se deberá disponer de un FPGA.

3. Requisitos de Usabilidad.

RFN 3.1. El sistema podrá ser utilizado para la comunicación serie del PLC con cualquier otro dispositivo que implemente Modbus.

4. Requisitos de Soporte.

RNF 4.1. El sistema será probado una vez finalizada su implementación.

5. Requisitos de Portabilidad.

RNF 5.1. Se deberá contar con al menos 8 KB de memoria en los FPGA´s donde se va a correr la aplicación.

6. Restricciones en el diseño y la implementación.

RNF 6.1. Se utilizará el Lenguaje C para la implementación.

RNF 6.2. Se empleará como herramienta de desarrollo el paquete EDK.

2.4. Modelo del Sistema.

2.4.1. Actores del Sistema.

Un actor es una idealización de una persona externa, de un proceso, o de una cosa que interactúa con un sistema, un subsistema, o una clase. Un actor caracteriza las interacciones que los usuarios exteriores pueden tener con el sistema, puede representar el rol que juega una o varias personas, un equipo o un sistema automatizado. Suelen ser definidos en jerarquías de generalización, en las cuales una descripción abstracta del actor es compartida y aumentada por una o más descripciones específicas del actor (1).

Por lo general estimula el sistema con eventos de entradas o recibe algo de él. O sea, es un rol de un usuario, que puede intercambiar información o puede ser un recipiente pasivo de información y representa a un ser humano, a un software o a una máquina que interactúa con el sistema.

Un actor es una entidad externa del sistema que de alguna manera participa en la historia del caso de uso.

A continuación en la tabla 2 se muestra la descripción del actor del sistema.

Actor	Función	Rol
Sistema Automatizado(PLC)	Es el encargado de recibir una trama Modbus, interpretar todos sus campos y posteriormente enviar la respuesta correspondiente.	• Decodificar el primer byte de la trama que indica la dirección del dispositivo al cual se le envía la trama. • En caso de que el dispositivo sea identificado, llevar a cabo la interpretación del resto de los campos de la trama. • Conformar la trama de respuesta y enviarla.

Tabla 2 Descripción de Actores del Sistema.

2.4.2. Diagrama de Casos de Uso del Sistema.

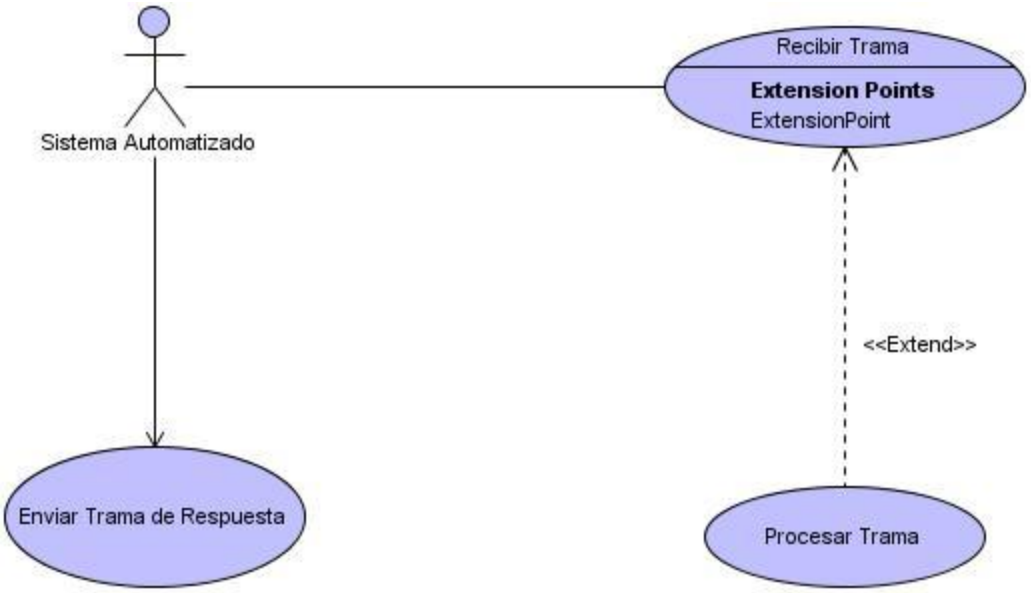

Figura 12 Diagrama de Casos de Uso del Sistema.

2.4.3. Descripción de Casos de Uso del Sistema.

El sistema tiene 3 casos de usos significativos Recibir Trama, Procesar Trama y Enviar Trama de Respuesta.

A continuación se muestran las descripciones de los mismos en las tablas 3, 4 y 5 respectivamente.

CU	Recibir Trama
Actor	Sistema automatizado(PLC)
Resumen	Este caso de uso se inicia cuando el sistema recibe una posible solicitud proveniente de un dispositivo. Posteriormente el sistema debe decodificar el primer byte de la trama

	con el objetivo de identificar la dirección de destino.
Referencia	RF 1, RF 1.1
Precondiciones	El sistema no ha recibido ninguna trama. anteriormente.
Poscondiciones	El sistema recibe una posible solicitud.

Tabla 3 Descripción de Caso de Uso Recibir Trama.

CU	Procesar Trama
Actor	Sistema automatizado(PLC)
Resumen	Este caso de uso se inicia cuando el sistema ha comprobado que le han enviado una solicitud y pasa a analizar los restantes campos de la trama.
Referencia	RF 1, RF 1.2, RF 1.3, RF 1.4
Precondiciones	El sistema ha sido identificado.
Poscondiciones	Analizar el resto de la trama.

Tabla 4 Descripción de Caso de Uso Procesar Trama.

CU	Enviar Trama de Respuesta
Actor	Sistema automatizado(PLC)
Resumen	Este caso de uso se inicia cuando el sistema ha analizado todos los campos de la solicitud por lo que debe enviar una respuesta.
Referencia	RF 2
Precondiciones	El sistema ha procesado la trama de solicitud.
Poscondiciones	Esperar una nueva solicitud.

Tabla 5 Descripción de Caso de Uso Enviar Trama de Respuesta.

2.5. Patrones de Diseño.

Un patrón es un modelo que podemos seguir para realizar algo. Los patrones capturan la experiencia existente y probada para promover buenas prácticas. Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular (1).

Un patrón de diseño es una solución estándar para un problema común de programación, es una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios. Podemos definirlo además como un proyecto o estructura de implementación que logra una finalidad determinada.

2.5.1. Patrones GRASP.

Los patrones GRASP (Patrones Generales de Software para Asignación de Responsabilidades) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Dentro del grupo de los patrones GRASP utilizaremos:

Experto.

Nos indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo.

En este caso cada uno de los métodos es implementado en las clases que conocen toda la información necesaria para crearlos (1).

Alta Cohesión.

La información que almacena una clase debe ser coherente y debe estar (en la medida de lo posible) relacionada con la clase (1).

En este caso la información que se almacena en cada clase está relacionada con ella, por ejemplo la clase modbus.h almacena toda la información del protocolo Modbus RTU.

Bajo Acoplamiento.

El Bajo Acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad (1).

El sistema se desarrolló de manera tal que las clases estuvieran lo menos ligadas entre sí, con el objetivo de que si se produce una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases.

2.6. Diagrama de Clases del Diseño.

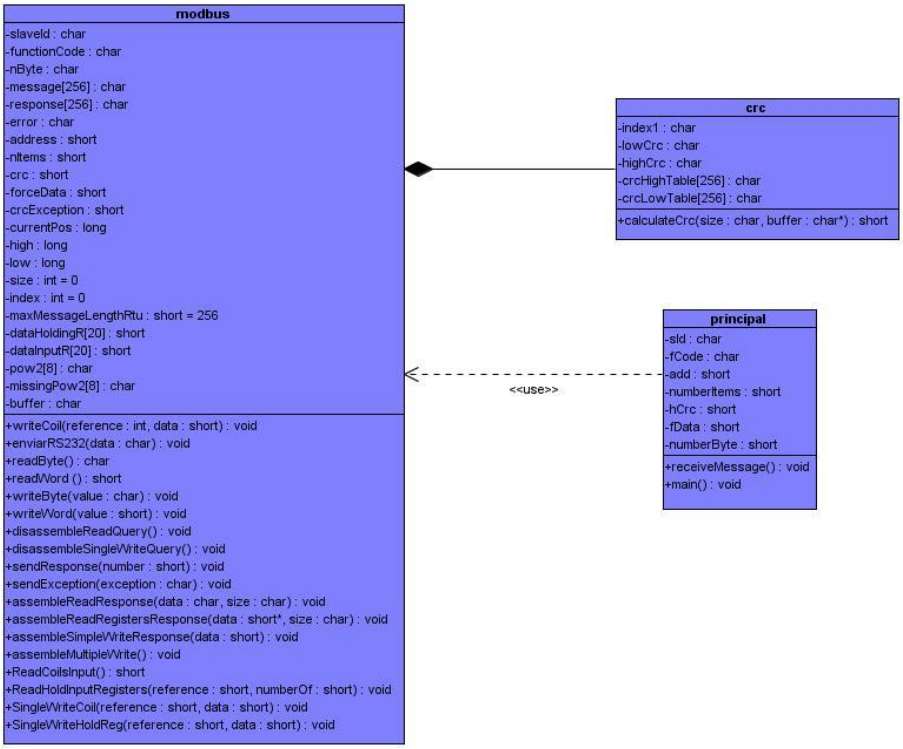

Figura 13 Diagrama de Clases del Diseño.

2.7. Descripción de la Arquitectura.

Una Arquitectura de Software, también denominada Arquitectura Lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software de un sistema de información. La Arquitectura de Software establece los fundamentos para trabajar en una línea común que permite alcanzar los objetivos y necesidades del sistema de información (1).

2.7.1. Arquitectura Cliente/Servidor.

Para el desarrollo del módulo de comunicación utilizando el protocolo Modbus RTU se utilizó la arquitectura Cliente/Servidor.

Este tipo de arquitectura proporciona que los recursos estén centralizados, debido a que el servidor es el centro de la red, puede administrar los recursos que son comunes a todos los usuarios, evitando problemas provocados por datos contradictorios y redundantes. Cliente/Servidor proporciona una seguridad mejorada, una administración al nivel del servidor ya que los clientes requieren menos administración sobre el sistema y una red escalable pues mediante esta arquitectura es posible quitar o agregar clientes sin afectar el funcionamiento de la red y sin la necesidad de realizar mayores modificaciones (1).

En este caso el cliente se corresponde con un Scada o una aplicación, ya sea un simulador u otro, que utilice Modbus y el servidor al PLC. El cliente es el que inicia la comunicación con el servidor, enviando un mensaje con la función que debe realizar el mismo, una vez que este ha verificado que el mensaje ha sido enviado para él, comienza a procesar dicha trama y luego de ejecutar la función conforma la trama de respuesta y la envía, a continuación en la figura 14 se muestra como quedaría este proceso:

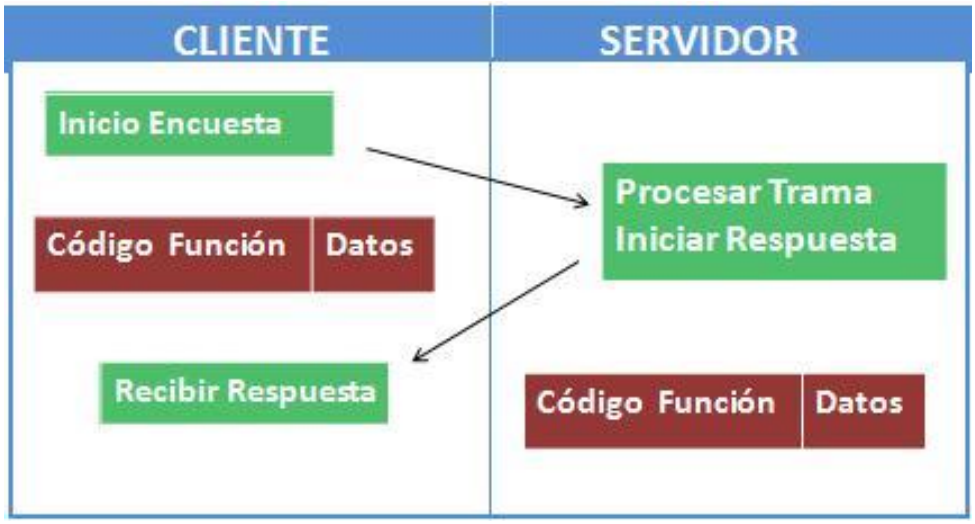

Figura 14 Arquitectura Cliente/Servidor.

2.8. Conclusiones Parciales.

En este capítulo se refinaron los requerimientos funcionales del sistema, los cuales son de gran importancia para su desarrollo, pues constituyen la funcionalidad del mismo, además fue presentado el diagrama de clases del diseño.

Se realizó un refinamiento de los requerimientos no funcionales, los que son de gran importancia ya que son propiedades o cualidades que el sistema debe cumplir y además se detallaron los casos de usos arquitectónicamente significativos para la implementación del sistema propuesto. Se seleccionó la arquitectura que se utilizará en el desarrollo de la aplicación.

Capítulo 3: Implementación y Validación Del Sistema.

3.1. Introducción.

En este capítulo se abordan los aspectos más importantes relacionados con la implementación del sistema y la validación del mismo, se realiza el diagrama de despliegue, se enuncian los estándares de codificación que se siguieron para generar el código. También se realiza la validación del sistema a través de los casos de pruebas que se le realizaron a las funcionalidades del sistema.

3.2. Modelo de Implementación.

Una vez realizado el diseño del sistema se le da paso a la etapa de implementación cuyo propósito es desarrollar la arquitectura y el sistema como un todo, en esta etapa es donde se implementan las clases encontradas durante el diseño.

3.2.1. Diagrama de Despliegue.

El diagrama de despliegue permite apreciar de forma visual cómo se encuentran relacionados físicamente los componentes de la aplicación. Describe la distribución física del sistema, muestra cómo están distribuidos los componentes de software entre los distintos nodos de cómputo. Este diagrama permite comprender la correspondencia entre ambas arquitecturas software y hardware. Los nodos se utilizan para modelar la topología del hardware sobre el cual se ejecuta el sistema. Representa típicamente un procesador o un dispositivo sobre el que se pueden desplegar los componentes (1).

En este caso, el usuario accede desde una aplicación de escritorio mediante el protocolo Modbus en su variante RTU al PLC que se encargará de ejecutar las órdenes que se le han enviado.

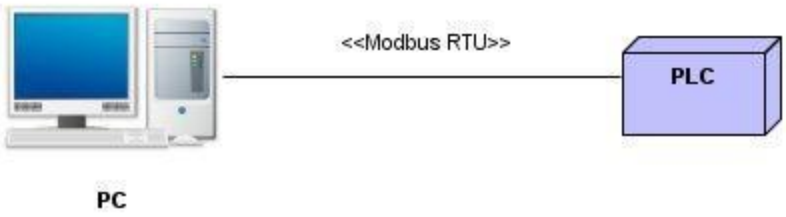

Figura 15 Diagrama de Despliegue.

3.2.3. Diagrama de Componentes.

Los diagramas de componentes se utilizan para mostrar las dependencias de compilación de los ficheros de código, relaciones de derivación entre ficheros de código fuente y ficheros que son resultados de la compilación, dependencias entre los elementos de implementación y los elementos correspondientes del diseño que son implementados (20).

Si se tienen en cuenta las dependencias asociadas al proceso de compilación, un componente podría ser un código ejecutable que puede depender de otros programas ejecutables con los que interactúa en tiempo de ejecución.

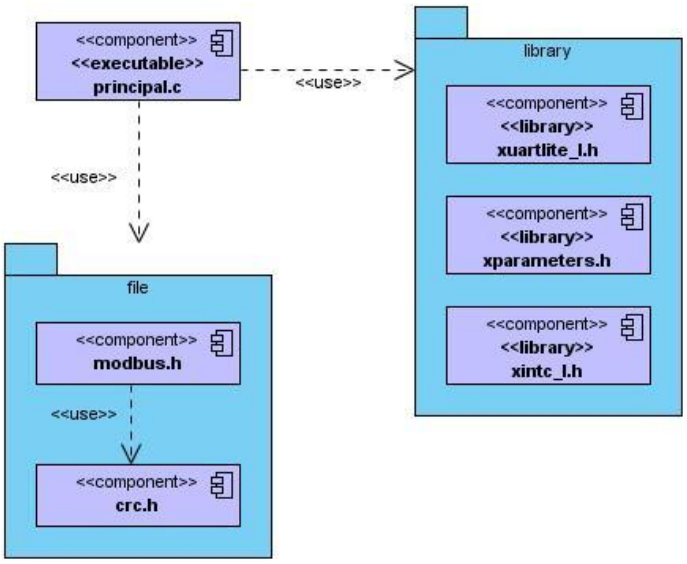

Figura 16 Diagrama de Componentes.

3.3. Estándares de Codificación.

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo armónico, la legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. Para mantener un código de calidad es necesario establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código.

Adoptar un estándar de codificación sólo es viable si se sigue desde el inicio hasta el final del proyecto de software. No es correcto adoptar un estándar de codificación una vez iniciado el trabajo.

En este caso los nombres de acciones deben estar en la nomenclatura “CamelCase”. Para nombrar las variables se seguirá la regla de escribir los identificadores con letras minúsculas y en inglés, no se usará separador entre palabras, sino que las siguientes palabras se escribirán con letra inicial mayúscula, tratando de usar nombres sugerentes a la acción de la variable.

Para comentar el código se utilizará, en el caso de una línea, al final de la misma el carácter “//“y seguido el comentario y en el caso de un bloque se utilizarán los caracteres “/* */”. Las llaves se usarán poniendo la llave inicial en una línea para ella sola, y en su respectiva columna la llave final también en una línea.

3.4. Funcionamiento del Sistema.

Para recibir la trama Modbus proveniente del dispositivo maestro se realiza la llamada al método receiveMessage() dentro del método main(),a partir de aquí se comienza a recibir el mensaje, primeramente se verifica la dirección del esclavo indicada en el primer byte recibido, en caso de que coincida con el dispositivo se recibe el siguiente byte correspondiente al código de la función que se quiere ejecutar, en este caso solo se ejecutarán las funciones de lectura y escritura de datos. Si la función es de lectura (códigos 01, 02, 03 y 04) se reciben los siguientes campos correspondientes a dicha función y se conforma el mensaje Modbus, luego se realiza la llamada al método disassembleReadQuery(), este es el encargado de interpretar cada campo del mensaje. En caso de que la función sea de simple escritura (código 05 y 06),

al igual que en las funciones de lectura se reciben los restantes campos de la trama, y luego mediante la llamada al método disassembleSingleWriteQuery() se realiza la interpretación de cada uno de los campos. En los dos métodos una vez que se interpretan los campos se realiza el cálculo del CRC para verificar la integridad del mensaje, mediante el método calculateCrc(), contenido en la clase crc.h, si se produce un error se envía una respuesta de excepción, dicha respuesta es enviada a través del método sendException(), en caso contrario se procede a ejecutar la función deseada y enviar la respuesta correspondiente, para ello se utiliza el método sendResponse(). Tanto sendException() como sendResponse(), utilizan el método enviarRS232(), que es el encargado de enviar un byte por el puerto serie. A continuación se muestran algunos de los métodos mencionados anteriormente.

Método para enviar un byte por el puerto serie.

```
void enviarRS232(unsigned char dato)
 {

       while(XUartLite_IsTransmitFull(XPAR_RS232_DCE_BASEADDR));
       XUartLite_SendByte(XPAR_RS232_DCE_BASEADDR,dato);


 }
```

Método principal

```
void main()
{

      for(;;)
      {

            receiveMessage();
      }


      while(1)
      {
      }

}
```

Método para recibir el mensaje

```
void receiveMessage()
 {
      //......Recibir codigo del esclavo.....
```

```
    while(XUartLite_IsReceiveEmpty(XPAR_RS232_DCE_BASEADDR));
    sId = XUartLite_RecvByte(XPAR_RS232_DCE_BASEADDR);
    size = 0;

if(sId==1)
{
     message[size ++]= sId;
     //......Recibir codigo de la funcion.....
     while(XUartLite_IsReceiveEmpty(XPAR_RS232_DCE_BASEADDR));
     fCode = XUartLite_RecvByte(XPAR_RS232_DCE_BASEADDR);
     message[size ++]= fCode;

  if(fCode == 1 || fCode == 2 || fCode==3  || fCode==4)
  {
    //......Recibir direccion del primer elemento.......
    for(i=0; i< 2; i++)
    {
        while(XUartLite_IsReceiveEmpty(XPAR_RS232_DCE_BASEADDR));
        add = XUartLite_RecvByte(XPAR_RS232_DCE_BASEADDR);
        message[size ++]= add;
    }

    //.......Recibir cantidad de elementos a leer......
    for(i=0; i< 2; i++)
    {
        while(XUartLite_IsReceiveEmpty(XPAR_RS232_DCE_BASEADDR));
        numberItems = XUartLite_RecvByte(XPAR_RS232_DCE_BASEADDR);
        message[size ++]= numberItems;

    }

    //........Recibir los 2 bytes de CRC.....
    for(i=0; i< 2; i++)
    {
         while(XUartLite_IsReceiveEmpty(XPAR_RS232_DCE_BASEADDR));
         hCRC = XUartLite_RecvByte(XPAR_RS232_DCE_BASEADDR);
         message[size ++]= hCRC;

    }
   disassembleReadQuery();
}
if(fCode == 5 || fCode == 6)
 {
     //......Recibir direccion del primer elemento.......
     for(i=0; i< 2; i++)
     {
         while(XUartLite_IsReceiveEmpty(XPAR_RS232_DCE_BASEADDR));
         add = XUartLite_RecvByte(XPAR_RS232_DCE_BASEADDR);
         message[size ++]= add;
     }

    //.......Recibir dato a forzar......
    for(i=0; i< 2; i++)
    {
         while(XUartLite_IsReceiveEmpty(XPAR_RS232_DCE_BASEADDR));
```

```
            fData = XUartLite_RecvByte(XPAR_RS232_DCE_BASEADDR);
            message[size ++]= fData;

        }

        //........Recibir los 2 bytes de CRC.....
        for(i=0; i< 2; i++)
        {
            while(XUartLite_IsReceiveEmpty(XPAR_RS232_DCE_BASEADDR));
            hCRC = XUartLite_RecvByte(XPAR_RS232_DCE_BASEADDR);
            message[size ++]= hCRC;

        }
      disassembleSingleWriteQuery();
   }
}
```

3.5. Fase de Prueba.

La fase de pruebas en el desarrollo de software es muy importante ya que esta consiste en probar las aplicaciones construidas, además las pruebas permiten verificar, validar y revelar la calidad del producto. En esta fase es donde se encuentran y se corrigen los errores de las fases anteriores (5).

3.5.1. Recursos empleados en la realización de las pruebas.

Para llevar a cabo las pruebas se emplearon una serie de recursos con el objetivo de facilitar el trabajo al ejecutar cada caso de prueba.

- Recursos Físicos

  Las computadoras utilizadas para el desarrollo de las pruebas contaron con 1.0 giga byte de memoria RAM, microprocesador Intel Core2Duo E4500 con velocidad de 2.20 Ghz, motherboard Intel y una capacidad en disco duro de 160 gigas.

- Recursos Lógicos

  El sistema operativo en el cual se desarrollaron las pruebas fue Windows SP 3.
  Se utilizó el simulador de Modbus ModScan32.

La velocidad de la red con que se contó fue de 9600 bits por segundo.

3.5.2. Pruebas de Caja Negra.

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software (14).

3.5.2.1. Casos de prueba.

Un caso de prueba permite detallar la forma en que se va a probar el sistema, incluyendo los datos de entrada con los que se realizará la prueba correspondiente, las condiciones de ejecución y resultados obtenidos (5).

Deben verificar:

- ✓ Si el producto satisface los requerimientos del usuario, tal y como se describe en las especificación de los requerimientos.
- ✓ Si el producto se comporta tal y como se describe en las especificaciones funcionales del diseño.

3.5.2.1.1 CPR 1 Lectura de bits.

Descripción General: El caso de prueba demuestra si se realiza la lectura del estado de uno o varios bits en la memoria del PLC.

Condiciones de Ejecución:

- Tiene que estar en ejecución el simulador ModScan32.
- Tiene que estar el PLC conectado a la PC por el puerto serie.
- Tiene que descargarse el bitstreams del sistema en el PLC.

Caso #	Clases Válidas	Resultados Esperados	Resultados Obtenidos	Observaciones
1	Se manda a leer el estado (ON/OFF) de un bit determinado.	Debe mostrarse el estado que tiene el bit en memoria.	Se muestra el estado del bit.	En caso de ocurrir un error debe mostrarse el código del mismo.
2	Se manda a leer el estado (ON/OFF) de 4 bits.	Debe mostrarse el estado que tienen los 4 bits en memoria.	Se muestra el estado de los 4 bits.	En caso de ocurrir un error debe mostrarse el código del mismo.
3	Se manda a leer el estado (ON/OFF) de 8 bits.	Debe mostrarse el estado que tienen los 8 bits en memoria.	Se muestra el estado de los 8 bits.	En caso de ocurrir un error debe mostrarse el código del mismo.

Tabla 6 Resultados de las pruebas a la funcionalidad Lectura de bits.

3.5.2.1.2 CPR 2 Lectura de palabras.

Descripción General: El caso de prueba demuestra si se realiza la lectura del estado de una o varias palabras en la memoria del PLC.

Condiciones de Ejecución:

- Tiene que estar en ejecución el simulador ModScan32.
- Tiene que estar el PLC conectado a la PC por el puerto serie.
- Tiene que descargarse el bitstreams del sistema en el PLC.

Caso #	Clases Válidas	Resultados Esperados	Resultados Obtenidos	Observaciones
1	Se manda a leer el estado de una palabra determinada.	Debe mostrarse el estado que tiene la palabra en memoria.	Se muestra el estado de la palabra.	En caso de ocurrir un error debe mostrarse el código del mismo.
2	Se manda a leer el estado de 10 palabras.	Debe mostrarse el estado que tienen las 10 palabras en memoria.	Se muestra el estado de las 10 palabras.	En caso de ocurrir un error debe mostrarse el código del mismo.
3	Se manda a leer el estado de 20 palabras.	Debe mostrarse el estado que tienen las 20 palabras en memoria.	Se muestra el estado de las 20 palabras.	En caso de ocurrir un error debe mostrarse el código del mismo.

Tabla 7 Resultados de las pruebas a la funcionalidad Lectura de palabras.

3.5.2.1.3 CPR 3 Escritura de bits.

Descripción General: El caso de prueba demuestra si se realiza la escritura de uno o varios bits en la memoria del PLC.

Condiciones de Ejecución:

- Tiene que estar en ejecución el simulador ModScan32.
- Tiene que estar el PLC conectado a la PC por el puerto serie.
- Tiene que descargarse el bitstreams del sistema en el PLC.

Caso #	Clases Válidas	Resultados Esperados	Resultados Obtenidos	Observaciones
1	Se manda a escribir el estado de un bit determinado.	El estado del bit debe cambiar y tomar el nuevo valor.	Cambia el estado del bit.	En caso de ocurrir un error debe mostrarse el código del mismo.
2	Se manda a escribir el estado de 5 bits.	El estado de los 5 bits debe cambiar y tomar los nuevos valores.	Cambia el estado de los 5 bits.	En caso de ocurrir un error debe mostrarse el código del mismo.
3	Se manda a escribir el estado de 8 bits.	El estado de los 8 bits debe cambiar y tomar los nuevos valores.	Cambia el estado de los 8 bits.	En caso de ocurrir un error debe mostrarse el código del mismo.

Tabla 8 Resultados de las pruebas a la funcionalidad Escritura de bits.

3.5.2.1.4 CPR 4 Escritura de palabras.

Descripción General: El caso de prueba demuestra si se realiza la escritura de una o varias palabras en la memoria del PLC.

Condiciones de Ejecución:

- Tiene que estar en ejecución el simulador ModScan32.
- Tiene que estar el PLC conectado a la PC por el puerto serie.
- Tiene que descargarse el bitstreams del sistema en el PLC.

Caso #	Clases Válidas	Resultados Esperados	Resultados Obtenidos	Observaciones
1	Se manda a escribir el estado de una palabra determinada.	El estado de la palabra debe cambiar y tomar el nuevo valor.	Cambia el estado de la palabra.	En caso de ocurrir un error debe mostrarse el código del mismo.
2	Se manda a escribir el estado de 2 palabras.	El estado de las 2 palabras debe cambiar y tomar los nuevos valores.	Cambia el estado de las 2 palabras.	En caso de ocurrir un error debe mostrarse el código del mismo.
3	Se manda a escribir el estado de 3 palabras.	El estado de las 3 palabras debe cambiar y tomar los nuevos valores.	Cambia el estado de las 3 palabras.	En caso de ocurrir un error debe mostrarse el código del mismo.

Tabla 9 Resultados de las pruebas a la funcionalidad Escritura de palabras.

3.6. Conclusiones Parciales.

En este capítulo se realizó el diseño del diagrama de despliegue del sistema donde se describen los nodos de procesamiento en tiempo real donde se ejecutará la aplicación y los vínculos entre ellos. Se enunciaron los estándares de codificación utilizados para generar el código del sistema. Se realizaron las pruebas a cada una de las funcionalidades que debe cumplir el sistema.

Conclusiones

- Se realizó un análisis de las tecnologías más usadas en la actualidad, concluyéndose en la utilización de ANSI C como lenguaje de desarrollo y el paquete EDK como herramienta de desarrollo. La aplicación se desarrolló siguiendo los pasos de la metodología RUP y se utilizó UML para el modelado. Se realizó el diseño del sistema a través del diagrama de clases del diseño y el diagrama de despliegue.
- Se llevó a cabo la validación del sistema mediante la realización de pruebas a cada una de las funcionalidades que el sistema debía cumplir.
- Finalmente se desarrollaron 6 de las funciones que ofrece el protocolo Modbus, las cuales componen el módulo de comunicación para el controlador lógico programable.
- Por todo lo anterior se concluye que el objetivo propuesto para la presente investigación se cumplió satisfactoriamente.

Recomendaciones

Teniendo en cuenta las experiencias obtenidas en el desarrollo de este trabajo y los resultados obtenidos se recomienda:

- Continuar el desarrollo del protocolo Modbus RTU con la implementación de las restantes funciones Modbus que no han sido abordadas en este trabajo.

Referencias Bibliográficas

1. **Rivas Torrente, Ronni y López Estrada, Alain.** *Aplicación para la Gestión de la información del Puesto de Mando Informático de los Centros de Diagnóstico Integrales en Venezuela.* Ciudad de la Habana : s.n., 2008. Tesis.

2. **Ramírez, Despaigne, Maikel.** *Controlador Lógico Programable basado en hardware reconfigurable.* Habana : s.n., 2009. Tesis.

3. **Pérez Ruizcalderón, Felix Carlos.** *Implementación de un servidor web sobre hardware reconfigurable.* Pinar del Río : s.n., 2009. Tesis.

4. **Molinari, Ing. Norberto.** *Curso sobre Controladores Lógicos(PLC).* 2010. pdf.

5. **Ávila Gonzalez, Rolando.** *Componente para visualizar curvas de registro de pozos.* Ciudad de la Habana : s.n., 2010. Tesis.

6. Visual Paradigm. *Visual Paradigm.* [En línea] 2004. [Citado el: 20 de enero de 2010.] http://www.visual-paradigm.com/product/vpuml/.

7. Taringa. *Taringa.* [En línea] [Citado el: 15 de noviembre de 2010.] http://www.taringa.net/posts/info/6858866/Herramientas-CASE.html.

8. Scribd. *Puerto Serie.* [En línea] 31 de julio de 2008. [Citado el: 5 de febrero de 2011.] http://www.scribd.com/doc/8256105/17/Puerto-serie.

9. slideshare. *Puertos De Comunicacion - Presentation Transcript.* [En línea] 2011. [Citado el: 13 de junio de 2011.] http://www.slideshare.net/cristian_tenesaca/puertos-de-comunicacion-presentation-892346.

10. monografias.com. *monografias.com.* [En línea] [Citado el: 15 de noviembre de 2010.] http://www.monografias.com/trabajos5/insof/insof.shtml.

11. Itescam. *Firmware.* [En línea] [Citado el: 10 de febrero de 2011.] http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r44721.DOC.

12. Entorno Visual de Aprendizaje. *Introduccion_ a_ la_ disciplina _de _Requisitos.* [En línea] 2010. [Citado el: 20 de enero de 2011.] http://eva.uci.cu/mod/resource/view.php?id=34734.

13. Entorno Visual de Aprendizaje. *Modelo del Dominio.doc.* [En línea] 2010. [Citado el: 20 de enero de 2011.] http://eva.uci.cu/mod/resource/view.php?id=21011.

14. Entorno Visual de Aprendiza. *Material_de_caja_b_y_caja_n.* [En línea] 2010. [Citado el: 20 de mayo de 2011.] http://eva.uci.cu/mod/resource/view.php?id=14104.

15. Buenas Tareas . *Buenas Tareas.* [En línea] [Citado el: 20 de enero de 2010 .] http://www.buenastareas.com/ensayos/Programacion-y-Algoritmos/2209698.html.

16. Buenas Tareas. *Buenas Tareas.* [En línea] [Citado el: 15 de noviembre de 2010.] http://www.buenastareas.com/ensayos/Desarrollo-De-Sistemas/81259.html.

17. Buenas Tareas. *Buenas Tareas.* [En línea] [Citado el: 8 de noviembre de 2010.] http://www.buenastareas.com/ensayos/Extreme-Programming-Xp/1979129.html.

18. Buenas Tareas. *Buenas Tareas.* [En línea] [Citado el: 6 de noviembre de 2010.] http://www.buenastareas.com/ensayos/Proceso-Unificado-De-Rational/1333981.html.

19. BR Consulting. *BR Consulting.* [En línea] [Citado el: 15 de noviembre de 2010.] http://brconsulting.info/portal/articulos/metodologias-de-desarrollo/extreme-programming---xp.html.

20. **Arias Guerra, yulaine.** *Desarrollo de una Biblioteca de Estructuras de Datos Avanzadas (listas, pilas, colas, tablas hash y DCEL).* Ciudad de la Habana : s.n., 2008. Tesis.

21. **Ramón Antunez, Romanuel y Hernández Montero, Lidisy.** *Desarrollo de una biblioteca de métodos numéricos (BMN), referente al cálculo de raíces de funciones, interpolación y ajuste de curvas y resolución de ecuaciones diferenciales.* Ciudad de la Habana : s.n., 2008. Tesis.

22. National Instruments(NI). *Introducción a la Tecnología FPGA: Los Cinco Beneficios Principales.* [En línea] 2011. [Citado el: 13 de junio de 2011.] http://zone.ni.com/devzone/cda/tut/p/id/8259.

Bibliografía

1. **Rivas Torrente, Ronni y López Estrada, Alain.** *Aplicación para la Gestión de la información del Puesto de Mando Informático de los Centros de Diagnóstico Integrales en Venezuela.* Ciudad de la Habana : s.n., 2008. Tesis.

2. **Ramírez, Despaigne, Maikel.** *Controlador Lógico Programable basado en hardware reconfigurable.* Habana : s.n., 2009. Tesis.

3. **Pérez Ruizcalderón, Felix Carlos.** *Implementación de un servidor web sobre hardware reconfigurable.* Pinar del Río : s.n., 2009. Tesis.

4. **Molinari, Ing. Norberto.** *Curso sobre Controladores Lógicos(PLC).* 2010. pdf.

5. **Ávila Gonzalez, Rolando.** *Componente para visualizar curvas de registro de pozos.* Ciudad de la Habana : s.n., 2010. Tesis.

6. Visual Paradigm. *Visual Paradigm.* [En línea] 2004. [Citado el: 20 de enero de 2010.] http://www.visual-paradigm.com/product/vpuml/.

7. Taringa. *Taringa.* [En línea] [Citado el: 15 de noviembre de 2010.] http://www.taringa.net/posts/info/6858866/Herramientas-CASE.html.

8. Scribd. *Puerto Serie.* [En línea] 31 de julio de 2008. [Citado el: 5 de febrero de 2011.] http://www.scribd.com/doc/8256105/17/Puerto-serie.

9. slideshare. *Puertos De Comunicacion - Presentation Transcript.* [En línea] 2011. [Citado el: 13 de junio de 2011.] http://www.slideshare.net/cristian_tenesaca/puertos-de-comunicacion-presentation-892346.

10. monografias.com. *monografias.com.* [En línea] [Citado el: 15 de noviembre de 2010.] http://www.monografias.com/trabajos5/insof/insof.shtml.

11. Itescam. *Firmware.* [En línea] [Citado el: 10 de febrero de 2011.] http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r44721.DOC.

12. Entorno Visual de Aprendizaje. *Introduccion_ a_ la_ disciplina _de _Requisitos.* [En línea] 2010. [Citado el: 20 de enero de 2011.] http://eva.uci.cu/mod/resource/view.php?id=34734.

13. Entorno Visual de Aprendizaje. *Modelo del Dominio.doc.* [En línea] 2010. [Citado el: 20 de enero de 2011.] http://eva.uci.cu/mod/resource/view.php?id=21011.

14. Entorno Visual de Aprendiza. *Material_de_caja_b_y_caja_n.* [En línea] 2010. [Citado el: 20 de mayo de 2011.] http://eva.uci.cu/mod/resource/view.php?id=14104.

15. Buenas Tareas . *Buenas Tareas.* [En línea] [Citado el: 20 de enero de 2010 .] http://www.buenastareas.com/ensayos/Programacion-y-Algoritmos/2209698.html.

16. Buenas Tareas. *Buenas Tareas.* [En línea] [Citado el: 15 de noviembre de 2010.] http://www.buenastareas.com/ensayos/Desarrollo-De-Sistemas/81259.html.

17. Buenas Tareas. *Buenas Tareas.* [En línea] [Citado el: 8 de noviembre de 2010.] http://www.buenastareas.com/ensayos/Extreme-Programming-Xp/1979129.html.

18. Buenas Tareas. *Buenas Tareas.* [En línea] [Citado el: 6 de noviembre de 2010.] http://www.buenastareas.com/ensayos/Proceso-Unificado-De-Rational/1333981.html.

19. BR Consulting. *BR Consulting.* [En línea] [Citado el: 15 de noviembre de 2010.] http://brconsulting.info/portal/articulos/metodologias-de-desarrollo/extreme-programming---xp.html.

20. **Arias Guerra, yulaine.** *Desarrollo de una Biblioteca de Estructuras de Datos Avanzadas (listas, pilas, colas, tablas hash y DCEL).* Ciudad de la Habana : s.n., 2008. Tesis.

21. **Ramón Antunez, Romanuel y Hernández Montero, Lidisy.** *Desarrollo de una biblioteca de métodos numéricos (BMN), referente al cálculo de raíces de funciones, interpolación y ajuste de curvas y resolución de ecuaciones diferenciales.* Ciudad de la Habana : s.n., 2008. Tesis.

22. **Khazmou, Ing. Samira.** Scribd. *Scribd.* [En línea] 2011. [Citado el: 6 de noviembre de 2010.] http://www.scribd.com/doc/1020505/GuiaUML.

23. *Curso de iniciación C.* pueden accerder al curso a través de la siguiente dirección: http://melca.com.ar/UTN/archivos/Curso%20de%20iniciacion%20al%20lenguaje%20C.pdf.

24. *PRACTICA 9. PROCEDIMIENTOS. PARÁMETROS POR VARIABLE O POR REFERENCIA.* pueden acceder al documento a través de la siguiente dirección: http://www.uhu.es/04004/material/Practica9_Curso0809.pdf .

25. **Salas, Angel.** *CURSO DE LENGUAJE C.* Zaragoza : s.n., 1991.

26. **Apoyo a Investigación C.P.D. y Servicios Informáticos U.C.M.** *CURSO BASICO DE PROGRAMACION EN C.*

Glosario de Términos

Autómata: es un sistema secuencial, aunque en ocasiones la palabra es utilizada también para referirse a un robot. Puede definirse como un equipo electrónico programable en lenguaje no informático y diseñado para controlar, en tiempo real y en ambiente industrial, procesos secuenciales.

Automatización: realización de una combinación específica de acciones por una máquina, sin la ayuda de personas.

Bitstreams: flujo de bits, término que se utiliza para describir los datos de configuración que se cargarán en el FPGA.

CamelCase: consiste en escribir frases o palabras compuestas, eliminando los espacios intermedios y poniendo en minúscula la primera letra y en mayúscula las demás primeras letras de cada palabra contigua.

Cautín: instrumento que se emplea para soldar con estaño.

Chip: pieza de silicio pequeña y con forma cuadrada o rectangular en cuyo interior hay un circuito integrado con millones de componentes; generalmente se combina con otros elementos para formar un sistema más complejo, como un ordenador.

EDK (Embedded Development Kit): kit de desarrollo embebido.

Electrónica: es el campo de la ingeniería y de la física aplicada relativo al diseño y aplicación de dispositivos, por lo general circuitos electrónicos, cuyo funcionamiento depende del flujo de electrones para la generación, transmisión, recepción, almacenamiento de información, entre otros.

E-mail (electronic email): correo electrónico, es un método para crear, enviar y recibir mensajes a través de sistemas de comunicación electrónica.

Enlace Multipunto: permite establecer áreas de cobertura de gran capacidad para enlazar diferentes puntos remotos hacia una central para implementar redes de datos voz y video.

Enrutador: es un dispositivo de hardware para interconexión de red de ordenadores que opera en la capa tres .Un enrutador es un dispositivo para la interconexión de redes informáticas que permite asegurar el enrutamiento de paquetes entre redes o determinar la ruta que debe tomar el paquete de datos.

Ethernet: es un estándar de transmisión de datos para redes de área local.

FPGA: campos de matrices programables.

FTP (File Transfer Protocol): protocolo para intercambiar archivos en Internet.

Hardware: es la parte física de un computador y más ampliamente de cualquier dispositivo electrónico.

IDE: Entorno integrado de desarrollo.

Interfaz: es la conexión entre dos ordenadores o máquinas de cualquier tipo dando una comunicación entre distintos niveles.

Memoria RAM (Random Access Memory): memoria de acceso aleatorio o memoria de lectura - escritura.

Memoria ROM (Read Only Memory): memoria de sólo lectura.

MicroBlaze: procesador de 32 bits de núcleo blando.

Microcontrolador: es un circuito integrado que incluye en su interior las tres unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada y salida.

Microprocesador: es la parte de la computadora diseñada para llevar acabo o ejecutar los programas. Este viene siendo el cerebro de la computadora, el motor, el corazón de esta

máquina. Este ejecuta instrucciones que se le dan a la computadora a muy bajo nivel haciendo operaciones lógicas simples, como sumar, restar, multiplicar y dividir.

Motherboard: placa base o tarjeta madre**,** es una placa de circuito impreso a la que se conectan los componentes que constituyen la computadora u ordenador.

Periférico: cualquier equipo electrónico susceptible de ser conectado a un ordenador mediante una de sus interfaces de entrada/salida (puerto serial, puerto paralelo, etc.), la mayoría de las veces a través de un conector.

Plataforma: en informática es precisamente el principio, en el cual se constituye un hardware, sobre el cual un software puede ejecutarse/desarrollarse.

SMTP (Simple Mail Transfer Protocol): protocolo simple de transferencia de correo.

Soft-Core: núcleo blando.

Software: conjunto de instrucciones escritas en un determinado lenguaje, que dirigen a un ordenador para la ejecución de una serie de operaciones, con el objetivo de resolver un problema que se ha definido previamente.

Telnet: es un protocolo de Internet estándar que permite conectar terminales y aplicaciones en Internet.

UDP (User Datagram Protocol): protocolo de datagrama de usuario.

Unidad Terminal Remota (UTR): define a un dispositivo basado en microprocesadores, el cual permite obtener señales independientes de los procesos y enviar la información a un sitio remoto donde se procese.

XPS: Xilinx Platform Studio.