ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN SERVIDOR DE DATOS INDUSTRIAL CON PROTOCOLO MODBUS TCP PARA LOS 8 CÓDIGOS DE FUNCIÓN BÁSICOS

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y CONTROL

VIOLETA ABIGAIL MALDONADO REINOSO

DIRECTORA: DRA.- ING. SILVANA DEL PILAR GAMBOA BENÍTEZ

CODIRECTORA: MBA. ANA VERÓNICA RODAS BENALCÁZAR

Quito, marzo 2021

AVAL

Certifico que el presente trabajo fue desarrollado por Violeta Abigail Maldonado Reinoso, bajo nuestra supervisión.

NOMBRE DIRECTOR

Dra.- Ing. Silvana del Pilar Gamboa Benítez

NOMBRE CODIRECTOR

MBA. Ana Verónica Rodas Benalcázar

DECLARACIÓN DE AUTORÍA

Yo, Violeta Abigail Maldonado Reinoso, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

_____

VIOLETA ABIGIAL MALDONADO REINOSO

DEDICATORIA

Con mucho cariño dedico este trabajo.

A mis padres, María del Carmen Reinoso y Víctor Manuel Maldonado.

A mis hermanas, Krupskaya y Gaby.

AGRADECIMIENTO

Me gustaría agradecer en este espacio a todas las personas que me brindaron su apoyo y consejos no solo durante la realización de este trabajo si no también durante el tiempo en la universidad, ya que gracias a ellas he alcanzado esta meta. Les agradezco inmensamente.

A mis padres, quienes me han guiado, aconsejado y dado su apoyo. Gracias a ellos, me he convertido en la persona que soy. Agradezco principalmente a mi madre por los desvelos, las madrugadas, su apoyo y cariño.

A mis hermanas, por su apoyo incondicional, los ánimos brindados, los consejos impartidos y ser un gran soporte en mi vida.

A mis amigos del prepo 611 y los que conocí durante toda la carrera, con quienes compartí grandes momentos durante toda la vida universitaria y de quienes también aprendí muchas cosas tanto en el ámbito académico como personal. Gracias por sus ánimos y consejos. Espero tengan grandes éxitos.

A mis profesores, quienes a lo largo de la carrera me impartieron su conocimiento y contribuyeron con mi instrucción profesional.

Principalmente agradezco a mi directora de tesis Dra. Silvana Gamboa, por su guía, tiempo y las revisiones realizadas durante la elaboración del proyecto, así como también agradezco a mi codirectora MBA. Ana Rodas, por su tiempo y observaciones del trabajo realizado.

A la Escuela Politécnica Nacional, lugar donde pase varios años, aprendí muchas cosas y conocí a grandiosas personas.

ÍNDICE DE CONTENIDO

RESUMEN

En la industria existen diversos drivers y software para la implementación de sistemas de control supervisorio, que trabajan con diferentes protocolos de comunicación, como el protocolo Modbus. Sin embargo, una desventaja de estos softwares comerciales es el alto costo en la obtención de licencias de los programas, convirtiéndose en un limitante para su adquisición por parte de las PYMES.

El presente trabajo desarrolla un servidor de datos industrial para el protocolo Modbus TCP, usando el software libre Java para el servidor y MySQL como gestor de base de datos. Se implementan los códigos de función básicos del protocolo Modbus para realizar la comunicación entre el servidor y el dispositivo industrial. De esta manera, al usar software libre, las PYMES tienen una alternativa de menor costo que los softwares comerciales y un acceso al código fuente del servidor desarrollado para futuras modificaciones o mejoras.

Así mismo, se desarrolló una interfaz para el servidor de datos industrial con dos modos de funcionamiento. El primer modo permite probar individualmente los códigos de función, mientras el segundo modo realiza una comunicación automática y almacena los datos de la comunicación en una base de datos, misma que funciona como un nexo entre el servidor desarrollado y un HMI en una aplicación de Windows. El servidor permite la implementación de aplicaciones de control supervisorio.

Las diferentes pruebas realizadas validaron el funcionamiento del servidor desarrollado para los dos modos de funcionamiento.

PALABRAS CLAVE: Comunicación, PLC, MySQL, Java, Modbus TCP, InTouch

ABSTRACT

In the industry there are many communication drivers and software for implementing supervisory control systems, which can connect with different industrial communication protocols such as, Modbus. However, a disadvantage of these commercials software is the high cost of obtaining their licenses programs, becoming a limitation for their acquisition by SMEs.

This work develops an industrial data server for Modbus TCP protocol, using Java free software by the server and MySQL as the database manager. The basic Modbus function codes are implemented to carry out communication between the server and the industrial devices. In this way, when using free software, SMEs have a lower-cost alternative to commercial software and access to the source code of the developed server for future modifications or improvements.

Likewise, an interface was developed for the industrial data server with two operating modes. First mode allows testing every function code individually. While the second mode performs automatic communication with field devices and acquired data is stored in a database, which works as link between the developed server and an HMI application implement in Windows. These server characteristics allow the implementation of a supervisory control system.

The different tests carried out validated the operation of the server developed for the two operation modes.

KEYWORDS: communication, PLC, MySQL, Java, Modbus TCP, InTouch

1. INTRODUCCIÓN

En los últimos años se ha realizado una integración de tecnologías de información y comunicación (TIC), revolucionando diversas áreas, entre ellas la industria, a través de modernos sistemas de automatización de procesos basados en nuevas tecnologías, mismos que además tienden a la estandarización de los sistemas de automatización de procesos[1]. A través de éstos se ha obtenido beneficios en el rendimiento de los procesos en los cuales se han incluido estos sistemas. Las redes industriales son un importante componente de la automatización del control de procesos industriales, siendo Industrial Ethernet uno de los protocolos de comunicación más empleados debido a sus capas de hardware y software[2]. Adicionalmente, una de las partes importantes de la automatización es el control supervisorio del proceso, para lo cual se requiere la comunicación entre los equipos industriales con sistemas informáticos, en los cuales se ejecuten programas o aplicaciones de software que realizan estas tareas. No obstante la inversión inicial del software de ingeniería es alta, dentro de esta inversión se encuentran los drivers de comunicación o servidor de datos industrial para aplicaciones de Windows, siendo este un limitante para la implementación de sistemas de control industrial como el SCADA, sobre todo en empresas con baja capacidad de inversión.

Una manera de reducir los costos de software de ingeniería es mediante la utilización de software libre, cuyo uso se ha ido incrementando en los últimos años, debido a los beneficios en reducción de costos de desarrollo al poseer una licencia gratuita, así como la independencia del proveedor y la posibilidad de adaptación.[3] En consecuencia esto ha permitido la popularización del software libre y que el mismo sea accesible a un mayor número de personas y sectores.

En la industria existen diversos protocolos de comunicación, sin embargo uno de los más utilizados es el protocolo MODBUS, mismo que tiene disponible la información del protocolo al público general y es sustentado por fabricantes de equipos industriales. Este protocolo tiene una versión para Ethernet conocido como Modbus TCP, el cual en la actualidad es uno de los más utilizados dentro del control de procesos y automatización de la industria, debido al bajo costo en los controladores Ethernet respecto a otros controladores y su velocidad de transmisión[4][5]. Además, este protocolo se encuentra estandarizado a través de sus diferentes códigos de función que permiten el envío y recepción de datos.

En el presente Proyecto Técnico, se desarrolla un servidor de datos industrial para el protocolo MODBUS TCP usando el software libre JAVA. Para garantizar la conexión del

driver desarrollado con cualquier dispositivo comercial que utiliza este protocolo, se implementan los códigos de función básicos establecidos según la normativa de MODBUS. Adicionalmente, se implementa una base de datos para almacenar los datos adquiridos a través del servidor desarrollado, permitiendo una posterior lectura, escritura y acceso desde un HMI implementado en un software industrial comercial como es el caso de InTouch. Finalmente, se realizará la comparación entre el servidor desarrollado y el servidor de datos comercial DASMBTCP.

1.1 Objetivos

El objetivo general de este Proyecto Técnico es:

- Desarrollar un servidor de datos industrial con protocolo MODBUS TCP para los 8 códigos de función básicos.

Los objetivos específicos de este Proyecto Técnico son:

- Estudiar la normativa del protocolo MODBUS TCP y determinar la estructura de cada uno sus principales códigos de función (01, 02, 03, 04, 05, 06, 15 y 16) y cuáles son los parámetros que pueden ser ingresados por el usuario.
- Desarrollar el servidor de datos industrial MODBUS TCP utilizando JAVA.
- Desarrollar las herramientas de software necesarias para almacenar los datos adquiridos en una base de datos relacional en MySQL.
- Desarrollar un interface que permita al usuario manejar el driver de comunicación.
- Validar la comunicación entre un PLC, el driver realizado, la base de datos e INTOUCH, comparando su funcionamiento con un software comercial.

1.2 Alcance

Para el presente Proyecto Técnico, se estudiará los métodos que se pueden utilizar para la conexión TCP entre dos dispositivos, además se determinará de acuerdo a la normativa del protocolo MODBUS TCP, la estructura de la trama de datos para cada uno de los principales códigos de función de este protocolo los cuales son: 01, 02, 03, 04, 05, 06, 15 y 16, y se determinará los parámetros que pueden ser ingresados por el usuario como parte de la configuración.

Se desarrollará el servidor de datos en el software libre JAVA, donde se implementará una interfaz para configuración del servidor de datos, permitiendo se ingresen los parámetros de configuración del servidor de datos propuesto.

Se realizará las pruebas del funcionamiento del servidor de datos con dispositivos industriales los cuales son: PLC con puerto Ethernet embebido y PLC a través de Gateway. Para confirmar el funcionamiento se comparará los datos visualizados en el PLC con los datos del servidor de datos.

Se diseñará una base de datos en MySQL misma que se comunicará con el servidor de datos realizado en el software libre JAVA. En la base de datos se guardará la dirección y el valor que ha sido escrito o leído.

Se establecerá diferentes comandos del lenguaje SQL para realizar la conexión, lectura y escritura de datos con la base de datos, además de eliminación de filas en caso de que los datos ingresados no cumplan con los parámetros de la normativa.

Se implementará la comunicación entre la base de datos y el servidor de datos, para que posteriormente se realice la lectura y escritura de datos.

Se realizará una interface HMI en InTouch el cual mantendrá una comunicación con la base de datos y contará con la dirección y valor del dato tanto en escritura como lectura de datos. Para lo cual se realizará la conexión con la base de datos desde InTouch.

Se realizará las pruebas del funcionamiento de las interfaces realizadas y la comunicación del servidor de datos implementado con los diferentes PLCs propuestos. Las pruebas realizadas se probarán tanto desde la interfaz realizada desde el programa InTouch como la interfaz del servidor de datos en JAVA. Se comprobará el correcto funcionamiento al comparar los valores de las interfaces con los correspondientes al PLC.

Se establecerán comparaciones entre el funcionamiento del software desarrollado con el software comercial DASMBTCP de Wonderware con el objetivo de validar el correcto funcionamiento del servidor de datos propuesto y se comprobará que cumplen con las características de conexión y comunicación.

1.3 Marco Teórico

En esta sección se detalla conceptos teóricos sobre los sistemas SCADA y el protocolo Modbus, además de otra información relevante para la comprensión del presente trabajo realizado.

1.3.1 Herramientas computacionales SCADA en el mercado

Uno de los componentes principales para la implementación de sistemas SCADA son los drivers de comunicación los cuales habilitan la comunicación entre los dispositivos industriales (PLCs) y software de sistema SCADA. En el m ercado existen varios

desarrolladores que proporcionan este software para el protocolo Modbus. En la Tabla 1.1 se muestra el detalle y costo de tres desarrolladores:

Tabla 1.1. Drivers de comunicación Modbus comerciales

Desarrollador	Software	Detalle	Licencia
Kepware	Modbus Suite v.6 for KEPServerEX	Permite la comunicación Modbus para los protocolos de comunicación (TCP, RTU, ASCII y Plus)	Cuenta con una versión gratuita de dos horas. El costo de licencia es de $ 475.00 [6]
Matrikon	Servidor MatrikonOPC para Modbus	Contiene drivers para diferentes protocolos Modbus y permite una integración entre los dispositivos de campo y diferentes servicios como una base de datos y HMI. Sistema operativo Windows.	Cuenta con una licencia gratuita por 30 días. Costo de licencia del software por un año es $1200 y del servidor OPC $2300. [7]
Wonderware	MBTCP DAServer	Tiene una comunicación con dispositivos con protocolo Modbus TCP, controladores Comprar 984 y TSX Momentum. Se comunica con clientes que usen comunicación OPC, SuiteLink o DDE. Sistema operativo: Windows.	Cuenta con una versión demo que dura 120 minutos, después se requiere licencia, el costo dependerá del tipo de licencia (estándar o profesional) de acuerdo al tipo de proyecto. [8][9] El costo del servidor más el InTouch de 500 tags versión 2020 tiene un costo aproximado de $3 329.43 + IVA. Se debe adquirir InTouch.

Se debe aclarar que los servidores de datos trabajan junto con otro software para la implementación de sistemas de control supervisorio para PC o también llamados software SCADA, por lo que es necesario también disponer de estos. Dentro de los softwares SCADA propietarios se encuentran Wonderware InTouch de AVEVA, SIMATIC WinCC de Siemens, IGSS de Scheider Electric, Movicon de Progea, entre otros.

El desarrollo de la tecnología y las ventajas que ofrece el software libre como: bajo costo de adquisición, independencia del proveedor, modificar el programa de acuerdo a las necesidades, entre otras, ha permitido que en la actualidad se desarrollen programas denominados SCADA “libres”, los cuales en su mayoría son desarrollados en el sistema operativo Linux, pero también existen desarrollos para Windows. Estos softwares tienen integrados driver de comunicación de acuerdo a los protocolos que manejan. Dentro de

los principales SCADA “libres” se encuentran: Argos, FreeSCADA, Indigo, IntegraXor, Likindoy, Lintouch, Mango M2M, Eclipse OpenSCADA, OScada/OpenSCADA, Proview, PVBrowser, SZARP.[10] De los mencionados anteriormente algunos son compatibles con el protocolo Modbus, como es el caso de Indigo, Likindoy, PVBrowser y OpenSCADA.

En Ecuador algunas empresas ya han utilizado SCADA libres para el control y supervisión de sus procesos, como es el caso de EERSA, la cual utiliza Likindoy para monitorizar relés y medidores en la S/E 1[11], empleando como protocolo de comunicación Modbus TCP.

Un trabajo realizado en la Escuela Politécnica Nacional en el software libre Java [12], realizó la implementación de un SCADA, utilizando los protocolos de comunicación Modbus RTU y Modbus TCP, además de la utilización de un navegador web para mostrar la interface gráfica del sistema SCADA. Sin embargo, a pesar del correcto funcionamiento del proyecto no presta facilidad para su integración con aplicaciones de Windows.

El presente trabajo utilizará como software libre Java y el protocolo de comunicación Modbus TCP con una base de datos en MySQL que, por una parte, puede ser instalada en cualquier máquina para el registro y almacenamiento de los datos adquiridos, y por otra parte es capaz de intercambiar datos con aplicaciones de Windows, en las cuales se implemente un interfaz de operador como por ejemplo InTouch.

1.3.2 Componentes de un sistema SCADA

SCADA es un acrónimo cuyo significado es sistema de control supervisorio y adquisición de datos. El sistema SCADA permite la operación de un proceso de forma remota, proporcionando comunicación entre los sensores, PLCs y diferentes dispositivos de campo con una pantalla de operación, desde la cual se podrá controlar y supervisar el proceso.[10] La comunicación con los equipos se la puede realizar mediante diferentes protocolos de comunicación y la adquisición de los datos se los obtiene en tiempo real. Además una de las funciones del sistema SCADA es almacenar los datos, permitiendo la posibilidad de tener históricos.

En la Figura 1.1 se puede observar los componentes de hardware en un sistema SCADA y a continuación, se detallará los mismos.

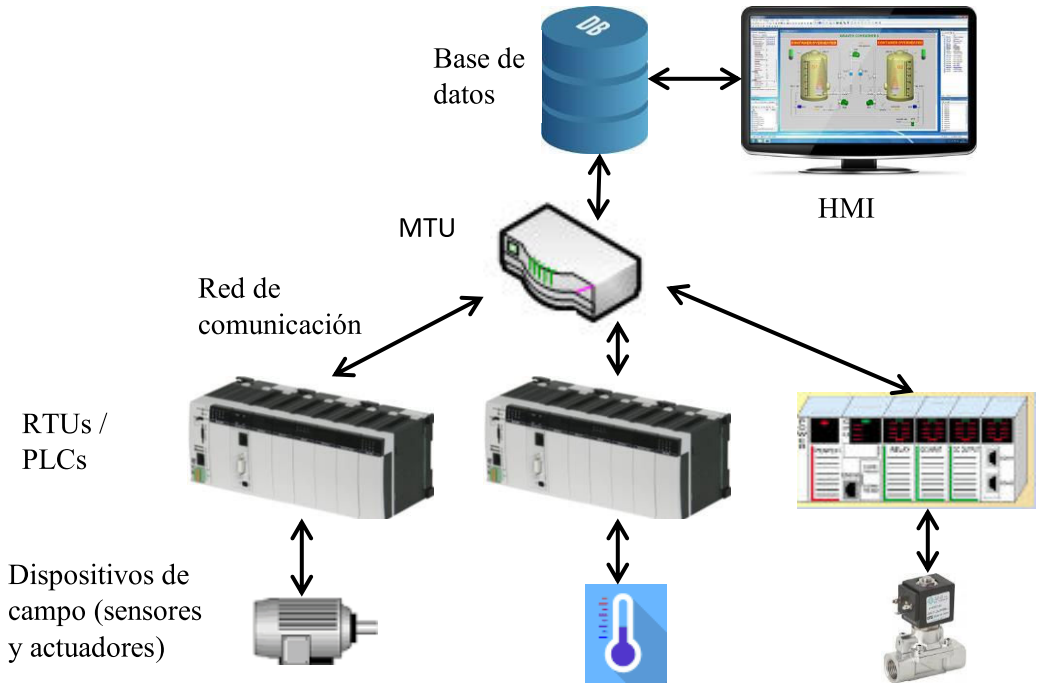


Figura 1.1. Componentes de un sistema SCADA

1.3.2.1 Unidad Terminal Maestra (MTU)

Una MTU es uno de los componentes principales de los sistemas SCADA, ya que se encarga de recoger y almacenar la información del proceso, permitiendo centralizar la información ya sea de uno o varios RTUs, además interactúa con el operador. [10]

Las principales funciones de una MTU son:

- Centralizar la información de las RTUs y tener un almacenamiento temporal de la información.
- Definir acciones sobre el proceso de acuerdo a los mandos del HMI.
- Presentar una interfaz al operador, mostrando información relevante para tener una supervisión y control global del proceso.
- Mantener una puerta de enlace de red, permitiendo comunicación con la red de supervisión o con otros sistemas de control industrial.

Cabe mencionar que una MTU puede ser un PC, un PLC o una combinación de PLC/PC.

En la MTU se encuentran los servidores de datos y el software que permite la comunicación con la RTU o PLC. Como se mencionó anteriormente una de las funciones de una MTU es mantener un almacenamiento temporal de la información por lo cual debe tener una base de datos para realizar este almacenamiento. Además, para poder

comunicarse con las RTUs esta terminal debe manejar el protocolo de comunicación que permita el intercambio de datos con la RTU.

Interfaz Hombre – Máquina (HMI)

Un HMI o Interfaz Hombre – Máquina es un interfaz de usuario en la cual el operador puede visualizar y controlar el comportamiento del proceso. [11] Es decir, los HMI permiten al operador comprender, monitorear y controlar de mejor manera el proceso.

1.3.2.2 Unidad Terminal Remota (RTU)

Una RTU está situada generalmente en sitios remotos estratégicos, en los cuales puedan adquirir los datos de los instrumentos de campo y controlar los elementos finales de control del proceso. Siendo así la principal función de la RTU adquirir datos para luego transmitirlos a la MTU, por esta razón una RTU cuenta con módulos de entradas y salidas analógicas y digitales que permiten tal adquisición de datos.

Actualmente la tendencia es utilizar PLCs como RTUs, lo cual mediante la integración de módulos de entradas/salida e interfaces de comunicación dan mayor capacidad al PLC y puede funcionar como RTU. De esta manera se puede reducir costos en procesos no muy complejos que permitan esta sustitución. [10]

1.3.2.3 Red de comunicación

La red de comunicación tiene la función de transferir la información de los diferentes dispositivos de campo hacia la MTU y a través de ella hacia la plataforma de visualización del sistema SCADA. El bus de campo con el cual se realiza la comunicación puede tener una variedad de protocolos, de acuerdo a lo que requiera el proceso. Actualmente debido a la estandarización, los sistemas SCADA pueden comunicarse con cualquier tipo de bus desde protocolos industriales hasta formas más modernas por ejemplo radio, microondas, redes satelitales, cable, etc.[10]

Para el presente trabajo se utilizará el protocolo Modbus TCP en la red de comunicación entre los PLCs y el servidor de datos de Java.

1.3.3 Protocolo MODBUS

Modbus es un protocolo de comunicación industrial desarrollado por Modicon en 1979, siendo actualmente uno de los protocolos más utilizados a nivel industrial, y considerado un estándar de facto.[13] Es un protocolo de mensajería de capa de aplicación, es decir se encuentra ubicado en el nivel 7 del modelo OSI, además permite una comunicación

cliente – servidor entre dos dispositivos inteligentes conectados a través de un enlace de comunicación. [2]

1.3.3.1 Descripción del protocolo Modbus

Modbus es un protocolo de solicitud/respuesta por lo cual su comunicación es en pares, es decir entre dos dispositivos; uno iniciará una solicitud y otro responderá a esta solicitud.[14]

Este protocolo se lo implementa para conectarse con diferentes tipos de dispositivos como PLC, HMI, Panel de control, dispositivos I/O, o diferentes equipos con protocolo MODBUS, para realizar una operación de forma remota. Dentro del protocolo Modbus los más utilizados, comúnmente son el tipo serial y TCP.

Como se mencionó anteriormente el protocolo Modbus utiliza el modo solicitud/respuesta presentando una arquitectura cliente – servidor y en las implementaciones sobre interfaces seriales se utiliza como mecanismo de acceso al medio el método maestro-esclavo. En este tipo de solicitud, Figura 1.2, el cliente envía un Request o solicitud al servidor, mismo que realiza la acción solicitada por el cliente y después retorna una respuesta o Response, el cual contiene los datos solicitados o un mensaje de confirmación de la acción realizada por el servidor. En caso de no poder realizar la acción solicitada se construye un mensaje de error y el mismo es envía al cliente.

Figura 1.2. Arquitectura Cliente – Servidor

1.3.3.2 Estructura de la trama Modbus

Inicialmente el protocolo Modbus no podía ser dividida en múltiples capas, pero con el tiempo se introdujo diferentes unidades de datos de aplicación, con lo cual se cambió el formato del paquete de datos utilizado inicialmente. Esto llevó a que la trama se dividiera en dos; la unidad de datos del protocolo (PDU) y la unidad de datos de aplicación (ADU). [15] Esta trama se utiliza para cada una de las versiones que tiene el protocolo Modbus. En la figura 1.3 se observa la trama de datos Modbus constituida por el PDU y ADU.

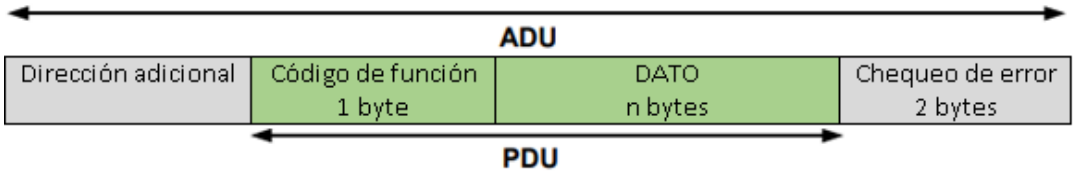


Figura 1.3. ADU del protocolo Modbus

El PDU como se muestra en la Figura 1.3 se encuentra constituido por el código de función, seguido por los datos. La sección de los datos variará para cada código de función, de acuerdo a los parámetros que cada uno requiere para la solicitud y la respuesta. El tamaño del PDU no puede exceder de 253 bytes. Una vez realizada la solicitud, lo primero que comprueba el servidor en el PDU es el código de función, el cual en caso de no corresponder a los que posee el protocolo Modbus envía un mensaje de error.[15]

El ADU corresponde al total de la estructura Modbus, por lo cual está constituido por el PDU, una cabecera o dirección adicional y en algunos casos el chequeo de errores. Existen tres formatos ADU estándares, de acuerdo a los diferentes tipos de protocolo Modbus que se implementa, mismo que son: TCP, RTU (Unidad Terminal Remota) y ASCII. [15]

A continuación se detallará el ADU para cada uno de los protocolos Modbus mencionados.

Modbus TCP

El ADU del protocolo Modbus TCP mostrado en la Figura 1.4 está conformado por dos secciones. La primera es la cabecera del protocolo de aplicación Modbus (MBAP), esta cabecera está conformada por los siguientes campos: identificación de transacción, identificación de protocolo, la longitud de campo y la identificación de unidad. Más adelante, en la sección 1.3.3.5 se detallará la cabecera MBAP.

La segunda sección es el PDU, el cual está conformado por el código de función y los datos.

Identificador de Transacción	Identificador de Protocolo	Longitud	Identificador de Unidad	Código de función	DATO

Figura 1.4. ADU de Modbus TCP

Modbus RTU

El ADU del protocolo Modbus RTU mostrado en la Figura 1.5 está conformado al inicio y al final de la trama por un periodo silencioso, es decir un periodo en el cual no existe comunicación en el bus o también llamado tiempo muerto. De acuerdo al estándar este período varía de acuerdo a la velocidad de transmisión, sin embargo el tiempo mínimo es de aproximadamente 2 ms.[15]

Luego del primer periodo silencioso la trama está conformada por: la dirección adicional, el PDU Modbus y finalmente por una verificación de redundancia cíclica o también llamada CRC (Cyclic Redundancy Check). En la dirección adicional se coloca la dirección del esclavo a la cual está dirigido la solicitud, cabe mencionar que si se coloca el valor de 0 el mensaje se enviará a todos los esclavos que se encuentren conectados, es decir corresponde a la dirección de broadcast.

Arranque (silencio 3.5 ch)	Dirección adicional	Código de función	DATO	CRC	Final (silencio 3.5 ch)

Figura 1.5. ADU de Modbus RTU

Modbus ASCII

En la figura 1.6 se puede observar la estructura del ADU para el protocolo Modbus ASCII, inicia su trama con el carácter "**:**", luego está constituido por la dirección del dispositivo al que se realizará la solicitud, el PDU Modbus, una sección para detectar errores que usa la técnica de verificación de redundancia longitudinal o LRC (Longitudinal Redundant Check) y finaliza la estructura con un retorno de carro o CR (Carriage Return) y una línea de alimentación o LF (Line Feed).

Cabe menciona que con este protocolo todos los datos son enviados como caracteres hexadecimales en ASCII.

0x3A ":"	Dirección	Código de función	DATO	LRC	0X0D CR	0X0A LF

Figura 1.6. ADU de Modbus ASCII

1.3.3.3 Tipos de datos

Los datos en el protocolo Modbus son almacenados en bloques de memoria, divididos en 4 tipos de datos, los cuales son: Coils, Discrete Input o Contacts, Input Register y Holding Register. De estos los dos primeros son variables de tipo Booleano, y los 2 siguientes son de tipo Word. Cada tipo de dato tiene un número identificador o prefijo de dirección, el cual se coloca al inicio de la dirección Modbus y define el bloque de memoria en donde se encontrará el dato, como se observa en la Tabla 1.2.

Tabla 1.2. Datos en el protocolo Modbus

Bloque de memoria	Tipo de dato	Acceso maestro	Prefijo	Dirección
Coil	1 Bit	Lectura / escritura	0	0XXX
Contact (Discrete Input)	1 Bit	Solo lectura	1	1XXX
Input Register	16 bits (Word)	Solo lectura	3	3XXX
Holding Register	16 bits (Word)	Lectura / escritura	4	4XXX

El tamaño de la dirección, es decir su cantidad de dígitos, dependerá del dispositivo Modbus, pudiendo encontrarse dispositivos de 4, 5 y 6 dígitos.

1.3.3.4 Códigos de función

El código de función es una parte importante en la trama, debido a que mediante este se especifica cuál es la acción requerida a realizarse en el dispositivo. Cada uno de los códigos de función tiene una determinada función y estructura, dentro del protocolo Modbus existen códigos de función públicos y códigos de excepción, estos últimos se producen como respuesta ante un error detectado, al no poder realizar la acción solicitada.

Los códigos de función públicos son aquellos que ya están definidos en la normativa, sin embargo se puede definir otros códigos de función. En la Tabla 1.3, se muestran los códigos de función para la lectura y escritura de los diferentes tipos de datos Modbus, adicionalmente hay otros códigos más especializados, sin embargo no son tan utilizados como por ejemplo los códigos del 20 al 24 que son utilizados para lectura y escritura de archivos, entre otras acciones, mientras que los códigos de función del 7 al 12 y el 17 son utilizados para diagnóstico.

Los principales códigos de función públicos básicos del protocolo Modbus son el 01, 02, 03, 04, 05, 06, 15 y 16.

Tabla 1.3. Códigos de función públicos

Código de función		Descripción
Código	Código (hex)	
00	00	Control de estaciones esclavas
01	01	Lectura de Coils
02	02	Lectura de Discrete Inputs
03	03	Lectura de Holding Register
04	04	Lectura de Input Register
05	05	Escritura de un Coil
06	06	Escritura de un Registro
07	07	Lectura estado de excepción (únicamente serial)
08	08	Diagnostico

11	0B	Obtener Com de eventos contador
12	0C	Obtener Com de eventos log
15	0F	Escritura de múltiples Coils
16	10	Escritura de múltiples Registros
17	11	Reporte de esclavo ID
20	14	Lectura de registro de archivo
21	15	Escritura de registro de archivo
22	16	Escribir a registro con máscara
23	17	Leer/ Escribir múltiples registros
24	18	Leer FIFO

Los códigos de excepción son una respuesta ante una solicitud enviada por el cliente que no puede ser realizada por el servidor. Cuando esto ocurre el PDU del Response estará conformado por el código de función y el código de excepción. No obstante, para que el cliente pueda identificar que ocurrió un error en la solicitud se coloca en 1 el MSB del código de función, esto significa que se suma el valor de 80 hexadecimal,[14] mientras que en el campo del código de excepción se colocará el valor de acuerdo a lo detallado en la Tabla 1.4. Cabe mencionar que los códigos de excepción del 1 al 4 ocurren con los códigos de función normalmente utilizados, los demás códigos de excepción se presentan ante una solicitud con comandos de programación o códigos de función más especializados.

Tabla 1.4. Códigos de Excepción Modbus

Código de excepción (hex)	Descripción
01	El código de función recibido no está soportado, o no existe.
02	Solicitud de acceso a una dirección no valida
03	La solicitud tiene datos incorrectos en los parámetros enviados.
04	Error al intentar realizar la acción solicitada
05	Uso especializado con comandos de programación. La solicitud está en proceso, pero tardará un largo tiempo
06	Uso especializado con comandos de programación. El servidor está realizando un proceso de larga duración, por lo cual el cliente debe volver a enviar la solicitud después.
08	Uso especializado con códigos de función 20 y 21. Error en paridad de memoria al leer el archivo
0A	Uso especializado con puerta de enlace. No se pudo asignar una ruta de comunicación interna a la puerta de enlace.
AB	Uso especializado con pasarelas. No se obtuvo respuesta del dispositivo de destino.

En el presente proyecto se realizará la implementación de los 8 códigos de función básicos para el primer modo de funcionamiento, modo maestro, detallados en la Tabla 1.5. Para el segundo modo, modo automático, de los mostrados en la tabla no se implementarán los códigos de función 01, 03 y 15.

Tabla 1.5. Códigos de función públicos básicos

Código de función		Tamaño de dato	Descripción
Código	Código (hex)		
01	01	1 bit	Lectura de Coils
02	02		Lectura de Discrete Inputs
03	03	16 bits	Lectura de Holding Register
04	04		Lectura de Input Register
05	05	1 bit	Escritura de un Coil
06	06	16 bits	Escritura de un Registro
15	0F	1 bit	Escritura de múltiples Coils
16	10	16 bits	Escritura de múltiples Registros

1.3.3.5 Protocolo Modbus TCP

Modbus TCP es una variante del protocolo Modbus, el cual utiliza Ethernet como medio físico, TCP/IP para realizar el transporte de datos, y el Protocolo Modbus como protocolo de aplicación.[4] Este protocolo se caracteriza por ser fácil de implementar en dispositivos que tengan o admitan sockets TCP/IP, como mecanismo para la comunicación de datos.

Con el tiempo el protocolo se ha vuelto muy utilizado debido a ser abierto, simple, de bajo costo y que requiere una mínima cantidad de hardware, por consiguiente en el mercado se encuentra una gran cantidad de dispositivos que se comunican mediante Modbus TCP.[11] La comunicación del protocolo se la realiza por del puerto 502, y a diferencia de Modbus RTU que tiene el campo de dirección adicional, donde se coloca la dirección del esclavo, Modbus TCP usa la dirección IP del dispositivo con el que se va a comunicar.

En la Figura 1.7 se muestra la cabecera MBAP del protocolo Modbus TCP, con un tamaño de 7 bytes, los cuales son: identificador de transacción, identificador de protocolo, longitud e identificador de unidad, detallados a continuación. [4]

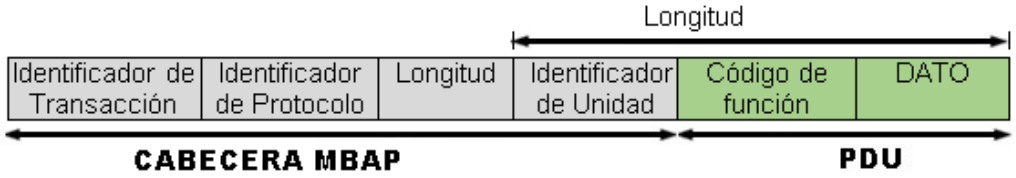


Figura 1.7. Trama Modbus TCP

- Identificador de transacción (2 bytes): Se utiliza para el emparejamiento de transacciones, es decir cuando hay múltiples solicitudes al mismo tiempo. Por ejemplo, el maestro envía solicitudes 1, 2 y 3, y el esclavo responde en el orden 3, 1 y 2. Por lo tanto, mediante este campo se identifican las respuestas de acuerdo a la solicitud 1 con 1, 2 con 2 y 3 con 3. [15]

- Identificador de protocolo (2 bytes): Permite identificar el protocolo que se va a utilizar, para el protocolo Modbus este valor siempre es 0.

- Longitud (2 bytes): Es la cantidad de bytes que posee la trama desde el identificador de unidad hasta el final de la trama. Para el protocolo Modbus TCP, contiene el identificador de unidad y el PDU Modbus. Es decir depende del tamaño de bytes de la trama del código de función, en la Tabla 1.6 se muestra la cantidad de bytes para los códigos de función. Sin embargo, en caso de producirse un código de excepción tiene un valor de 3.

- Identificador de unidad (1 byte): Este campo equivale al campo de dirección adicional o dirección de esclavo en Modbus RTU, se utiliza cuando se realiza una comunicación entre redes TCP y redes seriales. Si solo se realiza comunicación con una red TCP en este campo se coloca por defecto el valor de 0xFF. Cabe mencionar que en una red TCP la dirección del dispositivo se obtiene con la dirección IP por lo cual este parámetro es ignorado.

1.3.3.6 Formato de datos Endianness

Para el almacenamiento de números enteros de más de un byte, se utilizan Endianness lo cual consiste en ordenar los bytes que componen un número. Existen dos tipos de Endianness, Big Endian y Little Endian. El tipo Big Endian significa que el byte más significativo se envía primero, mientras que en el tipo Little Endian el bit menos significativo se transmite primero, en la siguiente figura se detalla los dos tipos.

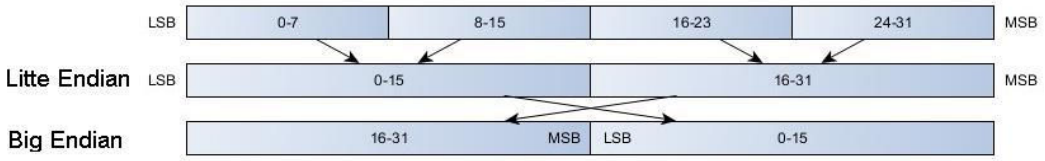


Figura 1.8. Formato de datos Big Endian y Little Endian [15]

Para el presente proyecto y de acuerdo al protocolo Modbus, se utilizará Big Endian para las tramas de datos en el PDU correspondientes a cantidad de registro y dirección Modbus. Sin embargo para los datos a ser leídos o escritos de tipo flotante, al emplear más de 2 bytes para su representación se utilizará los formatos Big Endian o Little Endian dependiendo del dispositivos, ya que a pesar de que la mayoría de dispositivos PLCs utilizan el formato Big Endian, existen algunos que usan el formato Little Endian.

1.3.4 Códigos de función MODBUS TCP

La trama del protocolo Modbus TCP está conformada por la cabecera MBAP y el PDU, Figura 1.7. La cabecera se encuentra conformado por el identificador de transacción e identificador de protocolo con un valor de 0. En la longitud, la cantidad de bytes es de acuerdo a la Tabla 1.6 donde N significa que es variable. Para el identificador de unidad, si se realizara una conexión con un puente serial se colocará la dirección del esclavo, caso contrario se puede colocar 0xFF o cualquier valor, ya que para comunicarse con el dispositivo se usa la dirección IP y se ignora este campo.

Tabla 1.6. Valor de campo longitud para los códigos de función de Modbus TCP

Código de función	Longitud	
	Request	Response
01	6	N
02	6	N
03	6	N
04	6	N
05	6	6
06	6	6
15	N	6
16	N	6

El PDU está conformado por el código de función y los datos, esto se encuentra normalizado por el protocolo Modbus[14], por lo cual para cada código de función tiene una respectiva trama, tanto para el Request como el Response, como se detalla a continuación en esta sección.

1.3.4.1 Código de función FC = 01

El código de función 01 pertenece a la función Leer Coils, misma que puede leer de 1 a 2000 estados de Coils, en la Tabla 1.7 se especifica la cantidad de bytes y rango de valores para el Request y en la Tabla 1.8 para el Response.

Tabla 1.7. Request código de función 01

Campo	Cantidad de bytes	Valor
Código función	1 byte	01
Dirección de inicio	2 bytes	De 0 a 65535 (0XFFFF)
Cantidad	2 bytes	De 1 a 2000

Para el Request, el PDU está conformado por: código de función, el cual tendrá un valor de 01, a continuación se debe colocar la dirección de inicio de la lectura y finalmente la cantidad de registros a leer.

Tabla 1.8. Response código de función 01

Campo	Cantidad de bytes	Valor
Código función	1 byte	01
Byte Count	2 bytes	N
Valor de registro	n bytes	Valores leídos

Para el Response, mostrado en la Tabla 1.8, el código de función tendrá el valor de 1. El Byte Count es el número de registros que se espera de acuerdo a la cantidad que se pidió leer; por lo tanto el valor de N es el valor entero de bytes que se lee, cabe recalcar que cada byte podrá leer hasta 8 Coils. Por ejemplo si lee 10 Coils, N tendrá un valor de 2 en donde los primeros 8 Coils irán en el primer byte y los demás en el siguiente, los espacios restantes en el byte se lee un valor de 0.

1.3.4.2 Código de función FC = 02

El código de función 02 pertenece a la función, leer entradas discretas o lectura de Contacts, esta función permite leer de 1 a 2000 estados, en la Tabla 1.9 se especifica la cantidad de bytes y rango de valores para el Request y en la Tabla 1.10 para el Response.

Tabla 1.9. Request código de función 02

Campo	Cantidad de bytes	Valor
Código función	1 byte	02
Dirección de inicio	2 bytes	De 0 a 65535 (0XFFFF)
Cantidad	2 bytes	De 1 a 2000

Para el Request, el PDU está conformado por: el código de función, la dirección de inicio de la lectura y la cantidad de Contacts a leer, ambos con un tamaño de dos bytes.

Tabla 1.10. PDU de Response código de función 02

Campo	Cantidad de bytes	Valor
Código función	1 byte	02
Byte Count	2 bytes	N
Valor de registro	n bytes	Valores leídos

Para el Response, mostrado en la Tabla 1.10, el PDU está conformado por: el código de función con el valor de 2. El Byte Count es el número de registros que se espera de acuerdo a la cantidad que se pidió leer en el Request y de igual manera que en el código de función 01 el valor de N es la cantidad de byte que contienen los valores leídos.

1.3.4.3 Código de función FC = 03

El código de función 03 pertenece a la función Leer Holding Register, los registros que se van a leer tienen un tamaño de 2 bytes en el caso de ser números enteros, por lo cual se pueden leer hasta 125.

Tabla 1.11. PDU de Request código de función 03

Campo	Cantidad de bytes	Valor
Código función	1 byte	03
Dirección de inicio	2 bytes	De 0 a 65535 (0XFFFF)
Cantidad	2 bytes	De 1 a 125

El PDU Modbus del Request, Tabla 1.11, para el código de función 3 está conformado por el código de función con un valor de 3, la dirección de inicio de lectura y la cantidad de registros que se van a leer.

Tabla 1.12. PDU de Response código de función 03

Campo	Cantidad de bytes	Valor
Código función	1 byte	03
Byte Count	2 bytes	2*N
Valor de registro	n bytes	Valores leídos

El PDU Modbus del Response, Tabla 1.12, consta de: el código de función con el valor de 03, a continuación, el Byte Count o contador de bytes, en donde irá el doble de la cantidad de datos a leer solicitado en el request, ya que cada valor a leer tiene un tamaño de 2 bytes.

1.3.4.4 Código de función FC = 04

El código de función 04 pertenece a la lectura de Input Register, si los registros a leer son enteros tienen un tamaño de 2 bytes para un número, por lo cual se pueden leer hasta la dirección 125. Si el tipo de número a leer en un Input Register es flotante el valor tendrá un tamaño de 4 bytes. Las Tablas 1.13 y 1.14 son para un número entero.

Tabla 1.13. PDU de Request código de función 04

Campo	Cantidad de bytes	Valor
Código función	1 byte	04
Dirección de inicio	2 bytes	De 0 a 65535 (0XFFFF)
Cantidad	2 bytes	De 1 a 125

Para el Request, Tabla 1.13, el PDU está formado por el código de función con un valor de 4, la dirección de inicio de la lectura y la cantidad de registros a leer.

Tabla 1.14. Response código de función 03

Campo	Cantidad de bytes	Valor
Código función	1 byte	04
Byte Count	2 bytes	2*N
Valor de registro	n bytes	Valores leídos

En cambio el PDU del Response, mostrado en la Tabla 1.14, está conformado por: el código de función con un valor de 04, después el Byte Count, donde N es la cantidad de registros a leer de acuerdo a la solicitud del Request, si la lectura es de un número entero este campo es dos veces N, mientras que si es un número flotante es 4 veces N. El último campo es los datos leídos, en caso de ser lectura de números flotantes los valores a leerse tendrán el formato de punto flotante IEEE 754 de simple precisión.

1.3.4.5 Código de función FC = 05

El código de función 05 es la escritura de un solo Coil, el cual solo puede tener dos valores, ON (1) u OFF (0).

Tabla 1.15. Request y Response código de función 05

Campo	Cantidad de bytes	Valor
Código función	1 byte	05
Dirección	2 bytes	De 0 a 65535 (0XFFFF)
Valor	2 bytes	0 o 1 (0x00 o 0XFF00)

El PDU del Request y del Response, mostrado en la Tabla 1.15, es el mismo para ambos casos y está conformado por: el código de función con valor de 5, la dirección en la cual se realizará la escritura y por último el valor a escribirse, como se mencionó antes, el mismo solo puede tener dos valores ON u OFF.

1.3.4.6 Código de función FC = 06

El código de función 06 también llamado Escritura de un Holding Register, se utiliza para la escritura de un registro con un número entero.

Tabla 1.16. PDU del Request y Request código de función 06

Campo	Cantidad de bytes	Valor
Código función	1 byte	06
Dirección	2 bytes	De 0 a 65535 (0XFFFF)
Valor	2 bytes	De 0 a 65535 (0XFFFF)

Igual que en el caso anterior el PDU del Request tiene la misma trama que el Response, por lo cual al servidor retornará la misma trama que se ha enviado. Como se muestra en la Tabla 1.16 el PDU está conformado por: el código de función con un valor de 06, la dirección en donde se escribirá el valor y finalmente el valor a escribirse.

1.3.4.7 Código de función FC = 15

El código de función 15 también llamado Escritura de múltiples Coils, es usado para la escribir varios Coils, por lo tanto los valores que se escribirán tendrán el valor de ON (1) u OFF (0).

Tabla 1.17. PDU del Request código de función 15

Campo	Cantidad de bytes	Valor
Código función	1 byte	15 (0x0F)
Dirección de inicio	2 bytes	De 0 a 65535 (0XFFFF)
Cantidad de Coils	2 bytes	De 1 a 1968 (0X07B0)
Byte Count	1 byte	N
Valores	N x 1byte	valor

En la Tabla 1.17 se muestra el PDU para el Request, donde el valor de N en el Byte Count representa la cantidad de bytes que se van a utilizar para la escritura de los Coils, cada Coil tiene el tamaño de un bit por lo cual al escribir 8 Coils se tiene un byte, el valor de N ira aumentando cuando se completen 8 Coils en un registro. Por ejemplo si se realiza la escritura de tres Coils el valor de N es 1, ya que la cantidad de Coils a escribirse es menor a 8.

Figura 1.9. Conformación de byte del valor a escribir

En la Figura 1.9 se muestra como está constituido el byte en donde se realizará la escritura de los Coils, en el ejemplo se realiza la escritura de tres Coils, el primero con el valor de 0 y los dos siguientes con el valor de 1.

Tabla 1.18. PDU de Response código de función 15

Campo	Cantidad de bytes	Valor
Código función	1 byte	06
Dirección de inicio	2 bytes	De 0 a 65535 (0XFFFF)
Cantidad	2 bytes	De 1 a 1968 (0X07B0)

Para el Response, el PDU mostrado en la Tabla 1.18, está formado por: el código de función con un valor de 06, la dirección de inicio y la cantidad de Coils que se escribieron en el dispositivo.

1.3.4.8 Código de función FC = 16

El código de función 16 denominado también escritura de múltiples registros, se utiliza para la escritura de múltiple Holding Register de manera continua.

Tabla 1.19. PDU del Request código de función 16

Campo	Cantidad de bytes	Valor
Código función	1 byte	16 (0x10)
Dirección de inicio	2 bytes	De 0 a 65535 (0XFFFF)
Cantidad de registros	2 bytes	De 1 a 123 (0X007B)
Byte Count	1 byte	2N
Valores	N x 2 bytes	valor

El PDU del Request, Tabla 1.19, está conformado por: el código de función con un valor de 16, la dirección de inicio, la cantidad de registros a escribirse los cuales pueden ser como máximo 123 datos, un contador de bytes con el doble de la cantidad de registros que se van a escribir en el dispositivo y finalmente los valores a escribirse.

Tabla 1.20. PDU de Response código de función 16

Campo	Cantidad de bytes	Valor
Código función	1 byte	16
Dirección de inicio	2 bytes	De 0 a 65535 (0XFFFF)
Cantidad	2 bytes	De 1 a 123 (0X007B)

El PDU del Response, Tabla 1.20, está conformado por: el código de función con un valor de 16, la dirección de inicio y la cantidad de datos que se escribieron en el dispositivo.

1.3.5 Estándar IEEE754 para representación de números con punto flotante

Para el caso de los códigos de función que trabajan con variables Holding Register o Input Register de tipo flotante se utiliza la representación de números con punto flotante de acuerdo al estándar IEEE754 el cual permite realizar una transformación entre el número flotante y su representación en binario.

En el sistema binario se puede representar números enteros positivos, enteros negativos o números con punto decimal, estos últimos representan a través del estándar IEEE754. En este estándar existen dos tipos de representaciones de acuerdo a su precisión, los cuales son: de precisión simple y de precisión doble, cada uno con una longitud de 32 y 64 bits respectivamente. [16]

Para la representación de los números con punto flotante se utiliza un registro dividido en 3 bloques: bit del signo, exponente y mantisa.

- Signo: Es el primer bloque de tamaño de 1 bit. Si el valor es 0, el número representado es positivo, caso contrario el número representado es negativo.
- Exponente: En este bloque se coloca el número de espacios que se debe recorrer el punto decimal en la mantisa, permitiendo dividir la parte entera del decimal en el número representado.
- Mantisa: Contiene el número completo en binario.

Para obtener el número descrito en un formato de punto flotante se debe realizar los siguientes pasos. [17]

1. Determinar si el número es positivo o negativo mediante el bloque bit de signo.

   1 = Número negativo

   0 = Número positivo

2. Transformar los número del bloque exponente de binario a entero, y a ese valor restar el desplazamiento de acuerdo a la precisión, en el caso de simple precisión es 127 y en el de doble precisión es 1023 (X - desplazamiento).
3. Agregar un 1 en la parte delantera de la mantisa.
4. En la mantisa contar hacia la derecha los espacios correspondientes al valor obtenido en el paso 2, en esta posición se va a colocar la coma o punto flotante.
5. Realizar la conversión del número de binario obtenido en el paso 4.

1.3.5.1 Precisión simple

La representación de punto flotante en precisión simple tiene un tamaño de 32 bits, el exponente se representa con 8 bits con desplazamiento de 127 en el exponente. La mantisa está representada en un sistema SM (24+1,23) lo cual significa sistema signo – magnitud y tiene 24 bits explícitos y uno implícito, de los cuales 23 bits son fraccionarios y 1 bit implícito para determinar el signo del número. [16] En la Figura 1.10 se muestra un ejemplo.

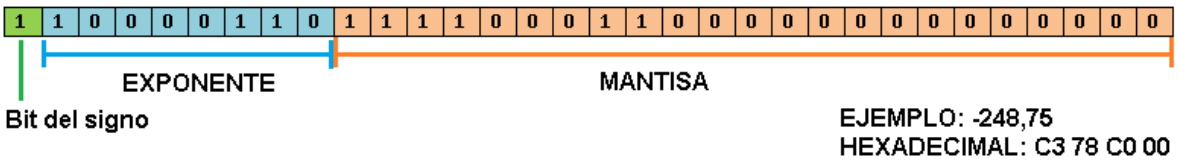


Figura 1.10. Ejemplo formato punto flotante con precisión simple

Para obtener el valor del número representado en la figura, se siguen los pasos detallados anteriormente.

1. El primer bit indica que es un número negativo.
2. El exponente tiene un valor de: 10000110, el cual al transformar a entero se obtiene 134. A continuación, se resta el desplazamiento X = 134 – 127 = 7
3. Se agrega un 1 delante de la mantisa obteniendo el número binario. 11111000011000000000000
4. Se cuentan 7 espacios y se coloca el punto flotante.

   1111000,11000000000000
5. De la parte entera, el número en binario es 11111000 y transformando tiene el valor de 248. De la parte decimal se obtiene 0,11 el cual transformado se obtiene el valor de 0,75. Finalmente uniendo todos los valores se obtiene el número de: -248,75 y en hexadecimal se obtiene el número C478 C000

1.3.5.2 Precisión doble

Figura 1.11. Formato punto flotante precisión doble

De forma similar al formato en precisión simple, se tiene un bit que representa el signo, los siguientes 11 bits pertenecen al exponente y los restantes 52 bits son la mantisa. El exponente tiene un desplazamiento de 1023. La mantisa tiene un bit implícito y 52 bits fraccionarios. [16]

Tabla 1.21. Tipos de estándar IEEE 754 de acuerdo a precisión

Descripción	Precisión Simple	Precisión doble
Bits de signo	1	1
Bits de exponente	8	11
Bits de mantisa	23	52
Total de bits	32	64

En la Tabla 1.21 se observa el tamaño de bits de cada uno de los bloques del formato IEEE 754 los cuales son el bit de signo, el exponente y la mantisa para los formatos de precisión simple y precisión doble.

1.3.6 Software JAVA

Como se mencionó anteriormente en la introducción, el servidor desarrollado en este trabajo se lo realiza en el software libre Java, por lo cual es importante conocer acerca de este software, así como los lenguajes de programación.

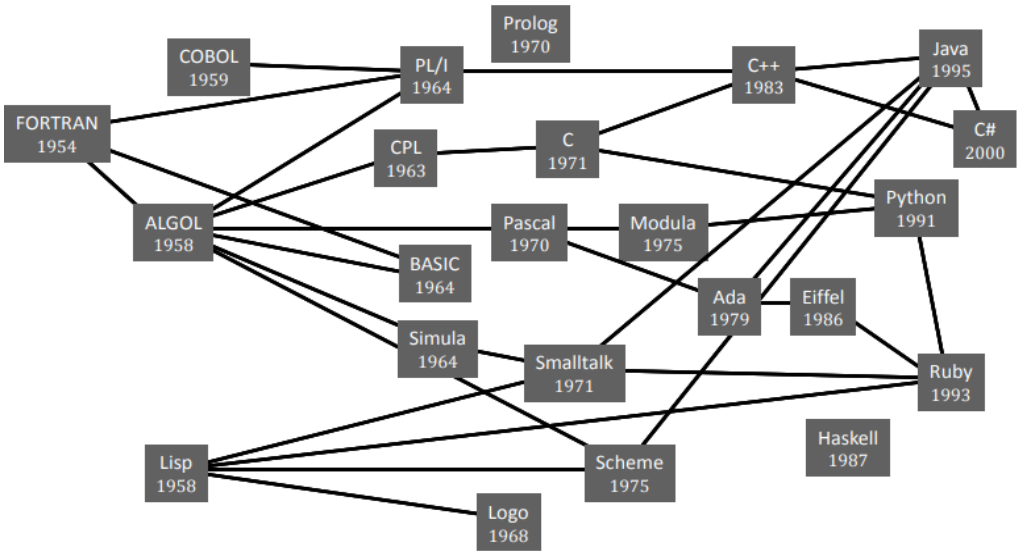


Figura 1.12. Genealogía de lenguajes de alto nivel [18]

Con los años se ha visto una evolución en los lenguajes de programación, los cuales iniciaron como un lenguaje máquina hasta llegar al lenguaje de alto nivel, por lo cual se tienen tres tipos de lenguajes de programación de acuerdo al nivel. El lenguaje de máquina, es el lenguaje que entiende el computador con código binario. Los lenguajes Ensamblador o lenguajes simbólicos, consisten en remplazar el código binario por instrucciones y direcciones con símbolos o mnemónicos. Y el lenguaje de alto nivel es más cercano al lenguaje natural y matemático, además de tener una mayor legibilidad, codificación y eficiencia respecto al lenguaje ensamblador, a través de la historia se han desarrollado diferentes programas y compiladores que utilizan lenguaje de alto nivel. [19] [18] En la Figura 1.12, se muestra una pequeña genealogía de algunos de los programas de alto nivel.

De acuerdo a la manera de resolver un problema los lenguajes de programación se clasifican en 4 tipos de paradigmas: de procedimiento, orientado a objetos, funcional y declarativo. Un paradigma es el modo en que un lenguaje de programación resuelve un problema.[19]

Paradigma de procedimiento: También llamado paradigma imperativo, utiliza objetos pasivos (datos, variables) a los cuales les da una acción o instrucciones y mediante un procedimiento resolver el problema. Algunos lenguajes con este paradigma son Fortran, Pascal, C, Basic, etc. [19]

Paradigma orientado a objetos: Utiliza objetos y clases de objetos, en este caso el objeto tiene un conjunto de acciones definidas. Para resolver un problema se crea una serie de objetos los cuales se utilizan para mediante una secuencia resolver el problema. Algunos lenguajes son C++, Java, Smalltalk, Visual Basic, etc. [19]

Paradigma funcional: Se basa en el concepto de función matemática usando funciones y expresiones. En este caso una función puede llamar otras funciones y tener funciones más simples dentro de la misma, o usar los parámetros de salida como entrada de una función. Los lenguajes con este paradigma son LISP y Scheme. [19]

Paradigma declarativo: Utiliza razonamiento lógico para resolver los problemas. En esta programación se declara una serie de hechos y reglas que se consideran verdaderos, mediante los cuales se da la respuesta de una consulta. [19]

1.3.6.1 Historia de Java

Java es un lenguaje de programación orientada a objetos, lanzada por primera vez en 1995 por Sun Microsystems. El programa fue creado en un proceso por etapas, el cual inició en 1990 para desarrollar un sistema que controle electrodomésticos y permita conexión a redes de ordenadores, creando un sistema operativo eficiente (SunOS) y el lenguaje de desarrollo Oak, precursor de Java, sin embargo debido al costo, el proyecto fracaso. Luego con la aparición de Mosaic y la World Wide Web se distribuyó el lenguaje por la red permitiendo que el lenguaje se popularizara y se diera un mayor uso del mismo, además de añadir varias clases y funcionalidad para TCP/IP. Finalmente se dio el nombre de Java y su lanzamiento a principios de 1995. [20]

Figura 1.13. Java Runtime Environment (JRE) [21]

Una de las ventajas de Java es que se puede ejecutar en diferentes sistemas operativos, ya que es independiente de la plataforma, por lo cual los programas Java son portables. Para la ejecución de los programas compilados se utiliza un entorno de ejecución denominado Java Runtime Environment (JRE). Este permite que el programa Java pueda ejecutarse en diferentes sistemas operativos. [21]

1.3.6.2 Entornos de desarrollo para JAVA

Existen diferentes tipos de entornos de desarrollo para Java, que permiten un entorno de trabajo integrado para facilitar la programación de aplicaciones, la compilación y verificación de los programas. Estos productos o entornos se denominan IDE (Integrated Development Environment). Los entornos de distribución libre son: NetBeans, Eclipse y BlueJ. En el presente proyecto se utilizará el IDE Eclipse.

1.3.6.3 Programación orientada a objetos (POO)

Java es un lenguaje de programación orientada a objetos, por lo cual se debe comprender las características de esta programación.

Esta programación usa el paradigma orientado a objetos, definiendo clases y objetos, a través de estos se organiza el código en unidades o paquetes y mediante la relación de los objetos se puede solucionar un problema.

Un objeto en el mundo real es cualquier persona, animal o cosa, incluso un concepto, el cual tiene determinadas características y acciones. En programación orientada a objetos, a las características se denominan atributos y sus acciones se denominan métodos.

Una clase es un conjunto de objetos con los mismos atributos y métodos, por lo cual a la clase se la podría comparar con la plantilla de un objeto.

La programación orientada a objetos tiene 4 pilares los cuales son Abstracción, Encapsulación, Polimorfismo y Herencia.

Abstracción: Consiste en la identificación y extracción de características esenciales de un objeto, de acuerdo al interés del observador. [22]

Encapsulación: Consiste en limitar el acceso a ciertos métodos o datos. Esto se da de acuerdo a los niveles de ocultamiento, los cuales son: publico (public), cualquiera puede acceder, protegido (protected), solo se puede acceder dentro de la clase y por subclases, privado (private), solo acceden métodos de la propia clase.

Herencia: Permite crear una clase derivada es decir, una clase general puede heredar características a una clase hija, la nueva clase heredará todos los atributos y métodos de

la clase padre y puede además definir nuevos atributos o métodos que no se encontraban en esta. [20]

Polimorfismo: Esta característica ocurre en clases heredadas, y permite que un objeto responda de diferentes formas ante un mismo método. Las clases hijas pueden sobrescribir un método de una clase padre, por lo cual puede actuar diferente al original. En consecuencia el objeto realizará la acción dependiendo si se está llamando al método de la clase padre o de la clase hija. [23]

Además de los pilares se tienen otras características como:

Modularidad: Permite dividir un programa en partes o paquetes que contengan clases y puedan compilarse por separado.

Jerarquía: Estructurar el proceso de acuerdo a niveles.

En el trabajo realizado se utiliza Java para realizar el servidor industrial, en donde se implementa el protocolo Modbus y mediante la utilización de sockets se comunica con un dispositivo industrial, adicionalmente se tiene una comunicación con una base de datos.

1.3.7 Base de datos

En el presente trabajo se utilizará la base de datos para almacenar la información que es enviada desde los dispositivos Modbus al servidor de datos desarrollado y viceversa. Es decir, la base de datos mantendrá un almacenamiento a corto y largo plazo de la información de interés como los datos de configuración del dispositivo industrial e información de los bloques de memoria del protocolo Modbus, explicado en la sección 1.3.3.3, como nombre, dirección Modbus y valor. Estos datos son almacenados tanto de la comunicación entre el servidor desarrollado con el dispositivo industrial como la comunicación con el HMI.

Una base de datos está definida como un lugar en donde se almacenan datos de interés para el usuario con un propósito específico, estos datos están relacionados y organizados en diferentes modos de acuerdo a los requerimientos de la aplicación en la que se usarán los datos.[24]

En el área informática existen programas conocidos como Sistemas Gestores de Base de Datos (SGBD) o también llamados Data Base Management System (DBMS), los cuales permiten al usuario gestionar una base de datos para almacenar datos y posteriormente

acceder a los mismos de una forma rápida y estructurada, además de mantener un registro de los datos.

El lenguaje SQL, el cual es un lenguaje de programación de alto nivel para realizar consultas, permite mediante diferentes tipos de sentencias estructuradas, realizar la acción solicitada en la base de datos. Más adelante se mostrarán algunas de las sentencias que tiene el lenguaje SQL para consulta.

1.3.7.1 Gestor de Base de Datos MySQL

MySQL es un sistema de gestión de base de datos relacional de código abierto, siendo uno de los más utilizados actualmente, y emplea el lenguaje SQL para la gestión de bases de datos.

MySQL inició con la empresa MySQLAB en donde Michael Windenis utilizó mSQL para la conexión de tablas manejando rutinas de bajo nivel, sin embargo, al darse cuenta que no era flexible empleó una interface SQL para las bases de datos y en 1995 la empresa MySQLAB lanzo la primera versión de MySQL, misma que ha ido mejorando en sus versiones posteriores incorporando funciones y otras características. Posteriormente la empresa fue adquirida por Sun MicroSystems en el 2008 y en 2009 fue comprada por Oracle. En la Figura 1.14 se observa una línea de tiempo de MySQL.

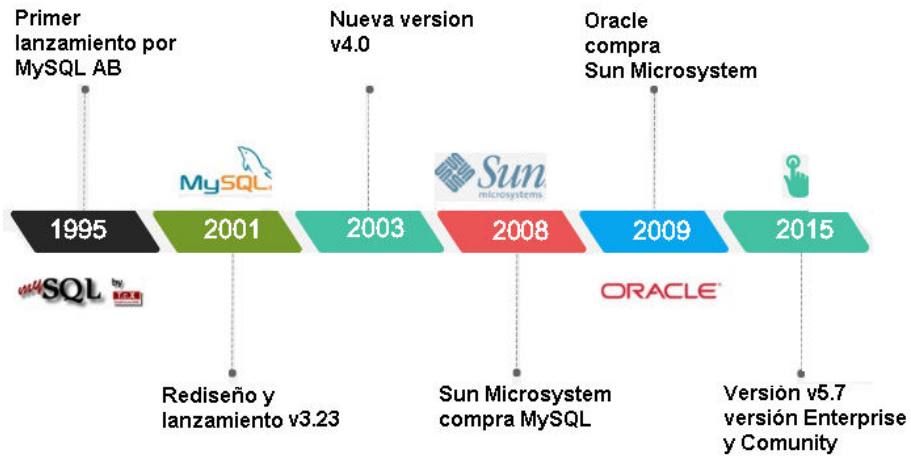


Figura 1.14. Línea de tiempo MySQL

1.3.7.2 Características de MySQL

Este gestor de base de datos, es muy conocido por su facilidad para utilizarlo, a continuación se detallarán sus características principales de acuerdo al Manual de referencia de MySQL 5.0.

- Portabilidad de sistemas, existen varios sistemas operativos que soportan MySQL como: Windows, Linux, Solaris, SunOS 4.x, entre otros. Lo que se requiere para el servidor es que tenga un compilador de C++ y una implementación funcional de subprocesos (threads), para el cliente solo es necesario un compilador C++.

- Uso de multi-threaded (múltiples hilos) a través de threads de kernel, lo cual permite potenciar el multiproceso y se puedan utilizar múltiples CPUs para distribuir el proceso.

- Tiene diversos tipos de columnas como: enteros con/sin signo de 1, 2, 3, 4, y 8 bytes de longitud, Float, Double, Char, Varchar, Text, Blob, Date, Time, DateTime, Timestamp, Year, Set, Enun, y espaciales de OpenGis.

- El lenguaje SQL permite realizar consultas en MySQL, ya sea para la creación de tablas o para trabajar con la información en las mismas. Además mediante los comandos SQL se puede chequear, optimizar y reparar tablas.

- Cuenta con un sistema de privilegio de usuarios y contraseñas, el cual es flexible y seguro. Las contraseñas están cifradas cuando se conectan con un servidor y mediante la asignación de privilegios a los usuarios, se puede limitar lo que se permite a cada uno.

- La conectividad de los clientes al servidor de MySQL se lo puede realizar mediante sockets TCP/IP. En sistemas Windows de la familia NT (NT, XP, 2000 o 2003) se usa named pipes y en sistemas Unix ficheros socket Unix.

- La conexión también se puede dar a través de un conector ODBC (MyODBC) o un conector JDBC para clientes Java.

1.3.7.3 Principales sentencias de SQL

MySQL utiliza algunas de las sentencias SQL que permiten realizar diferentes acciones en las bases de datos y las tablas. A continuación en la Tabla 1.22 se detallan las principales. [25]

Tabla 1.22. Principales sentencias lenguaje SQL

Sentencia	Descripción
SELECT	Se emplea para consulta de datos
INSERT	Se emplea para insertar datos en una tabla
UPDATE	Se emplea para actualizar o modificar datos ya existentes

DELETE	Se utiliza para borrar los datos de una tabla
ORDER BY	Se utiliza para ordenar los resultados consultados en orden ascendente o descendente
DISTINCT	Se utiliza para eliminar los duplicados de las consultas de datos
WHERE	Se utiliza para incluir condiciones en los datos que se van a consultar
CREATE TABLE	Permite crear una nueva tabla
DROP TABLE	Elimina una tabla
TRUNCATE	Elimina todas las filas en una tabla, pero mantiene la tabla
ALTER TABLE	Permite añadir o redefinir las características de una columna

1.3.8 Plataforma para sistemas de control supervisorio Wonderware

En la industria existen softwares comerciales que permiten la comunicación con dispositivos industriales a través de diferentes protocolos de comunicación. Uno de ellos es Wonderware, el cual maneja diferentes protocolos de comunicación como por ejemplo Modbus TCP. Debido a que en el presente trabajo se realiza el desarrollo de un servidor de datos industrial es importante realizar una comparación y evaluar su desempeño con otras aplicaciones de software industrial.

Wonderware es un software industrial que permite la automatización de procesos, siendo uno de los más conocidos, cuenta con un software HMI/SCADA y software de gestión de operaciones en tiempo real lo cual permite la implementación de un sistema SCADA mediante los siguientes paquetes que se ofrecen: Wonderware System Platform y Wonderware InTouch.

En el presente trabajo se realizará un HMI en InTouch para comprobar la conectividad del servidor de datos desarrollado con una aplicación de Windows y se comprará las características del servidor realizado en Java con el DAServer MBTCP de Wonderware.

1.3.8.1 Wonderware System Platform

Es una plataforma que permite conectarse y recoger datos de los diferentes dispositivos de control. Tiene además diferentes programas que permiten el desarrollo de la aplicación utilizando como arquitectura ArchestrA, misma que utiliza la tecnología .NET de Microsoft. Esta plataforma incluye diferente programas que se encuentran interconectados entre sí. [26]

- **Wonderware Historian:** Este programa es una base de datos que permite el almacenamiento y gestión de los datos.
- **Information Server:** Este programa proporciona servicios vía web.

- **Data Adquisition Server**: También denominado DA Server es en donde se encuentran los driver con diferentes protocolos para realizar la comunicación con los dispositivos de campo o dispositivos de control.

  En ArchestrA System Management Console (SMC) se realiza la configuración del driver a utilizar. En la Figura 1.15 se muestra el SMC. En la parte izquierda se despliega los drivers que se pueden utilizar en la plataforma, mientras en la parte derecha la respectiva configuración. En el caso de Modbus TCP se debe configurar DASMBTCP y en Diagnostics se puede observar los datos que han sido enviados y recibidos en el servidor. En la pestaña Device Groups se le asigna un nombre al driver y en Device Items se debe especificar la dirección Modbus de las variables y el nombre.

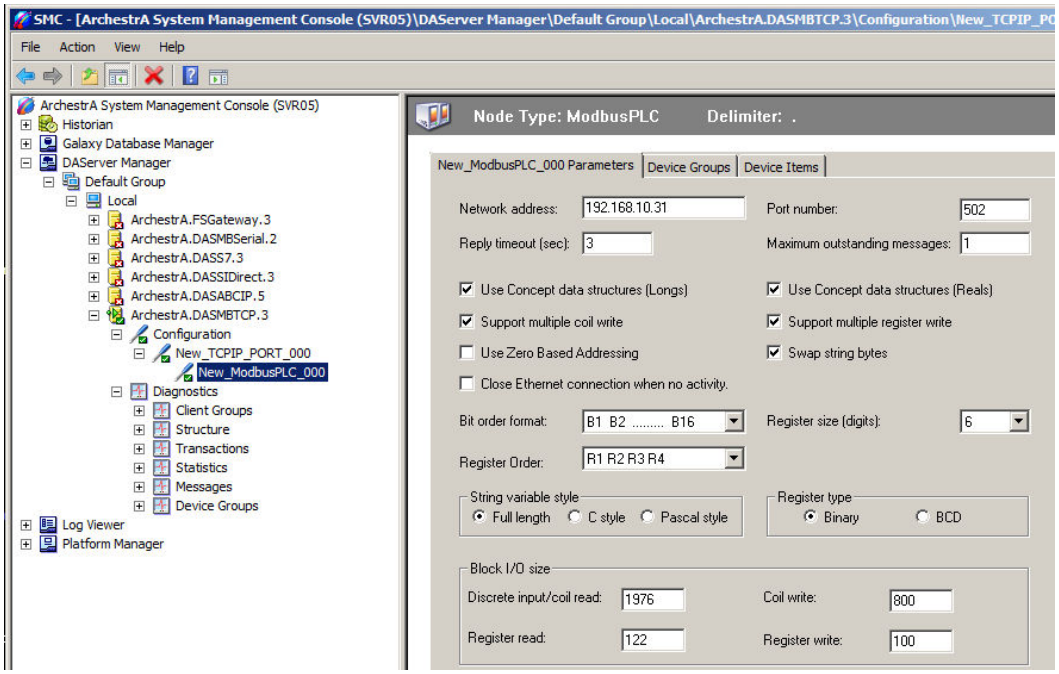


Figura 1.15. SMC con servidor DASMBTCP

MBTCP DAServer es una aplicación de Microsoft Windows el cual actúa como un servidor para el protocolo de comunicación Modbus, permitiendo que aplicaciones de Windows puedan acceder a controladores Modicon a traves de una conexión de Ethernet, incluyendo TSX Premium, TSX Quantum y TSX Momentum y el protocolo Modbus TCP/IP. Adicionalmente permite una conexión a travez de un puente Modbus. A pesar de que este servidor este diseñado para usarse con Wonderware InTouch puede ser utilizado con cualquier aplicación de Microsoft Windows que actue como un cliente DDE, SuiteLink u OPC. [8]

1.3.8.2 Wonderware InTouch

Wonderware InTouch es la herramienta para el desarrollo del interface Hombre – Máquina (HMI) de Wonderware System Platform. Mediante este programa se puede realizar la visualización y control del proceso, además a través de sus múltiples gráficos y funciones que ofrece permite realizar el diseño del HMI del proceso. Esta plataforma cuenta con dos partes, InTouch WindowMaker e InTouch WindowViewer.

- **InTouch WindowViewer:** Es el programa en donde se ejecuta la visualización y permite la interacción de las diferentes ventanas realizadas en InTouch WindowMarker.

- **InTouch WindowMarker:** Este programa permite diseñar y desarrollar el entorno gráfico de la interface Hombre – Máquina del proceso.

  El programa permite desarrollar aplicaciones tipo Stand Alone y Modern, este último tiene incorporado la utilización de Archestra Symbol en caso de que se quiera crear gráficos extras.

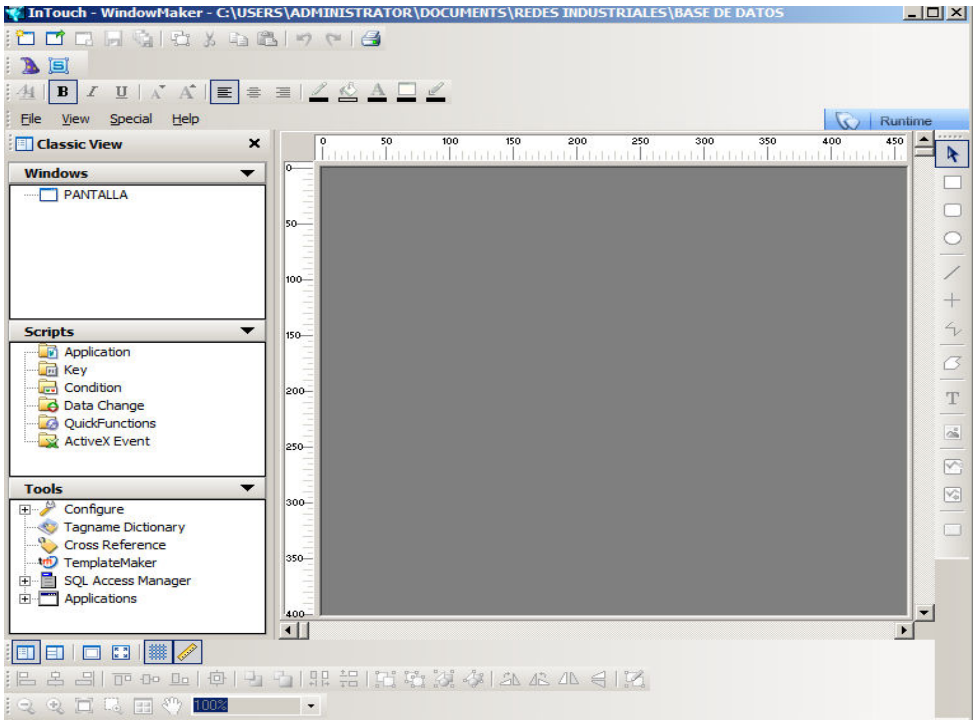


Figura 1.16. Pantalla de inicio InTouch – WindowMaker modo Stand Alone

Este programa cuenta con una serie de componentes para realizar el diseño del HMI, entre estos se encuentran: la creación de ventanas en donde se muestre el HMI, Scripts,

Tagname Dictionary, SQL Access Manager, gráficos, administrador de claves de usuario en caso de necesitarlo, etc.

SQL Access Manager permite la comunicación con bases de datos que manejen el lenguaje SQL, hay dos tipos: Bind List y Table Template. Para este trabajo se utilizará el primero. El Bind List permite asociar un Tagname con una columna de la base de datos, cabe mencionar que al utilizar un Bind List se debe realizar un script, ya sea de nivel de aplicación, ventana o de un objeto, en donde se implementen las sentencias SQL para realizar la conexión con la tabla y para la lectura o escritura de la variable. Adicionalmente se debe configurar en el ODBC los datos de la base de datos que se va a emplear.

En el Tagname Dictionary se realiza la configuración de las variables que se van a utilizar en el HMI. Para la configuración y creación de un nuevo Tagname se despliega una ventana en donde se detallan las características de la variable. En esta ventana de configuración, se debe especificar el tipo de variables, rangos de valores, en caso de ser una variable analógica se pueden colocar alarmas de tipo: alto, bajo, muy alto, muy bajo. Además mediante el Access Name se puede relacionar el Tagname con el driver configurado en el SMC. En la Figura 1.17 se muestra cada una de las partes de la ventana Tagname Dictionary, para la configuración de una variable de tipo real.

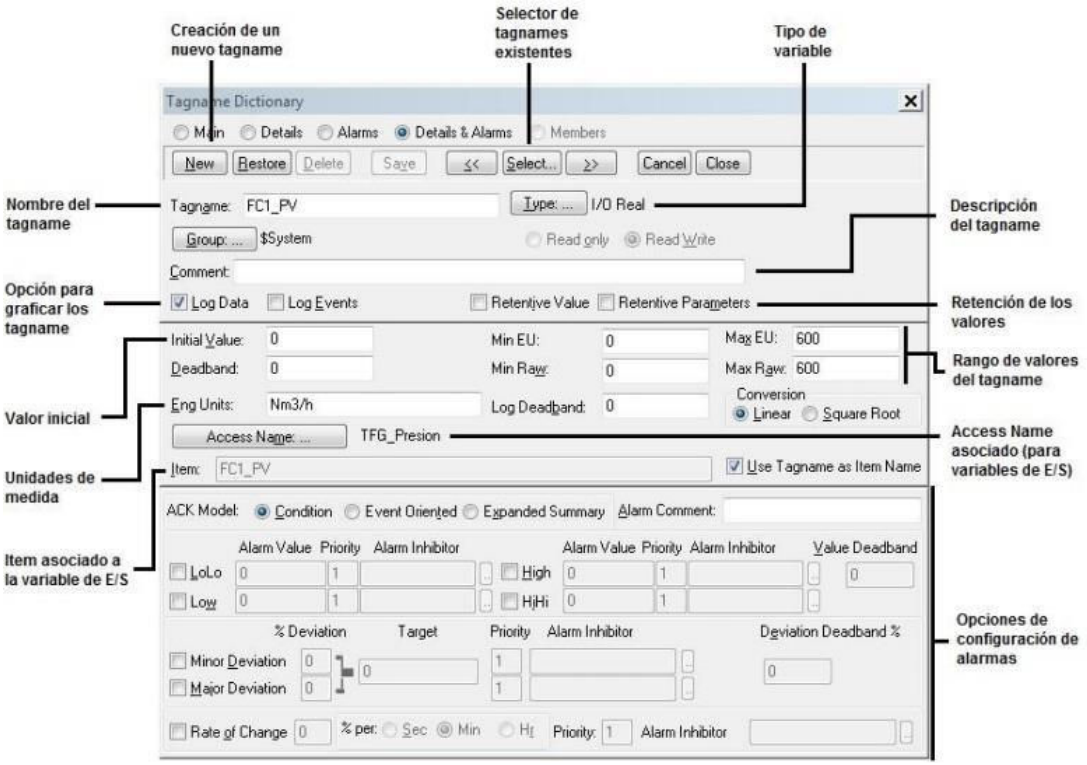


Figura 1.17. Ventana de configuración de tagname tipo Details & Alarms. [26]

2. METODOLOGÍA

En este capítulo se realiza un análisis de la estructura de los diferentes códigos de función básicos, de acuerdo a la normativa del protocolo MODBUS TCP para determinar los parámetros de comunicación que pueden ser configurados por el usuario, así como la estructura del mensajes Modbus, tanto el Request como el Response para los 8 códigos de función básicos.

Además, se realiza el diseño del servidor de datos y su posterior implementación en el software libre JAVA, en el cual se incluirá la conexión con una base de datos en MySQL para el almacenamiento de los datos enviados y recibidos desde los dispositivos industriales a través del protocolo Modbus TCP.

Adicionalmente en este capítulo se realiza el detalle de la conexión de la base de datos en MySQL con una aplicación de Windows, InTouch, y el diseño de la interfaz gráfica del servidor.

2.1. Diseño del driver de comunicación a desarrollar

Antes de realizar la implementación del servidor de datos se debe realizar el diseño de acuerdo a los requerimientos del mismo, los cuales serán detallados en la sección 2.1.1, mientras en la sección 2.1.2 se explicará cómo se cumplirá con tales requerimientos.

2.1.1. Requerimientos

Los requerimientos detallados a continuación permitirán establecer los parámetros que debe cumplir el servidor de datos a desarrollar para lograr un servidor de datos industrial en software libre con protocolo Modbus TCP que cumpla con las necesidades de la industria.

- Los programas a utilizarse deben ser en software libre, para garantizar que no se requiera comprar licencias.
- Empleo de un protocolo de comunicación industrial, el protocolo de comunicación entre el servidor y los dispositivos industriales será mediante MODBUS TCP, para aprovechar las ventajas que tiene el mismo.
- Mostrar a través de una interfaz visual el comportamiento del proceso, por lo cual se requiere una comunicación con aplicaciones en las cuales se pueda realizar un HMI.

- Almacenar y procesar una gran cantidad de datos, esto se realiza mediante el empleo de un sistema de gestor de datos.
- Recolectar y almacenar información en históricos para un posterior análisis, para lo cual se empleará un almacenamiento a largo plazo de las variables en una base de datos.
- Monitoreo y control de las variables de un proceso en tiempo real, para esto se realiza un almacenamiento temporal de las variables en una base de datos y se establece una comunicación continua.

2.1.2. Diseño

De acuerdo a los requerimientos mencionados previamente se realizó el diseño del servidor de datos. El desarrollo del servidor se lo implementó en el software libre Java, y se utiliza el sistema de gestión de base de datos MySQL, el cual también es un software libre. De esta manera gracias a la implementación en software libre se reduce el costo inicial, debido a que no se requieren licencias de los programas.

El servidor de datos a desarrollar manejará un protocolo de comunicación Modbus TCP utilizando los ocho códigos principales de función del protocolo. Para que el usuario pueda probar individualmente cada uno de los códigos de función, el software se dividió en dos modos de funcionamiento. El primero permite probar los ocho códigos de función de manera individual con el dispositivo industrial conectado, como se muestra en la Figura 2.1. Mientras el segundo modo, modo automático, permite realizar una comunicación continua con los dispositivos configurados y un almacenamiento de la información en una base de datos, con lo cual este modo se puede emplear para la implementación de un sistema SCADA, mostrado en la Figura 2.2.

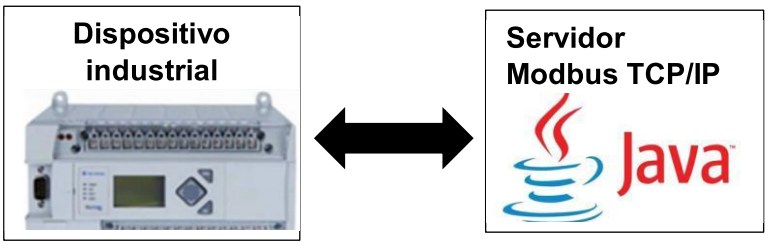


Figura 2.1. Comunicación modo de funcionamiento maestro

Ya que la base de datos requiere un almacenamiento temporal y un almacenamiento a largo plazo para el segundo modo de funcionamiento, la base de datos cuenta con dos tablas para cada dispositivo configurado, una tabla realiza un almacenamiento temporal y

en tiempo real de los registros, mientras que la segunda tabla funciona como un histórico en el cual se almacenan los últimos 1000 datos. Esta base de datos además funciona como un nexo entre el servidor de datos desarrollado y aplicaciones de Windows, permitiendo por ejemplo, la interconexión del servidor con un HMI basado en Windows, mismo que puede comunicarse con la base de datos a través de sentencias SQL. La comunicación de este modo se muestra en la Figura 2.2.

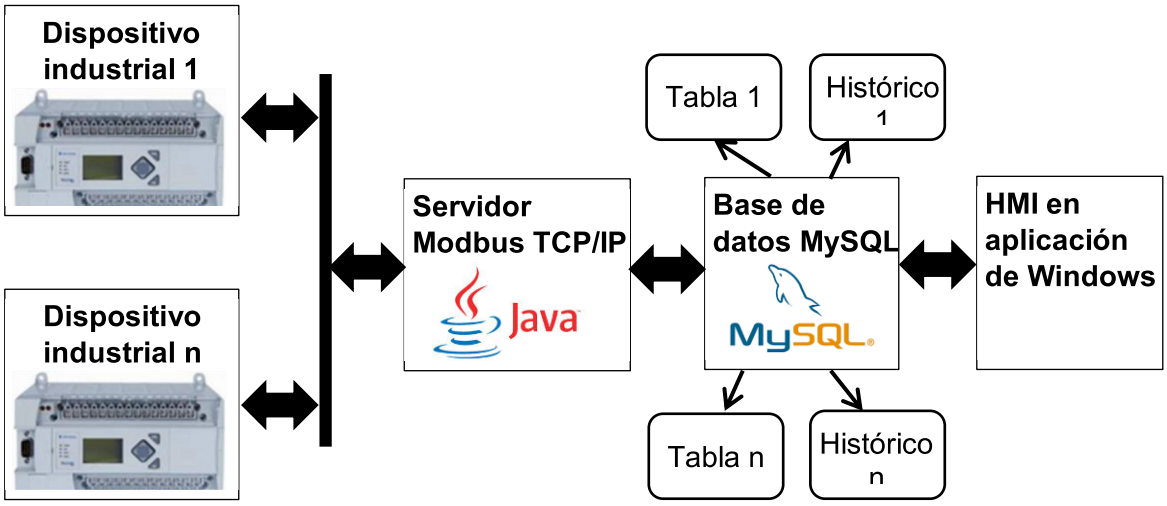


Figura 2.2. Comunicación modo de funcionamiento automático

2.1.3. Clases a implementar

Java utiliza clases para facilitar la programación, las cuales se relacionan entre sí. Para el presente trabajo se realizará una clase para cada uno de los ocho códigos de función a implementar, una clase para cada ventana de la interface del servidor, entre otras. Mediante diagramas de clases se detallará en cada bloque, el nombre de la clase, las propiedades y los métodos. La Figura 2.3 muestra las clases para las ventanas de la interface, por lo cual son clases tipo Jframe. La Figura 2.4 muestra las clases que se utilizan para el modo maestro y en la Figura 2.5 las del modo automático.

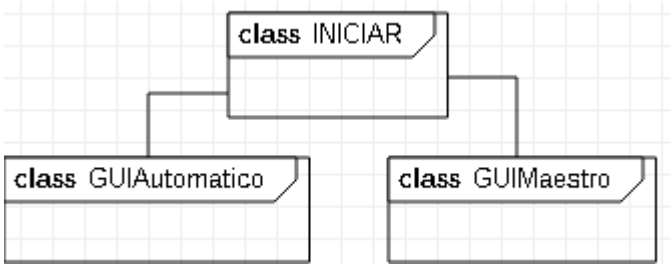


Figura 2.3. Clases de clases de ventanas de interface

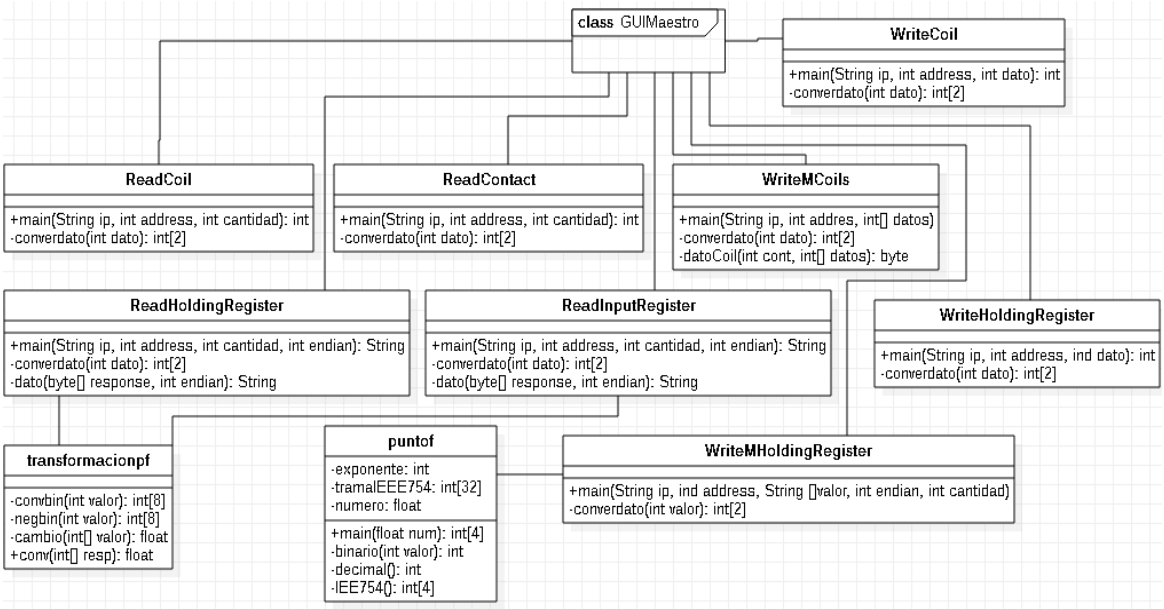


Figura 2.4. Diagrama de clases de modo maestro

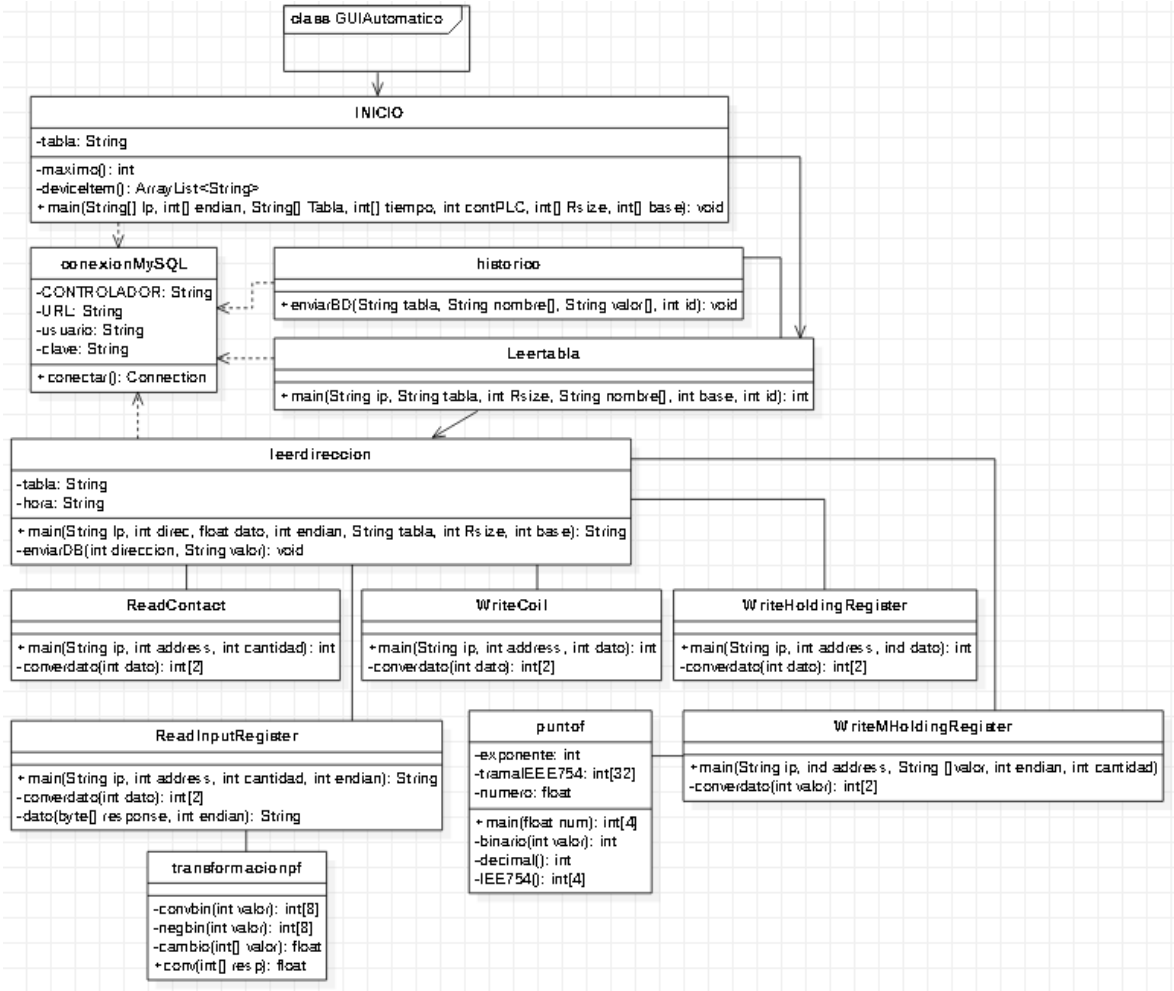


Figura 2.5. Diagrama de clases de modo automático

2.2. Estructura de los códigos de función

Para la realización de los códigos de función, se asignó una clase para cada uno, en donde se implementa el código correspondiente de acuerdo a las entradas que requiere cada código de función para armar la trama del mismo.

Debido a que la trama de Modbus se encuentra estructurado de acuerdo a bytes, se emplean los métodos *write(byte [ ] b)* para él envió de la trama y el método *read(byte [ ] b)* para la lectura de los datos enviados por el dispositivo, estos métodos se utilizan junto con el socket. Quedando para la escritura *getOutStream().write(b)* y para la lectura *getInputStream().read(b)*

2.2.1 Lectura de Coils FC = 01

El código de función lectura de Coils, se utiliza para leer uno o varios Coils por lo cual se requiere como parámetros de entrada los siguientes datos:

- Dirección de inicio de lectura
- Cantidad de Coils a leer

En el Response, ya que los datos leídos son de tipo byte, cada registro contendrá 8 Coils, es decir cada bit del registro representa al valor del Coil leído.

De acuerdo a la trama Modbus los datos de cantidad y dirección de inicio de lectura tienen un tamaño de dos bytes, por lo cual se utiliza el orden de bytes tipo Big Endian mostrado en la Figura 2.6, para estos dos casos de acuerdo a la siguiente manera.

Figura 2.6. Bytes de cantidad y dirección de inicio con formato Big Endian

En el mensaje tipo Request, los bytes correspondientes a la dirección de inicio son los de la posición 8 y 9 mientras que los de cantidad son 10 y 11. Para el dato de longitud al ser la cantidad de bytes desde el identificador de unidad hasta cantidad de registros tiene un valor de 6.

ID Transacción		ID Protocolo		Longitud		ID unidad	Código de función	Dirección de inicio		Cantidad de registros	
0	0	0	0	0	6	1	01	0	2	0	3

Figura 2.7. Ejemplo de Request código de función 1

En la Figura 2.7 se muestra la trama completa del Request mientras en la Figura 2.8 está la trama del Response.

ID Transacción		ID Protocolo		Longitud		ID unidad	Código de función	Contador de bytes	Valor de registro
0	0	0	0	0	4	1	01	1	3

Figura 2.8. Ejemplo de Response código de función 1

Como se observa en las Figuras 2.7 y 2.8 la cabecera es la misma, lo que cambia es el PDU Modbus. En la trama del Response el contador de bytes muestra cuantos registros se han leído de acuerdo a la cantidad enviada en el request. El registro leído en binario para el ejemplo tiene el siguiente valor 00000011, lo cual corresponde a los valores de 1 para los dos primeros Coils leídos y 0 para el tercero. En el diagrama de flujo de la figura 2.11 se detalla el proceso para el código de función lectura de Coils, en la parte de armado de la trama de Request es de acuerdo a la mostrada en la figura 2.7.

2.2.2 Lectura de Entradas Discretas o Contacts FC = 02

El código de función 2 se utiliza para la lectura de entradas discretas, para esta clase se requieren los siguientes datos a ser ingresados por el usuario:

- Dirección de inicio de lectura
- Cantidad de valores a leer

Para el armado de la trama del Request se emplea el ordenamiento de bytes tipo Big Endian para los datos de dirección de inicio y cantidad de registros, debido a que tienen un tamaño de 2 bytes.

ID Transacción		ID Protocolo		Longitud		ID unidad	Código de función	Dirección de inicio		Cantidad de registros	
0	0	0	0	0	6	1	02	0	3	0	4

Figura 2.9. Ejemplo de Request código de función 2

En la Figura 2.9 se muestra un ejemplo con la trama completa para este código, para la cabecera del protocolo los valores de Identificador de transacción e identificador de protocolo tienen un valor de 0, mientras que la longitud tendrá un valor de 6 y el identificador de unidad el valor de 1, el identificador de unidad se utiliza si se emplea un puente serial caso contrario no importa el valor en este campo, para mayor explicación de este campo ver la sección 1.3.3.5. En el PDU se solicita la lectura de cuatro Contacts o

entradas discretas a partir de la dirección 3, es decir se solicita los valores de las direcciones de la 3 a la 6.

ID Transacción		ID Protocolo		Longitud		ID unidad	Código de función	Byte Count	Valor de registro
0	0	0	0	0	4	1	02	1	2

Figura 2.10. Ejemplo de Response código de función 2

En la Figura 2.10 se muestra la trama del Response, en donde el último byte contiene el valor de los Contacts o entradas discretas leídas. El registro leído se lo debe pasar a binario y cada bit representa el valor de un Contact leído. En el ejemplo formulado se obtuvo como respuesta 000000010 por lo tanto el valor de la dirección 3 es cero, el de la dirección 4 es 1 y el de las dos restantes es 0. En la Figura 2.11 se muestra el diagrama de flujo de este código de función, las tramas de Request y Response es de acuerdo a lo mostrado en las figuras 2.9 y 2.10.

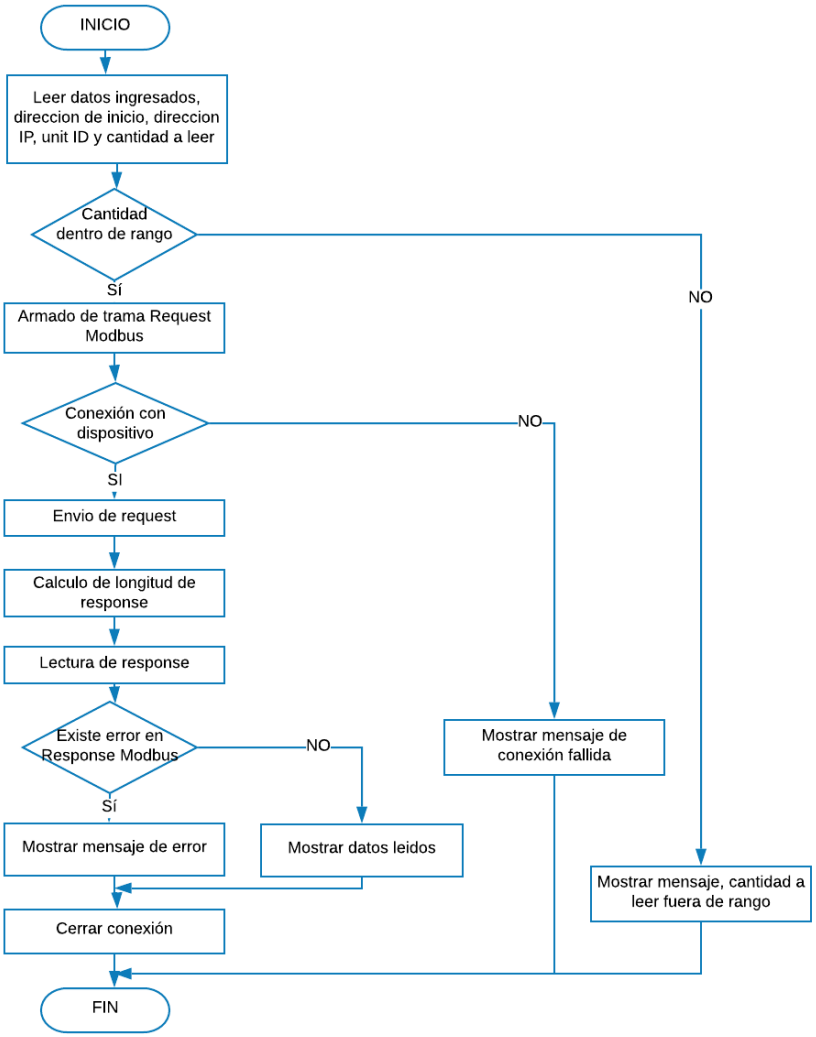


Figura 2.11. Diagrama de flujo lectura de Coils y lectura de Contacts

El diagrama de flujo es el mismo para los códigos de función 01 y 02 pero cambia el armado de la trama por el respectivo de cada código.

2.2.3 Lectura de Holding Register FC = 03

El código de función 3, lectura de Holding Register permite la lectura de uno o varios Holding Register, para esta función se requiere el usuario ingrese los siguientes datos:

- Dirección de inicio de lectura
- Cantidad de registros a leer
- Tipo de variables (Entero o Flotante)

En el armado de la trama igual que en los dos casos anteriores la dirección de inicio y la cantidad de registros a leer se los representa de acuerdo al formato Big Endian mostrado en la Figura 2.6. La cabecera del protocolo TCP permanece con los mismos valores que en los códigos de función anteriores.

ID Transacción		ID Protocolo		Longitud		ID unidad	Código de función	Dirección de inicio		Cantidad	
0	0	0	0	0	6	1	03	0	3	0	2

Figura 2.12. Ejemplo de Request código de función 3

En la Figura 2.12 se muestra la trama para la solicitud de la lectura de dos Holding Register a partir de la dirección 3, lo cual significa la lectura de los registros 3 y 4.

ID Transacción		ID Protocolo		Longitud		ID unidad	Código de función	Contador de bytes	Valor de registro		Valor de registro	
0	0	0	0	0	7	1	03	1	0	10	0	23

Figura 2.13. Ejemplo de Response código de función 3

En la Figura 2.13 se muestra la trama del Response, los valores de los registros solicitados tienen un valor de 10 y 23 respectivamente. En el ejemplo planteado el número es de tipo Entero con el formato Big Endian.

Para este código de función al momento de realizar la lectura existen varios tipos de variables, Word (2 bytes), Float (4 bytes) y Long (4 bytes), la lectura de un valor tipo Float y Long es igual, para las variables tipo Word que representa a números enteros se mantiene el formato Big Endian, sin embargo para las variables de tipo Float el orden de bytes puede ser Big Endian o Little Endian dependiendo del dispositivo, adicionalmente al

ser un número del tipo flotante el dato estará representado por el formato IEEE 754 de simple precisión.

Debido a que el número en tipo Float puede tener cuatro representaciones para ordenar los bytes, se requiere el usuario escoja que orden de bytes se utilizará para la lectura. Las cuatro representaciones son (AB CD), (CD AB), (BA DC) y (DC BA). Al usuario se le permitirá escoger entre dos formas de representar los bytes Big Endian y Little Endian.

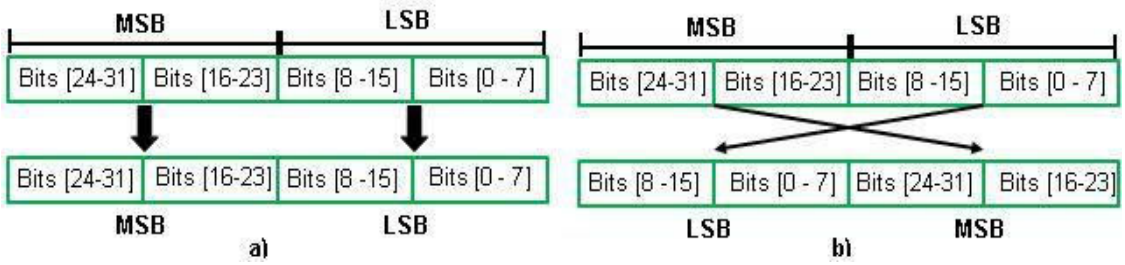


Figura 2.14. a) Datos con orden Big Endian (AB CD), b) Datos con orden Little Endian (CD AB)

En la Figura 2.14 a) se observa el orden de byte para Big Endian como se observa el orden del mismo no cambia, sin embargo en el caso de Little Endian se realiza un intercambio en pares como se observa en la Figura 2.15 b).

Para el bloque de Contador de bytes en el caso de un número de tipo entero el valor es el doble de la cantidad que se está leyendo, debido a que cada número tiene un tamaño de 2 bytes, mientras que para un número Float o Long es cuatro veces la cantidad de datos que se lee ya que cada dato tiene un tamaño de 4 bytes.

En la Figura 2.17 se muestra el diagrama de flujo para el código de función 3 correspondiente a Lectura de Holding Register, para la parte de armado de la trama del Request es de acuerdo a la figura 2.12 y el Response de acuerdo a la figura 2.13.

2.2.4 Lectura de Input Register FC = 04

El código de función 4 o lectura de Input Register, tiene una estructura parecida a la del código de función anterior, tanto en la cabecera del protocolo como en el PDU, pero en el campo código de función cambia el número. Para esta función se requiere el usuario ingrese los siguientes datos:

- Dirección de inicio de lectura
- Cantidad de registros a leer
- Tipo de variables a leer (Entero o Flotante)

El orden de los bytes para las secciones con más de 1 byte es de acuerdo al orden Big Endian, mientras que en el caso del Response las secciones correspondientes a los valores leídos dependerá de que tipo de datos se leen, enteros o flotantes.

ID Transacción		ID Protocolo		Longitud		ID unidad	Código de función	Dirección de inicio		Cantidad de registros	
0	0	0	0	0	6	1	04	0	3	0	2

Figura 2.15. Ejemplo de Request código de función 4

En la Figura 2.15 se observa la solicitud de lectura de dos Input Register a partir de la dirección 3, por lo cual se leerán las direcciones 3 y 4 del bloque Input Register.

ID Transacción		ID Protocolo		Longitud		ID unidad	Código de función	Contador de bytes	Valor de registro		Valor de registro	
0	0	0	0	0	7	1	04	1	0	15	0	87

Figura 2.16. Ejemplo de Response código de función 4

En la Figura 2.16 se muestra el Response de la solicitud realizada en donde se obtiene los valores 15 y 87 pertenecientes a las direcciones 3 y 4.

Como se mencionó anteriormente, de acuerdo al tipo de dato que se va a leer cambia el orden de bytes y el tamaño de los datos, para números enteros como en el ejemplo de la figura 2.18 el tamaño de los datos es de 2 bytes, manteniendo el orden de bytes tipo Big Endian. Para los datos que son tipo Flotante el tamaño de datos es de 4 bytes por lo cual su orden variará de acuerdo a si el dispositivo maneja Big Endian o Little Endian.

Este código de función es utilizado para leer las entradas analógicas del dispositivo, para números enteros se obtiene el dato en binario y se transforma a entero, sin embargo en el caso de número con decimales son representados por el formato IEEE 754 de simple precisión.

En las Figura 2.14 se muestran el orden de bytes para la lectura de los datos de tipo flotante de acuerdo a si son tipo Big Endian o Little Endian.

En la Figura 2.17 se muestra el diagrama de flujo para el código de función 04 y el código de función 03, ya que lo que cambia en la trama Modbus el uno respecto al otro es el campo correspondiente al código de función por el correspondiente a cada código de función.

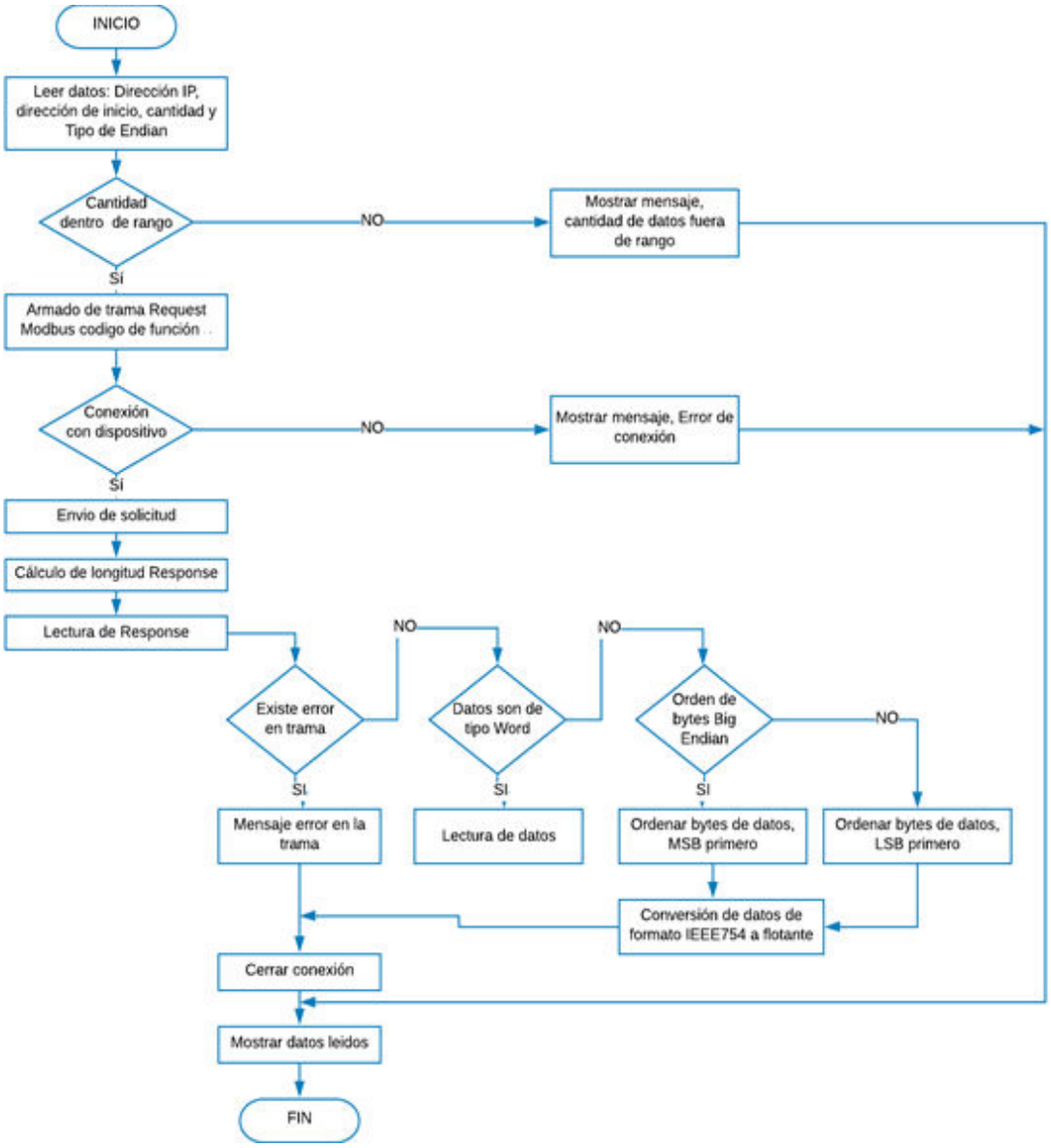


Figura 2.17. Diagrama de flujo lectura de Input Register y lectura de Holding Register

2.2.5 Escritura de un Coil FC = 05

El código de función 5 permite la escritura de un Coil, modificando de esta manera de forma remota el valor de una salida discreta, para esta función se requiere se ingrese los siguientes datos:

- Dirección de escritura
- Valor a escribir (0 o 1)

Para este código de función, la trama de datos Modbus del Request y Response es la misma. En la cabecera del protocolo TCP los valores de Identificador de transacción e Identificador de protocolo son de 0 para el protocolo Modbus, la longitud tiene un valor de

6 y el identificador de unidad un valor de 1, en caso de usar un puente serial se colocará la dirección del esclavo. Para los campos con un tamaño de 2 bytes se mantiene el orden de bytes tipo Big Endian, lo cual significa que el byte más significativo se envía primero en la trama.

ID Transacción		ID Protocolo		Longitud		ID unidad	Código de función	Dirección de inicio		Cantidad de registros	
0	0	0	0	0	6	1	05	0	2	0	0

Figura 2.18. Ejemplo de escritura de un Coil

En la Figura 2.18 se muestra la trama para el Request y Response, en este caso se solicita la escritura en el Coil en dirección 2 con el valor de 0. Obteniendo como respuesta la misma trama.

En la Figura 2.19 se muestra en diagrama de flujo para el código de función 05, en donde se observa los diferentes mensajes que se muestran en caso de error y como está realizado la escritura de un Coil a través del protocolo Modbus TCP.

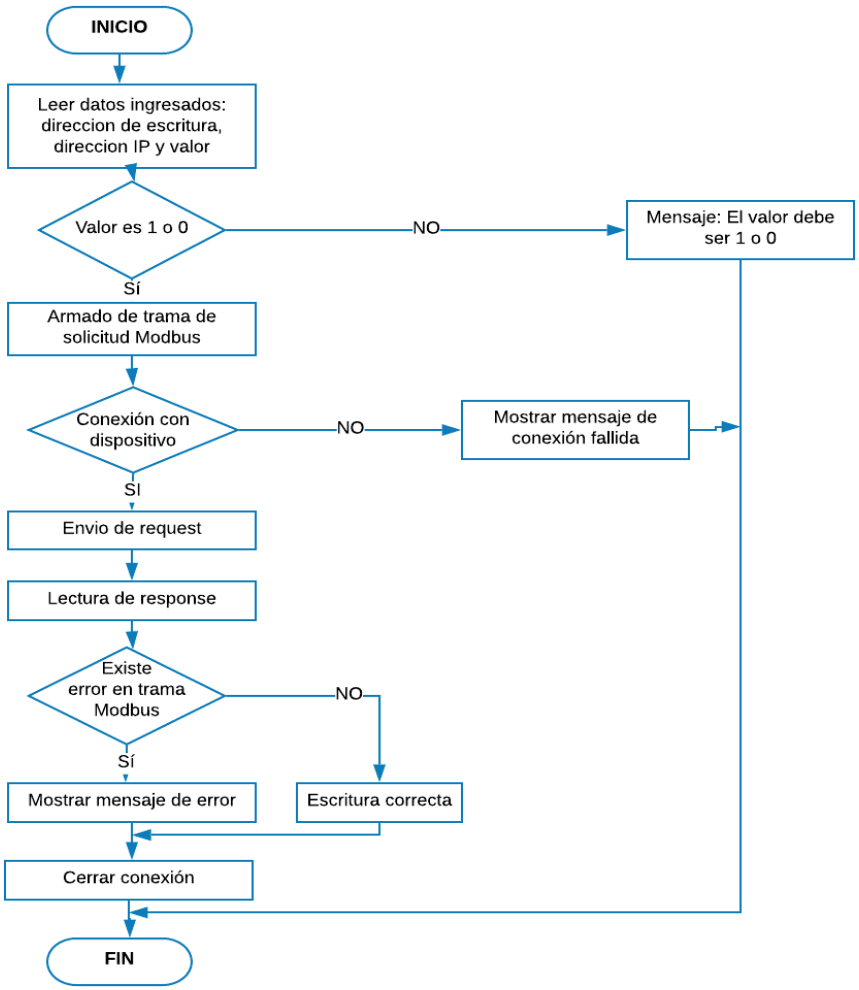


Figura 2.19. Diagrama de flujo escritura de un Coil

2.2.6 Escritura de un Holding Register FC = 06

El código de función 6 permite la escritura de un Holding Register de tipo entero, por lo cual no se puede realizar la escritura de números flotantes, para la escritura de números flotantes se emplea el código de función 16. Para este código de función se requieren los siguientes parámetros de entrada.

- Dirección en la que se va a escribir
- Valor a escribir

En la Figura 2.20 se muestra el diagrama de flujo correspondiente al código de función 6, en el mismo se observa como está constituido este código de función.

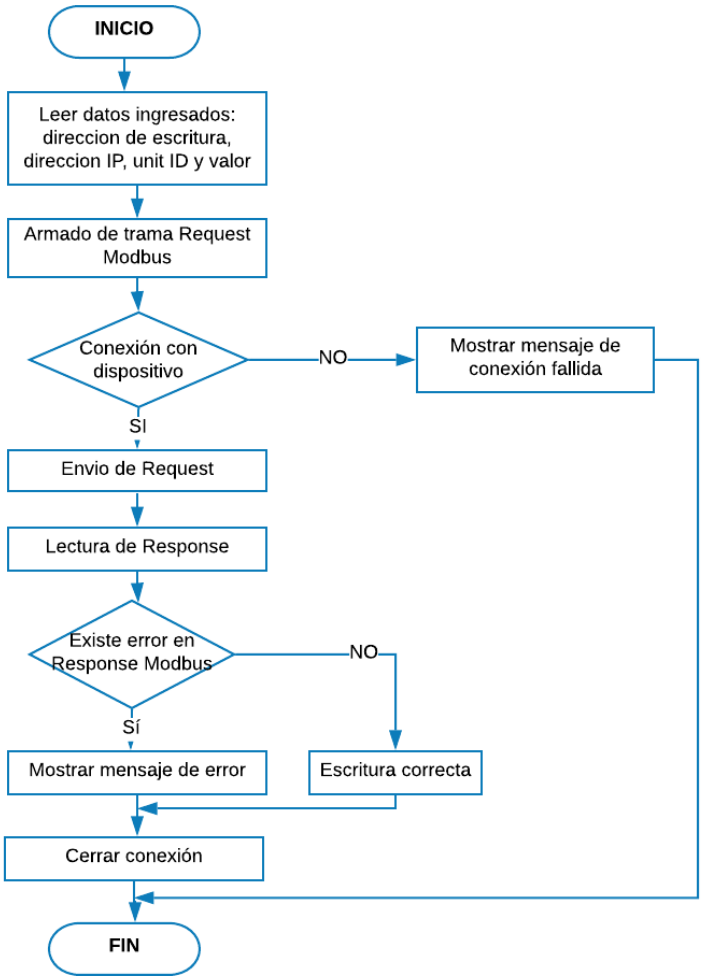


Figura 2.20. Diagrama de flujo escritura de un Holding Register

El Request y el Response tienen la misma trama y por ende la misma longitud, siendo en total 12 bytes, las secciones que tienen un tamaño de 2 bytes emplean el orden de datos tipo Big Endian.

Para la cabecera TCP se tendrá los mismos valores que en el código de función anterior. En la figura 2.21 se puede observar un ejemplo de la trama del Request y Response, en este caso se solicita la escritura del valor 30 en la dirección 2 obteniendo como respuesta la misma trama.

ID Transacción		ID Protocolo		Longitud		ID unidad	Código de función	Dirección de inicio		Cantidad de registros	
0	0	0	0	0	6	1	06	0	2	0	30

Figura 2.21. Ejemplo de escritura de un Holding Register

2.2.7 Escritura de varios Coils FC = 15

El código de función 15 permite realizar la escritura de varios Coils, para lo cual se requiere se ingrese los siguientes datos:

- Dirección de inicio de escritura
- Cantidad de Coils a escribir
- Valores a escribirse

Para realizar el armado de la trama del Request Modbus, primero se debe calcular la cantidad de registros a escribir de acuerdo al número de bytes que se necesitaran para escribir la cantidad de Coils ingresados. Es importante mencionar que cada Coil tiene un tamaño de un bit, por lo cual 8 Coils completarán un byte (Registro). Una vez calculada la cantidad de registros a escribirse y construir los datos registros se realiza el armado de la trama.

La cabecera del protocolo tiene los mismos valores que en los códigos de función anteriores a excepción de la longitud, la cual es el número de byte desde el identificador de unidad hasta el final de la trama. Para los valores de dirección de inicio y cantidad se mantendrá el orden de bytes de acuerdo a Big Endian.

ID Transacción		ID Protocolo		Longitud		ID unidad	Código de función	Dirección de inicio		Cantidad		Contador bytes	Valor de registro
0	0	0	0	0	8	1	15	0	2	0	3	1	6

Figura 2.22. Ejemplo Request de la función 15

En la Figura 2.22 se encuentra la trama para la solicitud de escritura de 3 Coils a partir de la dirección 2, al ser la escritura de 3 Coils solo se requiere de un registro.

ID Transacción		ID Protocolo		Longitud		ID unidad	Código de función	Dirección de inicio		Cantidad	
0	0	0	0	0	6	1	15	0	2	0	3

Figura 2.23. Ejemplo Response de la función 15

En la Figura 2.23 la trama del Response está conformada por la cabecera MBAP y su respectivo PDU conformado por el código de función, la dirección de inicio y la cantidad de Coils escritos.

A continuación en la Figura 2.24 se muestra el diagrama de flujo para este código de función.

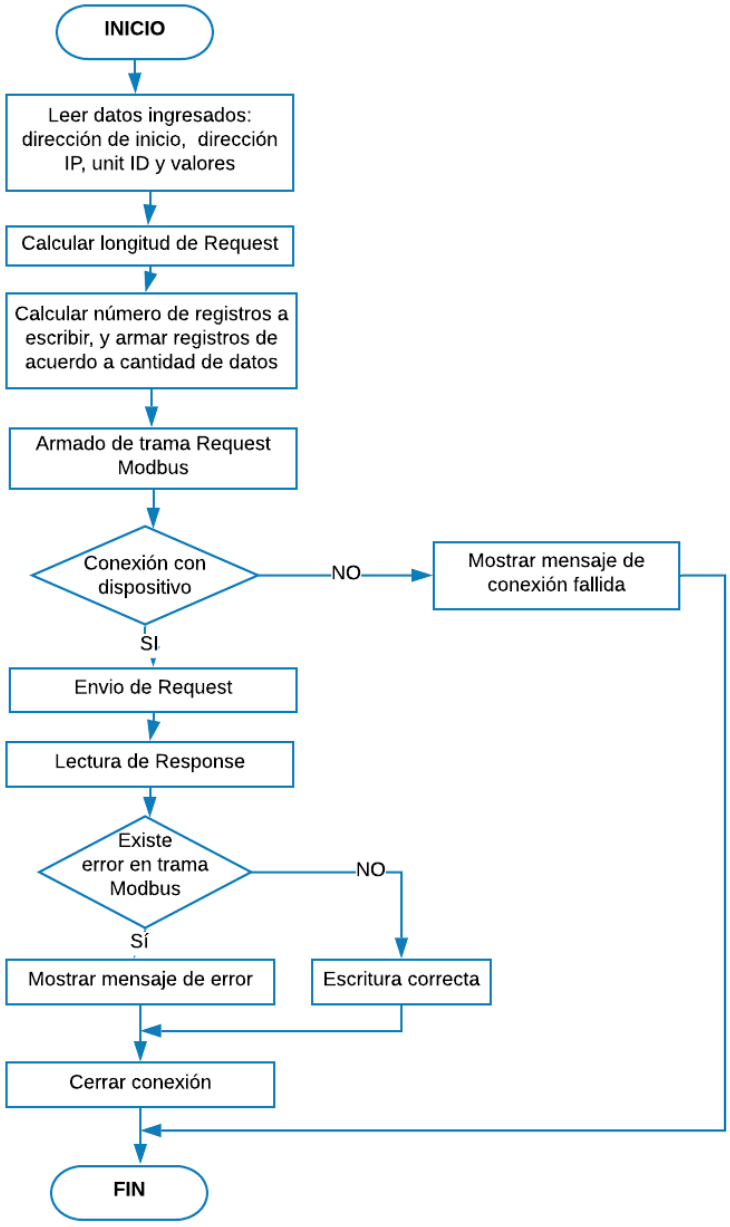


Figura 2.24. Diagrama de flujo código de función 15

2.2.8 Escritura de varios Holding Register FC = 16

El código de función 16 permite la escritura de varios Holding Register de manera continua, los valores a escribirse pueden ser de tipo Entero o Flotante, en el caso de ser Flotante se deberá elegir como se ordenaran los bytes. Los datos a ingresar necesarios son:

- Dirección de inicio
- Cantidad de datos a escribirse
- Valores a escribirse
- Tipo de datos a escribirse (Entero o Flotante)

Para el armado de la trama Modbus, la cabecera TCP está conformado por el identificador de transacción y el de protocolo con un valor de 0, la longitud es la cantidad de bytes desde el identificador de unidad hasta el último bit de la trama y el identificador de unidad tiene un valor de 1.

Para el Request, los datos de dirección y cantidad se mantienen con el orden de bytes de forma Big Endian, pero el orden de los bytes de los datos a escribirse variarán si los número son de tipo flotante, para los datos de tipo Entero se mantiene la forma de Big Endian.

ID Transacción		ID Protocolo		Longitud		ID unidad	Código función	Dirección de inicio		Cantidad		Contador Bytes	Registro 1		Registro 2	
0	0	0	0	0	11	1	16	0	2	0	2	4	0	25	0	14

Figura 2.25. Ejemplo request de la función 16

En el ejemplo realizado en la Figura 2.25 se muestra la trama del Request para solicitar la escritura de dos Holding Register en las direcciones 2 y 3 con los valores de 25 y 14 respectivamente. En la Figura 2.26 se muestra el Response que se obtiene, además toda la trama mantiene el orden de byte Big Endian.

ID		Identificador de protocolo		Longitud		Id unidad	Código de función	Dirección de inicio		Cantidad	
0	0	0	0	0	6	1	16	0	2	0	2

Figura 2.26. Ejemplo response de la función 16

Como se mencionó anteriormente, este código de función permite la escritura de diferentes tipos de datos, para los datos de tipo Entero (Word) tienen un tamaño de 2 bytes, mientras que para los datos de tipo Flotante tienen un tamaño de 4 bytes.

Adicionalmente para los datos de tipo flotante se los debe transformar al formato IEEE754 de simple precisión para su representación y el orden de los bytes puede ser Big Endian o Little Endian dependiendo del dispositivo. En el caso de un número flotante tipo Big Endian en la trama primero se coloca el byte más significativo, mientras que en el caso de Little Endian se envía primero los bytes menos significativos. En las figura 2. 15 se muestra como es el orden los bytes de acuerdo a si son tipo Big Endian o Little Endian.

A continuación en la Figura 2.27 se muestra el diagrama de flujo para este código de función.

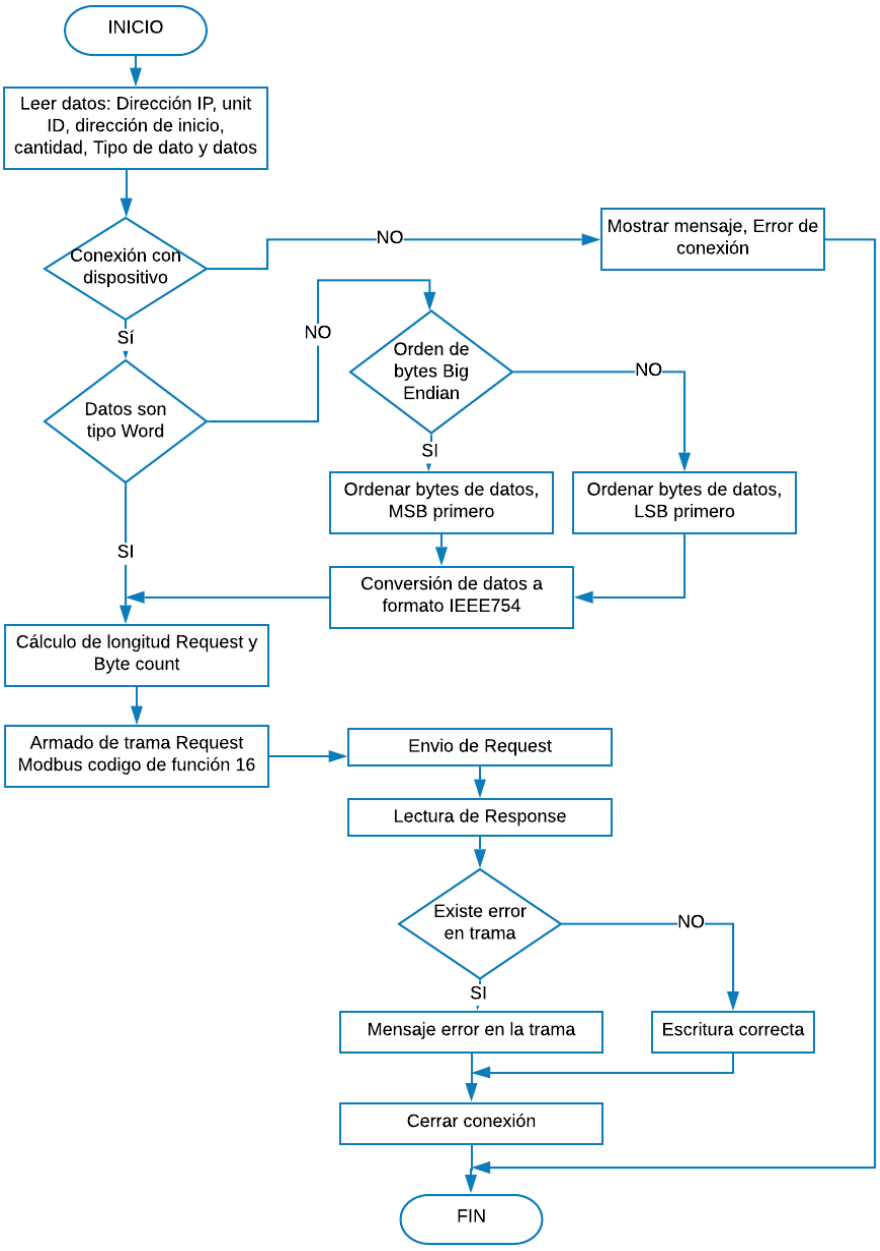


Figura 2.27. Diagrama de flujo de código de función 16

2.3. Conexión TCP/IP desde el software Java

Se inicia definiendo los requerimientos para la comunicación del software libre Java con un dispositivo mediante el protocolo TCP/IP a través de sockets, tras lo cual se establecen las clases y los métodos a utilizar, así como los parámetros que deberán configurarse, estos se detallarán a continuación.

Para abrir una conexión son importantes dos datos, los cuales son: la dirección IP y el puerto de conexión, sin embargo se pueden agregar otros parámetros como el tiempo de espera. Java cuenta con dos clases que permiten la conexión TCP, estas son Socket y ServerSocket. La clase ServerSocket se utiliza cuando el dispositivo funciona como servidor, permitiendo que se abra una conexión IP y se mantenga escuchando hasta que un cliente se conecte. La clase Socket se utiliza para realizar la conexión de un cliente. Estas dos clases cuentan con diferentes métodos que permiten la conexión, envió y recepción de datos. En la Tabla 2.1 se muestran los métodos más importantes de la clase Socket.

Tabla 2.1. Métodos de la clase Socket

Tipo de retorno	Método	Descripción
Void	close ()	Cierra la conexión del socket
Void	connect (SocketAddress endpoint, int timeout)	Conecta el socket con un tiempo específico de espera de respuesta.
InputStream	getInputStream()	Retorna una secuencia de entrada del socket
OutputStream	getOutStream()	Retorna una secuencia de salida del socket
InetAddress	getInetAddress()	Retorna la dirección a la cual se realizó la conexión

En el trabajo realizado se emplea el método *getOutStream()* para enviar la trama de datos Modbus y el método *getInputStream()* para obtener la trama de datos enviados por el dispositivo industrial hacia el software.

A continuación en la Figura 2.28 se muestra el diagrama de flujo para la conexión TCP/IP de Java con un dispositivo, empleando la clase Socket y los métodos mencionados anteriormente. Adicionalmente se debe detectar si la conexión fue exitosa o no, mediante el uso de las sentencias TRY y CATCH para el manejo de Excepciones.

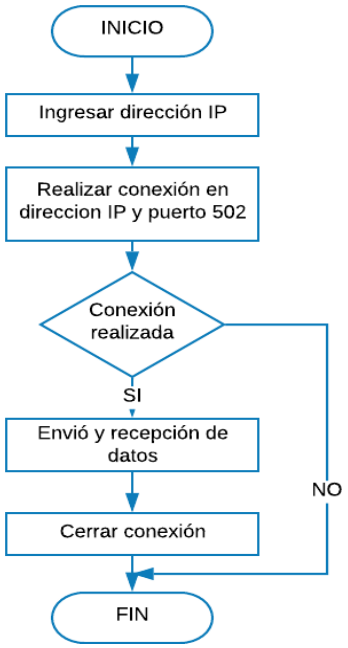


Figura 2.28. Diagrama de flujo conexión TCP/IP

2.4. Creación y Gestión de la Base de Datos

Como gestor de base de datos se utiliza MySQL, mismo que mediante lenguaje SQL puede mantener una comunicación con aplicaciones desarrolladas en Java, sin embargo es necesario agregar un conector/J para MySQL en el proyecto de Java, para poder realizar la comunicación entre las dos aplicaciones. En el presente trabajo se ha usado el conector/J 5.1.47, el cual tiene las siguientes características.[27]

- Adecuado para el uso con versiones 5.5, 5.6, 5.7 y 8.0 de MySQL Server
- Compatible con la API 4.2 de Java Database Connectivity (JDBC)

2.4.1 Conexión de Java a la base de datos

Para poder realizar la conexión a la base de datos es importante conocer los siguientes datos:

- Dirección del controlador del conector: para el conector JDBC que se usa en este proyecto, la dirección es: *com.mysql.jdbc.Driver*
- URL: Dirección en la cual se encuentra la base de datos, en esta dirección se encuentran datos como: protocolo de conexión, hosts, puerto de conexión, nombre de la base de datos, y llaves en caso de existir. Para este proyecto el URL es: *jdbc:mysql://localhost:3307/modbus*
- Usuario: usuario que tenga acceso a la base de datos, el usuario general es *root.*

- Contraseña: La contraseña del usuario.

A continuación en la Tabla 2.2 se muestra las clases que se utilizan para la conexión de Java con la base de datos, adicionalmente se emplea TRY y CATCH para el manejo de excepciones relacionadas con la conexión y reportando si se produjo algún error en la conexión.

Tabla 2.2. Clases y métodos utilizados para conexión de Java a MySQL

Estructura	Descripción
Class.forName(String ClassName)	Retorna el nombre de la clase asociado a la cadena de caracteres ingresados. Permite cargar el controlador
DriverManager.getConnection(String url, String user, String password)	La clase DriverManager permite administrar los controladores JDBC y el método getConnection establece una conexión con los parámetros ingresados

2.4.2 Lectura de datos de base de datos

Para la lectura de datos se debe primero realizar la conexión a la base de datos, como se mostró en el punto anterior.

Para poder realizar la lectura de datos se utiliza los métodos *createStatement()* para la sentencia SQL y el método *executeQuery(string SQL)* para ejecutar la consulta por ejemplo:

```
ResultSet rs= null;

Statement stn = cn.createStatement();

rs = stn.executeQuery("SELECT * FROM nombre_de _tabla");

while(rs.next){ // lectura de filas de la tabla

int valor=rs.getInt(1); }
```

Mediante estos métodos se realiza la lectura de la tabla. El metodo next () se emplea para leer cada fila, y se lo coloca en un while para que realice la lectura de todas las filas de la tabla, dentro del while se coloca de que columnas se quiere el valor. En el ejemplo mostrado se lee los valores de la primera columna de la tabla. Para el presente trabajo los valores leidos se guardan en un vector, para su posterior utilización.

2.4.3 Escritura en base de datos

Se utilizan el método *prepareStatement (String sql)* para escribir la sentencia sql que se va a enviar y el método *executeUpdate()* para generar la consulta.

En el caso de la escritura de datos hay diferentes sentencias sql que se pueden utilizan:

- **UPDATE**: Actualiza el valor de una o más columna en una fila ya existente, la estructura es la siguiente para Java:

  UPDATE tabla SET columna=valor WHERE nombrecolumna=condición

- **INSERT**: Inserta una nueva fila a continuación en la tabla, la estructura de la sentencia es la siguiente:

  INSERT INTO tabla (columna1, columna2,...columnaN) values (valor1, valor2,...valorN)

2.4.4 Sentencias SQL para interactuar con una base de datos

Los métodos mencionamos anteriormente *prepareStatement (String sql) y executeUpdate()* permiten realizar cualquier consulta SQL que no espere un retorno, por lo cual se pueden emplear diferentes sentencias SQL. A continuación se detallaran los utilizados en este trabajo

- **Drop Table:** Permite eliminar todos los datos de una tabla. Su estructura es: DROP TABLE nombre_tabla.

- **Create Table:** Crea una nueva tabla con las columnas y características especificadas. Su estructura SQL es:

  CREATE TABLE nombre_tabla (columna1, tipo_columna1, atributos,…., campoN, tipo_columnaN, atributo)

- **Alter Table:** Permite agregar una nueva columna en una tabla que ya existe con características específicas. Su estructura SQL es:

  ALTER TABLE nombre_tabla ADD columna, tipo_columna, atributos AFTER/ BEFORE columna_existente

- **Selec max:** Obtiene el máximo número de una columna en la tabla especificada. Su estructura SQL es:

  SELECT MAX(nombre_columna) FROM nombre_tabla

2.5. Diseño de la interfaz gráfica

El diseño de la interfaz gráfica de usuario desarrollado en Java permite al usuario realizar la configuración del servidor de datos implementado y utilizarlo de acuerdo a sus modos de funcionamiento. La interfaz cuenta con tres ventanas: Menú Principal, Modo de funcionamiento 1 y Modo de funcionamiento 2. El primer modo permite al usuario realizar la conexión a un dispositivo y probar los 8 códigos de función básicos en forma individual, introduciendo los parámetros necesarios de acuerdo a cada código de función. El segundo modo, cuya configuración se ha estructurado en forma similar a servidores de datos comerciales, permite una conexión entre varios dispositivos Modbus TCP y el servidor de datos, manteniendo una comunicación automática.

2.5.1. Ventana de Inicio

La ventana de inicio mostrada en la Figura 2.29 contiene una presentación del servidor y permite seleccionar con cuál de los dos modos se va a trabajar, el primer modo denominado MAESTRO para probar los 8 códigos de función individualmente y el segundo modo, denominado AUTOMÁTICO con comunicación automática.

Figura 2.29. Ventana de Inicio

2.5.2 Ventana Modo de Maestro

Para este modo se emplea una comunicación entre el dispositivo industrial y el servidor de datos desarrollado en Java, como se muestra en la Figura 2.30. La comunicación en

este modo se la realiza utilizando el protocolo Modbus TCP por lo cual se emplean los códigos de función detallados anteriormente.

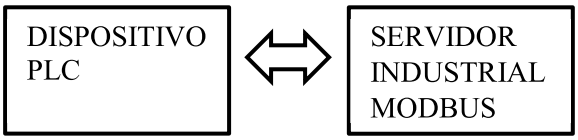


Figura 2.30. Comunicación de primer modo de funcionamiento

Este modo permite comprobar el funcionamiento de los 8 códigos de función básicos. La ventana para este modo está dividida en secciones para una mayor facilidad, como se muestra en la Figura 2.31.

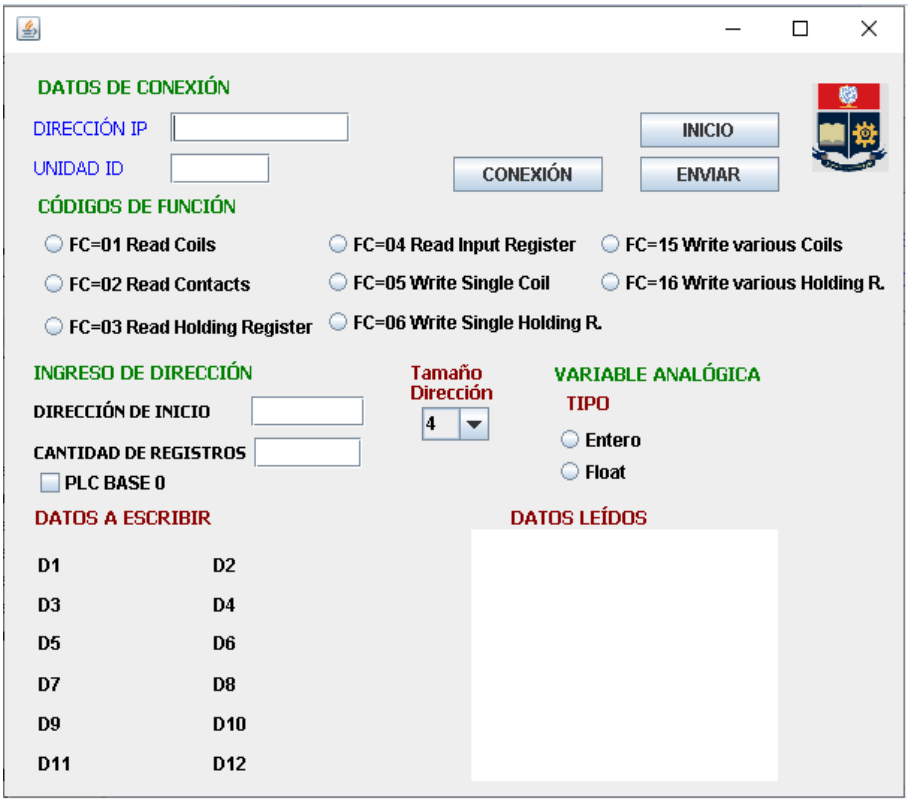


Figura 2.31. Ventana modo 1

A continuación, se detalla las secciones que componen esta ventana

- **Datos de conexión:** En esta sección se realizará el ingreso de la dirección IP del dispositivo a conectar y el identificador de unidad (UNIDAD ID). La dirección IP a introducción debe mantener el formato de 4 octetos separados por un punto XXX.XXX.XXX.XXX por ejemplo 192.168.10.43. El identificador de unidad puede tener valores de 1 a 255.

- **Códigos de función:** Se puede elegir el código de función con el cual se trabajara. Los códigos de función 01, 02, 03 y 04 son de lectura mientras los códigos de función 05, 06, 15 y 16 son de escritura.
- **Ingreso de Dirección Modbus:** En esta sección se ingresara la dirección de inicio del registro Modbus, además de especificar el tamaño de la dirección, mismo que puede ser de un tamaño de 4, 5 o 6 dependiendo el dispositivo a conectarse. Adicionalmente se escoge si el PLC es de base 0 o no. En la cantidad de registro se ingresa cuantos valores se van a leer o escribir, este parámetro es necesario para todos los códigos de función a excepción de los códigos de función 5 y 6 debido a que realizan la escritura de un valor.
- **Datos a escribir:** Una vez ingresado la cantidad de registros a escribir se habilita el ingreso de los datos, para los códigos de función 15 y 16 de acuerdo a la cantidad especificada anteriormente, mientras que para los códigos de función 05 y 06 se habilita solo un cuadro para el ingreso del dato.
- **Datos Leídos:** Muestra los datos leídos con los códigos de función 01, 02, 03 y 04.
- **Variable analógica:** Para los códigos de función 03, 04 y 16 que trabajan con una variable tipo Holding Register o Input Register al poder ser de dos tipos se requiere se especifique si se trata de una variable de tipo entero o flotante. Adicionalmente en caso de ser una variable tipo flotante se debe especificar el tipo de Endian del dispositivo.

Finalmente se tiene tres botones, mismos que son detallados a continuación:

- **INICIO:** Regresa a la ventana anterior para seleccionar el tipo de modo
- **CONEXIÓN:** Permite comprobar la conexión con el dispositivo cuya dirección IP fue ingresada en la sección Dirección IP.
- **ENVIAR:** Ejecuta el código de función seleccionado.

2.5.3 Ventana Modo de funcionamiento 2

El segundo modo realiza una comunicación automática manteniendo una comunicación como se muestra en la Figura 2.32. Como se observa en la figura el servidor industrial Modbus funciona como un intermediario entre el dispositivo industrial y una base de datos en MySQL. La comunicación entre el servidor y el dispositivo industrial se la realiza

empleando los códigos de función del protocolo Modbus detallados anteriormente, y la comunicación con la base de datos se la realiza empleando el lenguaje SQL también mencionado anteriormente. La base de datos facilita la interacción con otras aplicaciones de Windows en donde se ejecute un HMI, como por ejemplo un HMI desarrollado en InTouch el cual utilice los datos almacenados en la Base de Datos.

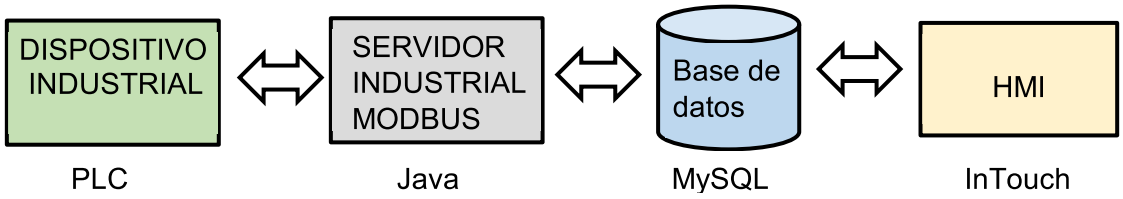


Figura 2.32. Comunicación de segundo modo de funcionamiento

Este modo a diferencia del anterior no trabaja con todos los códigos de función, si no solo con los necesarios para realizar la comunicación automática entre el dispositivo y el servidor de datos industrial. En la Tabla 2.3 se observan los códigos de función utilizados.

Tabla 2.3. Códigos de función modo de funcionamiento 2

Código de función		Tamaño de dato	Descripción
Código	Código (hex)		
02	02	1 bit	Lectura de Contact
04	04	16 bits o 32bits	Lectura de Input Register
05	05	1 bit	Escritura de un Coil
06	06	16 bits	Escritura de un Holding Register
16	10	32 bits	Escritura de varios Registros

Los códigos de función 2 y 4 permiten conocer el estado de las entradas Discretas y Analógicas mientras que el código de función 5 permite modificar el estado de una salida discreta, para el caso de la salida analógica (Holding Register) al ser de dos tipos se emplea el código de función 06 para modificar las variables Enteras y el código de función 16 para las variables flotantes.

Para este modo, se tienen tres pestañas en las cuales se puede realizar la configuración del PLC, ingreso de datos y visualización de datos.

2.5.3.1 Configuración del servidor

La pestaña CONFIGURACION, mostrada en la Figura 2.33, permite realizar la configuración respectiva de los dispositivos a conectarse. A continuación son detallados los parámetros a configurar.

- Dirección IP.

- Identificador de unidad.
- Tiempo de muestreo, puede ser diferente para cada PLC.
- Nombre de tabla, en la cual se almacenaran los datos en la base de datos.
- Tipo de variable analógica, en caso de ser flotante se debe especificar el Endian.
- Tamaño de dirección del registro Modbus, pueden ser de tres tamaños 4, 5 o 6.
- Si el PLC es de base cero o no.

Figura 2.33. Ventana configuración de datos

La principal función de esta pantalla es: permitir el ingreso de los datos del dispositivo a conectar, crear una tabla para guardar los datos en caso de no existir y realizar la conexión de manera automática. Cuenta con los siguientes botones:

- **AGREGAR PLC:** Permite guardar la configuración de un dispositivo y continuar con la siguiente configuración. La comunicación se realizará con los dispositivos que hayan sido agregados al servidor.
- **CREAR TABLA:** Permite crear dos nuevas tablas en la base de datos, una tipo histórico y otra que almacena los datos en tiempo real del PLC configurado.
- **GUARDAR CONFIGURACIÓN:** Permite guardar la última configuración de los PLCs que han sido agregados al servidor en una base de datos creada en MySQL.
- **CARGAR CONFIGURACIÓN:** Carga en el servidor la última configuración guardada.

- **CONECTAR:** Comienza la comunicación continua con el/los PLCs agregados al servidor.

2.5.3.2 Base de Datos

La pestaña BaseDatos, mostrada en la Figura 2.34, permite realizar la asignación de los registros Modbus del dispositivo para ser almacenados en la base de datos. En esta parte se ingresa el nombre del registro y la dirección Modbus. La dirección ingresada debe tener el tamaño que se seleccionó en la pestaña anterior, caso contrario al realizar la conexión se mostrará que hay un error en la dirección.

En la parte Cantidad de Registros se ingresará cuantos registros se van a configurar y al dar Enter, se habilitarán los recuadros para el ingreso de los datos.

Al presionar el botón ACTUALIZAR se actualizará los datos ingresados en la base de datos en la tabla ingresada en la pestaña anterior.

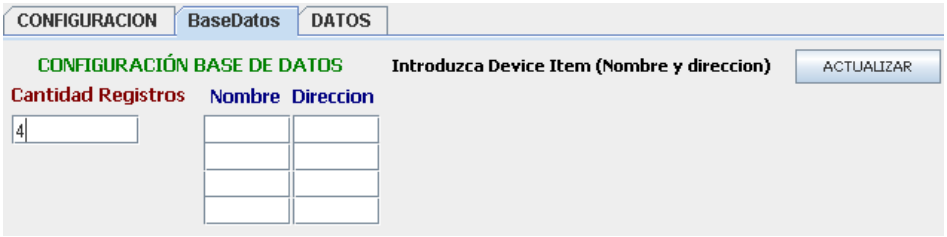


Figura 2.34. Pestaña de Base de Datos

La Figura 2.35 muestra un diagrama de flujo acerca del funcionamiento de esta pestaña.

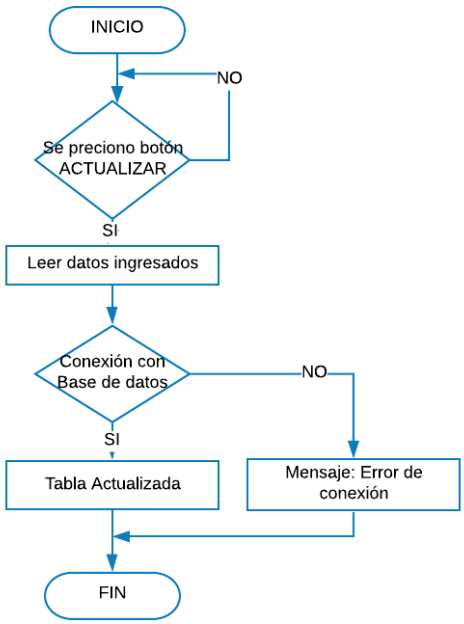


Figura 2.35. Diagrama de flujo ingreso datos de los registros del PLC

2.5.3.3 Visualización de Datos

En la Figura 2.36 se muestra la pestaña DATOS, misma que permite la visualización de los datos en tiempo real de la tabla solicitada, estos valores se actualizarán cada 1 ms.

Figura 2.36. Pestaña DATOS

En la parte de TABLA se debe colocar el nombre de la tabla que se desea observar y al momento de presionar MOSTRAR empezara a mostrarse la tabla con las columnas de: nombre de la variable, valor y el tiempo de su última actualización.

En la Figura 2.37 se observa el diagrama de flujo de la pestaña de DATOS, en la cual se detalla el funcionamiento de la misma.

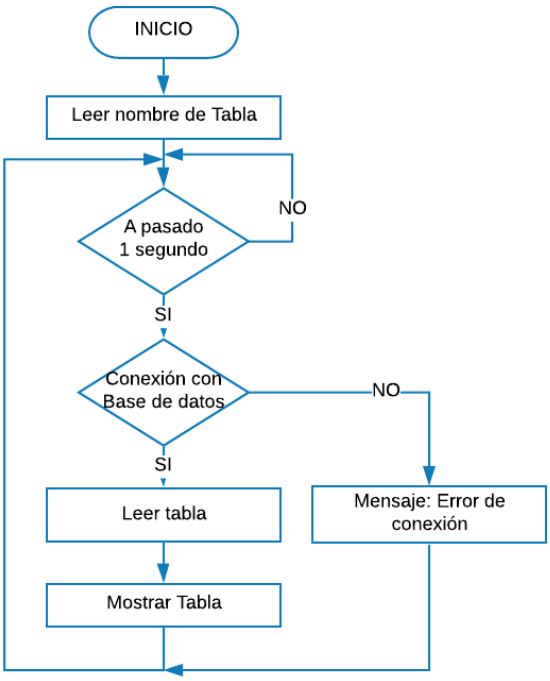


Figura 2.37. Diagrama de flujo

2.6. Conexión de la base de datos en MySQL con InTouch

Para la validación de la interoperabilidad con aplicaciones de software comerciales del servidor desarrollado, se realizó pruebas de integración con aplicaciones de la plataforma

Wonderware, una de estas aplicaciones es InTouch. Por ello en esta sección se detalla el procedimiento para establecer una comunicación entre el servidor desarrollado e InTouch.

Para poder conectar una base de datos de MySQL con InTouch es necesario primero instalar un conector ODBC, una vez instalado se realiza la configuración de este conector, ingresando los datos de la base de datos, como: puerto de conexión, usuario, contraseña, nombre de la base de datos y la dirección IP de la PC en donde se encuentre el gestor de base de MySQL con la base objetivo. En caso de estar en la misma máquina la base de datos e InTouch, se coloca localhost o la dirección IP 127.0.0.1.

InTouch ya cuenta con sentencias SQL para poder realizar la comunicación, estas sentencias tienen una estructura de acuerdo a los parámetros necesarios para cada sentencia. Sin embargo para utilizar las sentencias SQL en InTouch es necesario la creación de un BlindList, mismo que permitirá relacionar las columnas de la base de datos con un tagname. Las sentencias utilizadas de SQL en el presente trabajo son las siguientes, mostradas en la Tabla 2.4.

Tabla 2.4. Sentencias SQL de InTouch

Función	Descripción
SQLConnect(ConnectionID, ”Provider=MSDASQL;DSN=MySQL”);	Abre la conexión con la base de datos de MySQL, atreves del conector ODBC
SQLDisconnect(ConnectionID);	Cierra la conexión con la base de datos
SQLUpdate(ConnectionID,”tabla”, ”Blind List”,”Where expresion”);	Escribe el valor de los tagname relacionados con las columnas en el Blind List en la fila que cumpla expresión escrita.
SQLSelect(ConnectionID,”tabla”, ”Blind List”, ”Where expression”,””);	Lee el valor del registro en la expresión dada sin ningún orden.
SQLLast(ConnectionID);	Esta función selecciona el ultimo valor obtenido de la función colocada previamente SQLSelect()
SQLEnd(ConnectionID);	Libera memoria utilizada para almacenar el contenido de la tabla.

3. RESULTADOS Y DISCUSIÓN

En esta sección se presenta los resultados más relevantes obtenidos en las pruebas realizadas para verificar el funcionamiento del software desarrollado.

A continuación, se detallará tres secciones, las dos primeras muestran las pruebas para cada modo implementado y en la tercera se presenta una comparación entre el servidor realizado con el software comercial DASBMTCP de Wonderware.

En la prueba de funcionamiento del modo Maestro, mediante los PLCs ML1400 y M580 se verifica el funcionamiento de algunos códigos de función, mientras con el Simulador Modbus se probarán todos los códigos de función.

Para el modo Automático, inicialmente se muestra las pruebas realizadas con los PLCs ML1400 y M850, mismo que se encuentran en red con un programa básico que posee los cuatro bloques de memoria del protocolo y su respectivo HMI en InTouch. Además se muestra un prototipo con el proceso de fabricación de Chocolate, en el cual se utilizan 3 PLCs para controlar y monitorear el proceso.

3.1. Pruebas realizadas modo Maestro

Como se mencionó anteriormente, este modo permite realizar la comunicación del servidor con un dispositivo industrial y probar los 8 códigos de función individualmente. Con el simulador PLC Modbus se probaron los 8 códigos de función, mientras que con los dispositivos PLC M580 y ML1400 se realizaron las pruebas con algunos códigos de función.

3.1.1. Pruebas de códigos de función con Simulador de PLC Modbus

A continuación se muestran las pruebas realizadas para los 8 códigos de función con el simulador Slave Modbus.

El simulador tiene las siguientes características:

- Dirección IP: dirección local (127.0.0.1).
- Base: cero.
- Tamaño de dígitos: 5.
- Variable analógica: Tipo Entero o Tipo Float con orden de bits Big Endian o Little Endian.

La prueba realizada con el código de función 1 se muestra en la Figura 3.1. Se realiza la lectura de 5 Coils a partir de la dirección 1. En la parte derecha se muestra el bloque de Coils del Simulador de PLC y en la parte izquierda la interface del modo Maestro. En la sección de Datos Leídos, se encuentran los datos solicitados, los cuales son los mismos del simulador.

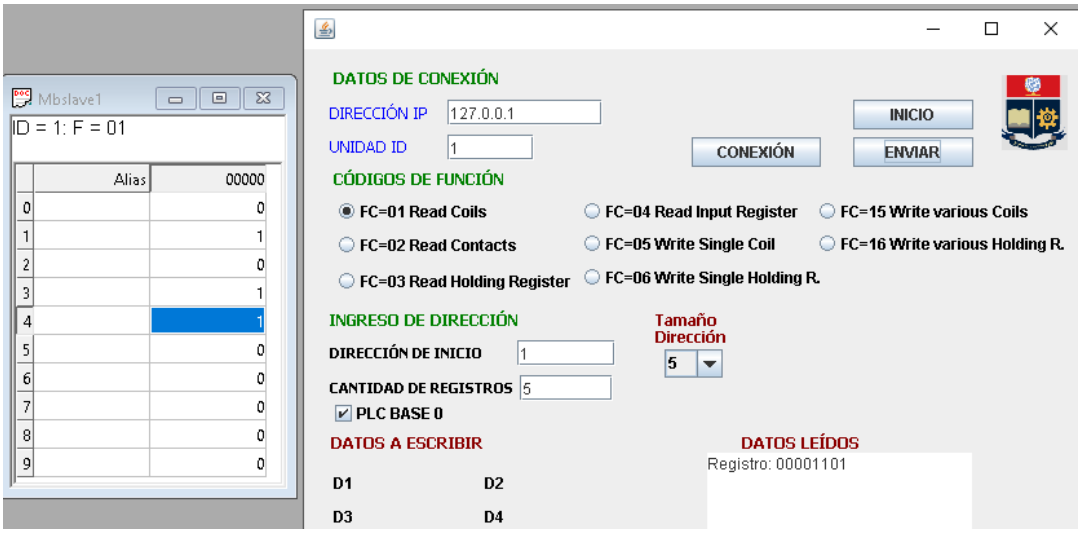


Figura 3.1. Prueba Código de función 1 con Simulador de PLC Modbus

En la Figura 3.2 se muestra la prueba para el código de función 2, Lectura de Contacts. La prueba realizada consiste en la lectura de 10 Contacts, ya que el simulador tiene base 0 se lee desde la dirección 10000. Como se comprueba en la interface, los datos leídos son los mismos que se encuentran en el Simulador de PLC.

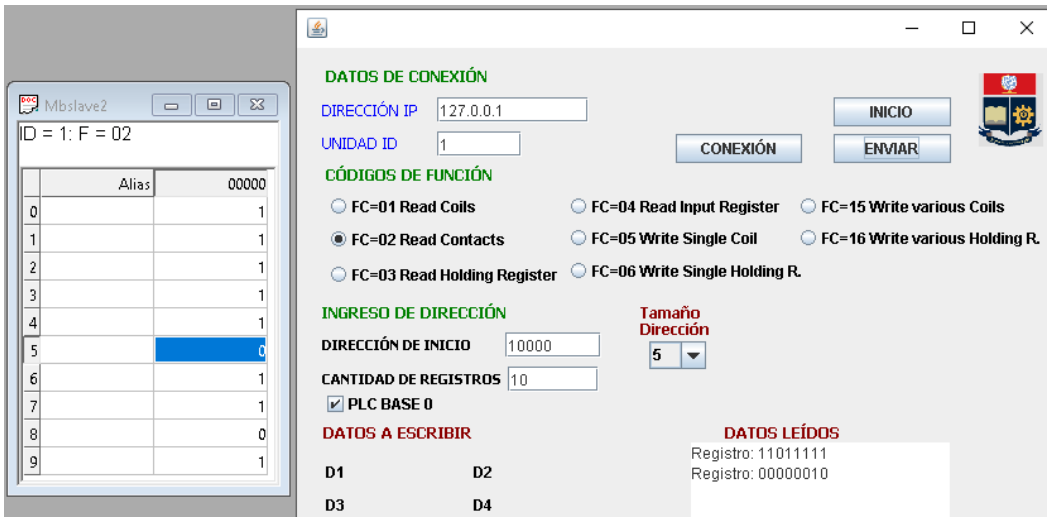


Figura 3.2. Prueba Código de función 2 con Simulador de PLC Modbus

En la Figura 3.3 se encuentra la prueba realizada para el código de función 03. Para esta prueba se leen tres registros Holding Register tipo Float con ordenamiento de bytes tipo Big Endian. Como se observa los registros leídos son los mismos que en el simulador, obteniendo como datos, 35.64, -234.04 y 10002.75

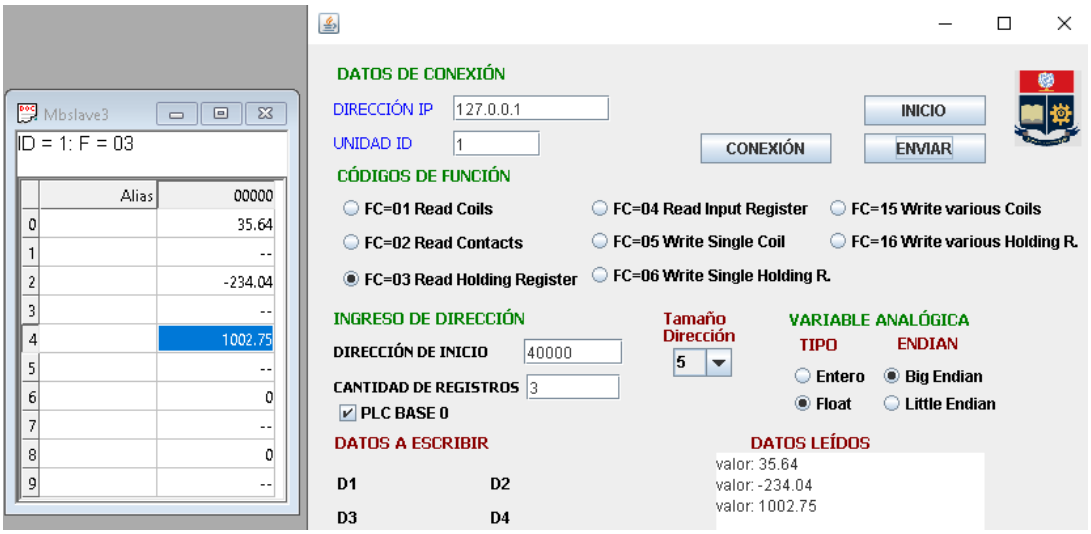


Figura 3.3. Prueba Código de función 3 con Simulador de PLC Modbus

En la Figura 3.4 se muestra la prueba realizada para el código de función 04. Se realizó la lectura de cuatro registros Input Register tipo Float con ordenamiento de bytes Little Endian desde la dirección 30000. Como se observa en la figura los valores obtenidos de la lectura son los mismos del simulador mostrado en la parte izquierda, en la parte derecha en la sección datos leídos se muestran los valores de los registros, los cuales son: 0, 765.8, -98.6 y 0.01.

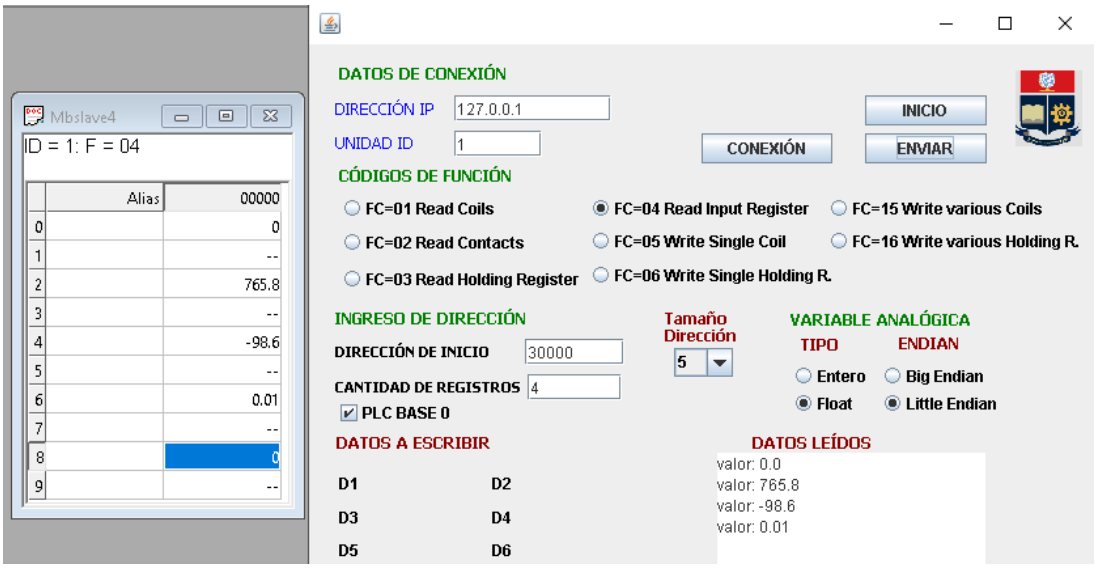


Figura 3.4. Prueba Código de función 4 con Simulador de PLC Modbus

En la Figura 3.5 se muestra la prueba realizada para el código de función 05. Realizando la escritura de un Coil en la dirección 9 con el valor de 1. En la parte izquierda se muestra el simulador y en la derecha la interfaz del servidor desarrollado. Como se observa en la figura, se realiza correctamente la escritura del Coil con el valor planteado.

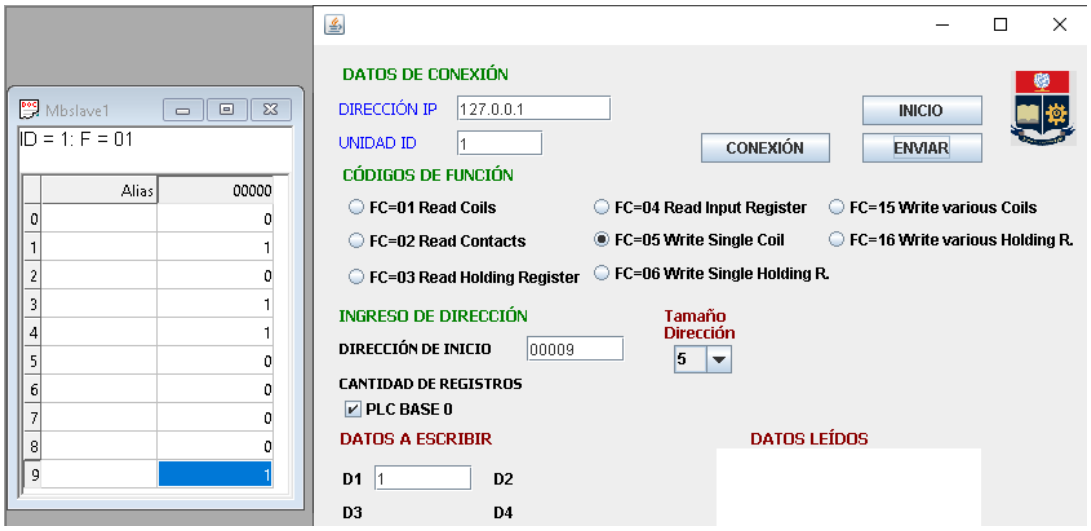


Figura 3.5. Prueba Código de función 5 con Simulador de PLC Modbus

En la Figura 3.6 se muestra la prueba realizada para el código de función 06. La prueba consiste en la escritura de un Holding Register en la dirección 40003 con el valor de 25. Ya que este código de función se emplea con números enteros se cambia en el simulador a Enteros. En la prueba realizada se valida el funcionamiento al escribirse en el simulador en valor introducido en el servidor.

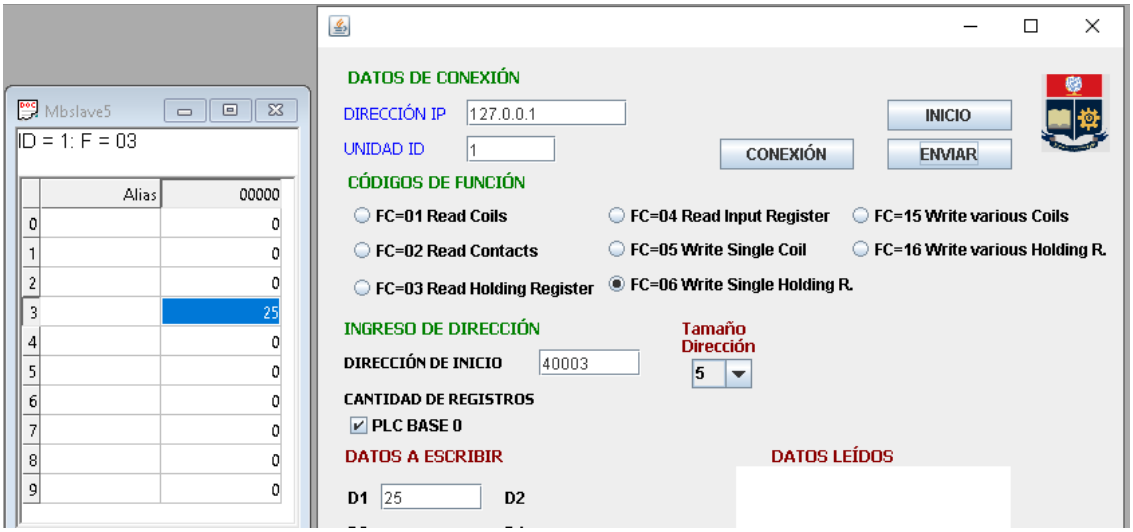


Figura 3.6. Prueba Código de función 6 con Simulador de PLC Modbus

En la Figura 3.7 se realiza la prueba para el código de función 15, donde a partir de la dirección 0 se realiza la escritura de 9 Coils. Los datos son introducidos en la sección de Datos a escribir en la Interfaz en JAVA y como se observa en el simulador de PLC, son los mismos datos.

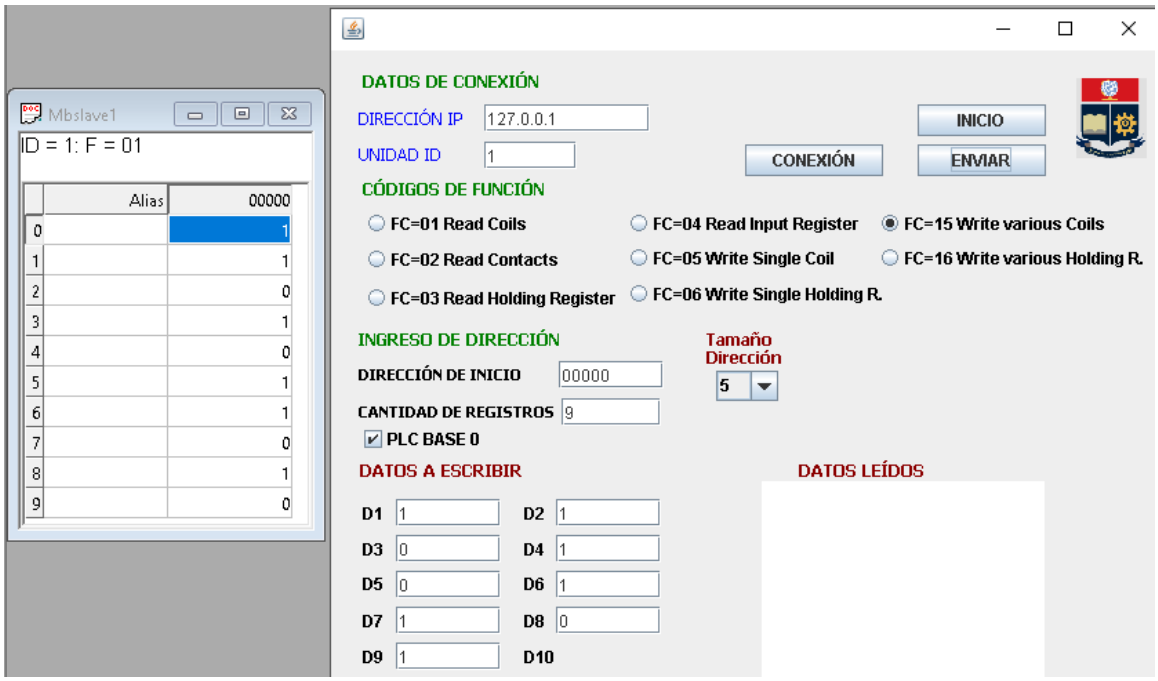


Figura 3.7. Prueba Código de función 15 con Simulador de PLC Modbus

Finalmente, en la Figura 3.8 se realiza la escritura de 3 Holding Register desde la dirección 40000. Los datos escritos son tipo flotante con un ordenamiento de bytes tipo Big Endian y se escriben los valores de 86.5, 6 y 1.4

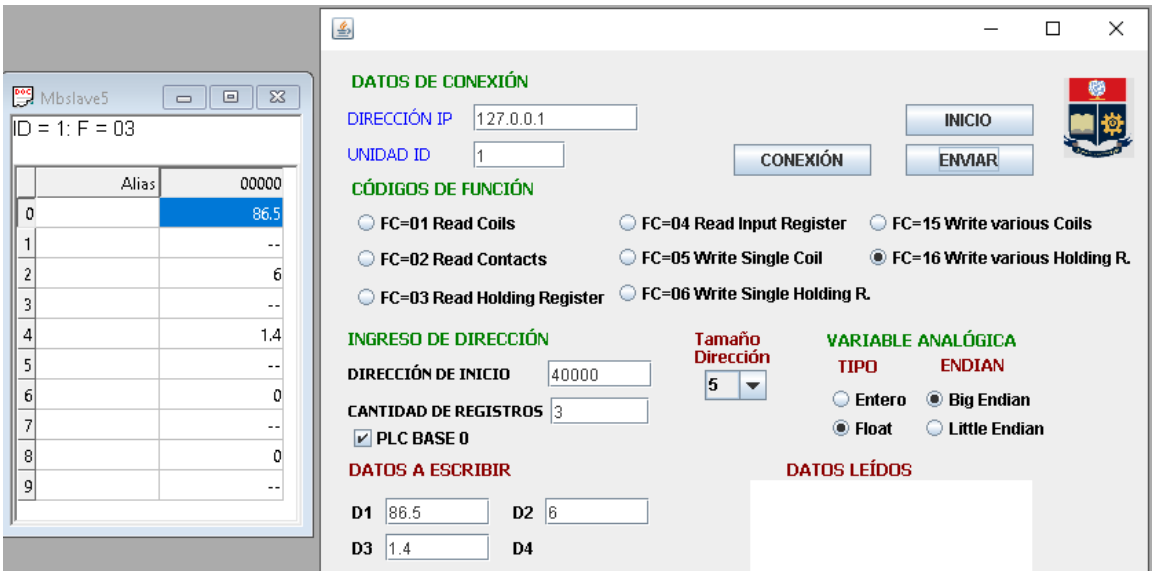


Figura 3.8. Prueba Código de función 16 con Simulador de PLC Modbus

Al realizar las pruebas con el Simulador Slave Modbus se pudo validar el funcionamiento de los 8 códigos de función del protocolo, así como su funcionamiento con números Enteros y flotantes.

3.1.2. Pruebas realizadas con PLCs ML1400 y M580

En esta subsección se realizan pruebas individuales para ciertos códigos de función con los dispositivos ML1400 y M580

El PLC M580 tiene las siguientes características:

- Dirección IP (192.168.10.1).
- Base 1.
- Tamaño de registro 5.
- Variable analógica tipo entero.

En la Figura 3.9 se muestra la prueba realizada para el código de función 03 con el PLC M850, en la parte derecha se encuentra la interface del servidor desarrollado y en la izquierda el Online del programa en Unity Pro. Se realiza la lectura de dos Holding Register desde la dirección 40004, como se observa el valor del PLC es el mismo del software Unity Pro.

Figura 3.9. Prueba Código de función 03 con PLC M580

En la Figura 3.10 se muestra la prueba para el código de función 15, en donde se realiza la escritura de 4 Coils desde la dirección 1. En la parte derecha se muestra la interface del software y en la izquierda el programa de Unity Pro.

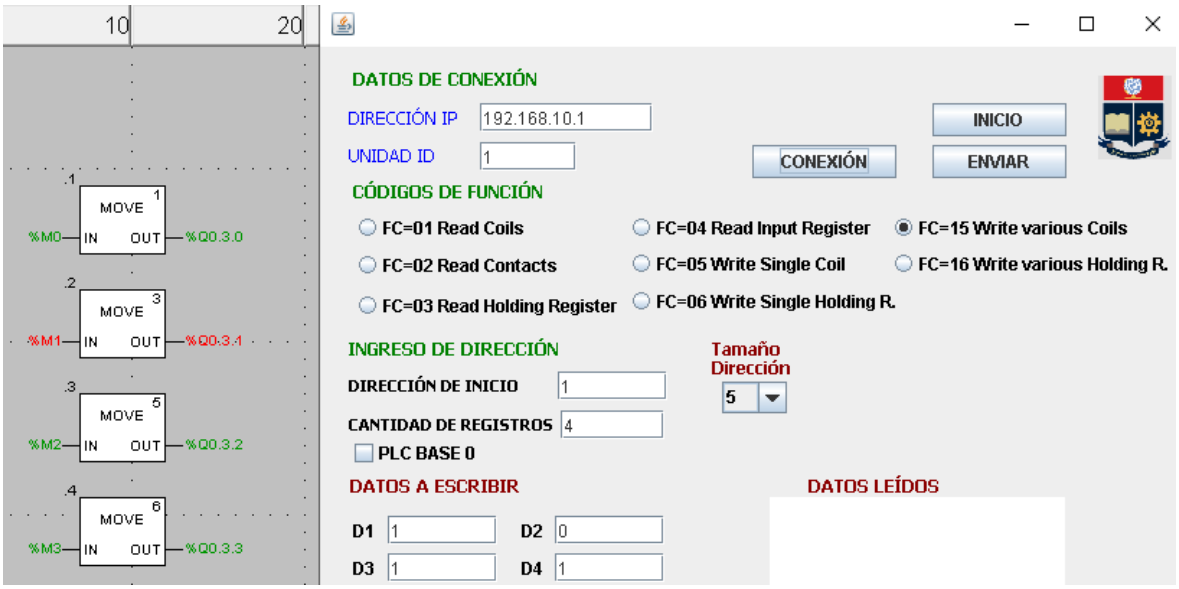


Figura 3.10. Prueba Código de función 15 con PLC M580

A continuación, se realiza pruebas con el PLC ML1400 el cual tiene las siguientes características:

- Dirección IP: (192.168.10.30).
- Base 1.
- Tamaño de registro 5
- Variable analógica tipo entero.

En la Figura 3.11 se muestra la configuración de la prueba realizada con el PLC ML1400, se realiza la escritura de un Coil con el estado de ON (1) en la dirección 1. Se valida el funcionamiento al verificar en el PLC la activación de la salida digital asignada a este registro en la programación del PLC.

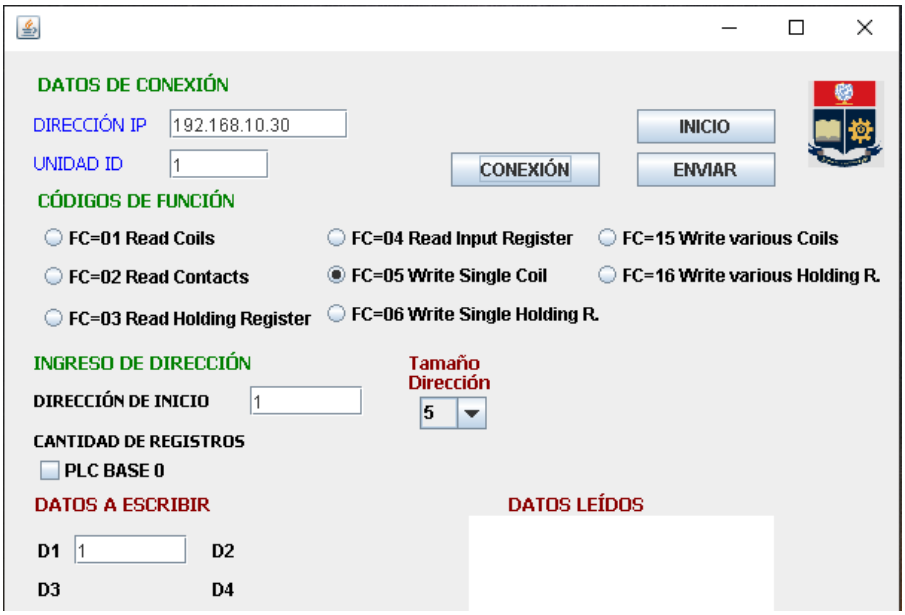


Figura 3.11. Prueba código de función 05 con PLC ML1400

Finalmente en la Figura 3.12 se muestra la configuración de la prueba realizada para el código de función 6 en el PLC ML1400. La prueba consiste en la escritura de un Holding Register con el valor de 100. Se valida el funcionamiento al medir la salida analógica del PLC.

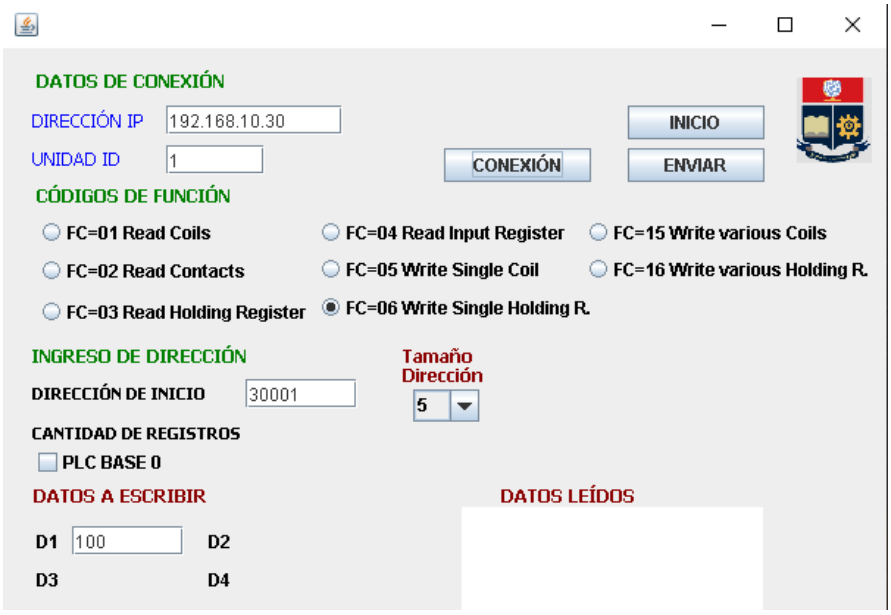


Figura 3.12. Prueba código de función 06 con PLC ML1400

De acuerdo a las pruebas realizadas anteriormente, se pudo validar el funcionamiento de los 8 códigos de función básicos del protocolo Modbus. Mostrando que se puede emplear

los códigos de función implementados con cualquier dispositivo industrial que maneje una comunicación Modbus TCP, la interfaz de este modo permite que ingresando los parámetros necesarios para cada código de función se prueben los códigos de función.

3.2. Prueba en modo Automático

El segundo modo, modo Automático, realiza una comunicación automática entre el dispositivo industrial y el servidor de datos industrial desarrollado. La ejecución del código de función es de acuerdo al primer dígito de la dirección del registro Modbus, como se muestra en la Tabla 3.1.

Tabla 3.1. Relación de dirección de registro Modbus con código de función a ejecutarse

Dirección	Tipo de variable	Código de función
0XXX	Coil	05
1XXX	Contact	02
3XXX	Input Register	04
4XXX	Holding Register	03 (Entero) y 16(Flotante)

En la sección 3.2.1 se muestra las pruebas con un proceso básico que utiliza los 4 bloques de memoria Modbus, esta sección muestra la prueba con dos PLC que se encuentran en red. La sección 3.2.2 muestra un proceso más complejo con el proceso de fabricación de chocolate, el cual tiene tres PLC para el control y monitoreo de los diferentes subprocesos y mediante un HMI realizado en InTouch se realiza la visualización y control de las variables del proceso.

3.2.1. Prueba de los códigos empleados con un proceso básico

El proceso básico con el cual se realiza la prueba es un Marcha – Paro y adicionalmente tiene una entrada analógica y una salida analógica. De esta manera se tiene los cuatro tipos de variable (Contact, Coil, Input Register y Holding Register). Ambos PLC pertenecen a la marca Allen Bradley. En la Figura 3.13 se muestra su configuración.

Figura 3.13. Configuración PLC, a) Micro 850 y b) ML1400

En la Tabla 3.2 se muestra las direcciones Modbus de cada PLC.

Tabla 3.2. Direcciones Modbus de los PLC

	Micro 850	ML1400
Variable	**Dirección Modbus**	**Dirección Modbus**
START	1	1
STOP	2	2
STATUS	1000001	100001
TANQUE	3000001	300001
VALOR	4000001	400001

A continuación se muestra las pruebas realizadas, en la Figura 3.14 para el Micro 850 y en la Figura 3.15 para el ML1400. En las figuras se observa en la derecha el HMI realizado en InTouch, en la izquierda parte superior la tabla en MySQL y en la parte inferior la pestaña DATOS de la interfaz en Java.

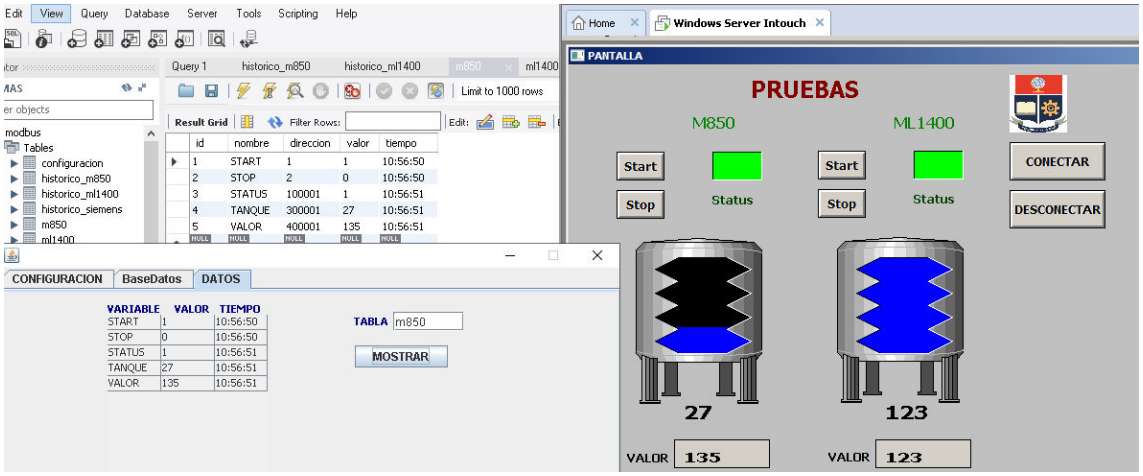


Figura 3.14. Prueba con Micro 850.

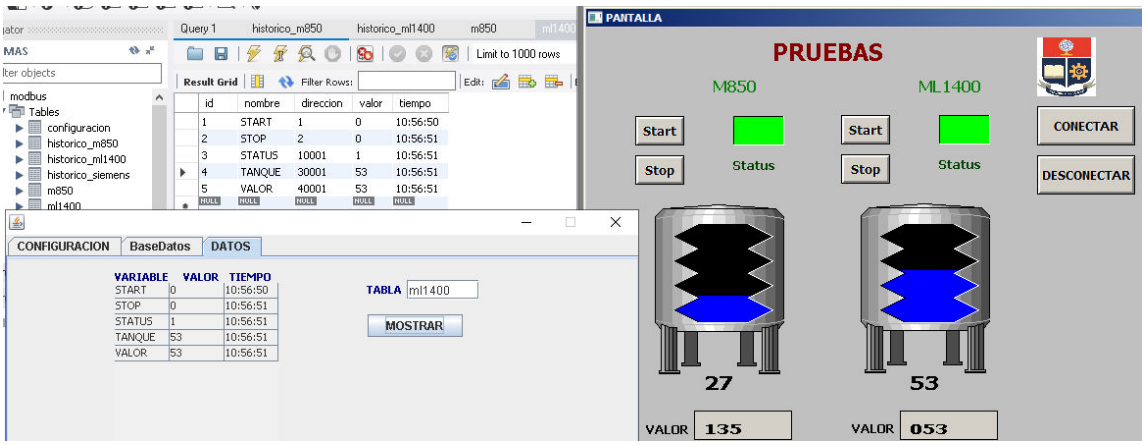
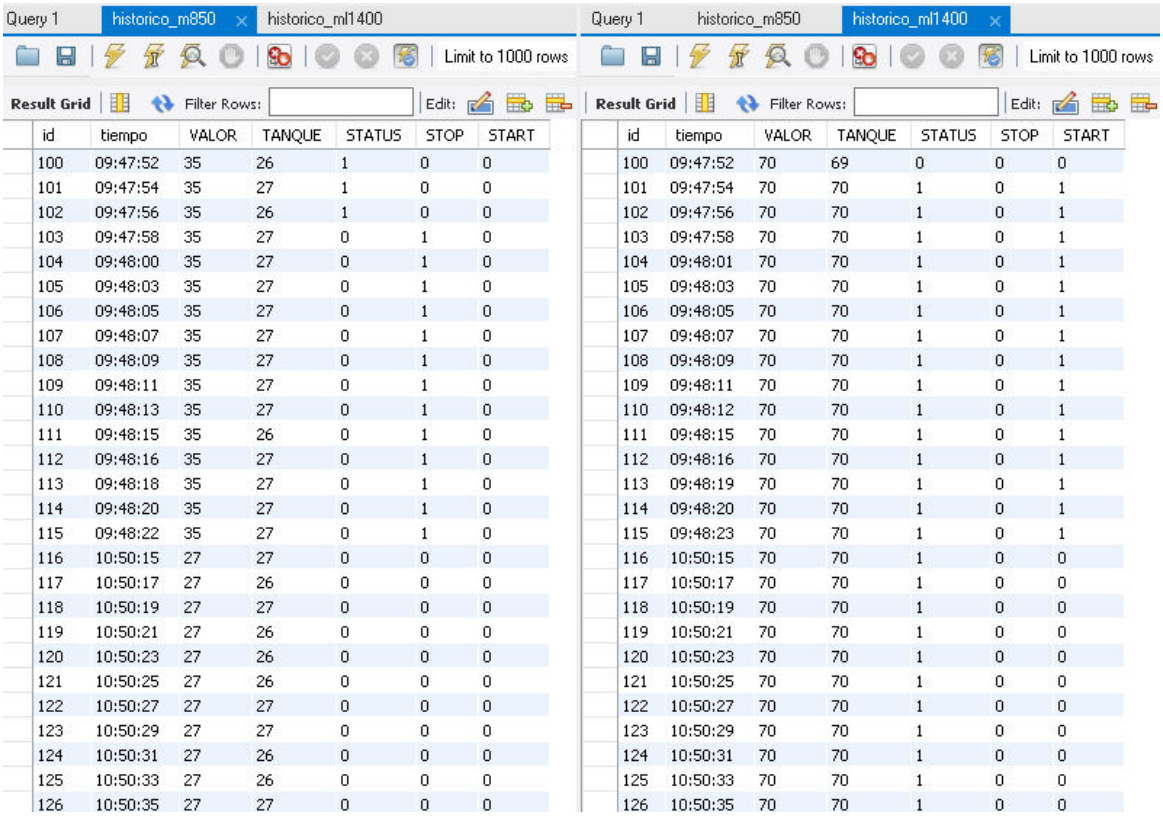


Figura 3.15. Prueba con ML1400

En las figuras anteriores se puede observar, que los valores mostrados en la tabla de MySQL y la de la interface de Java son los mismos valores que se presentan en la interface de InTouch. Por consiguiente se comprueba que se tiene una comunicación entre los tres programas y que los mismos presentan los mismos valores, adicionalmente en los PLC respectivos se activa la salida digital.



Query 1 | historico_m850 | historico_ml1400 — Limit to 1000 rows — Result Grid — Filter Rows: — Edit:

id	tiempo	VALOR	TANQUE	STATUS	STOP	START
100	09:47:52	35	26	1	0	0
101	09:47:54	35	27	1	0	0
102	09:47:56	35	26	1	0	0
103	09:47:58	35	27	0	1	0
104	09:48:00	35	27	0	1	0
105	09:48:03	35	27	0	1	0
106	09:48:05	35	27	0	1	0
107	09:48:07	35	27	0	1	0
108	09:48:09	35	27	0	1	0
109	09:48:11	35	27	0	1	0
110	09:48:13	35	27	0	1	0
111	09:48:15	35	26	0	1	0
112	09:48:16	35	27	0	1	0
113	09:48:18	35	27	0	1	0
114	09:48:20	35	27	0	1	0
115	09:48:22	35	27	0	1	0
116	10:50:15	27	27	0	0	0
117	10:50:17	27	26	0	0	0
118	10:50:19	27	27	0	0	0
119	10:50:21	27	26	0	0	0
120	10:50:23	27	26	0	0	0
121	10:50:25	27	26	0	0	0
122	10:50:27	27	27	0	0	0
123	10:50:29	27	27	0	0	0
124	10:50:31	27	26	0	0	0
125	10:50:33	27	26	0	0	0
126	10:50:35	27	27	0	0	0

Query 1 | historico_m850 | historico_ml1400 — Limit to 1000 rows — Result Grid — Filter Rows: — Edit:

id	tiempo	VALOR	TANQUE	STATUS	STOP	START
100	09:47:52	70	69	0	0	0
101	09:47:54	70	70	1	0	1
102	09:47:56	70	70	1	0	1
103	09:47:58	70	70	1	0	1
104	09:48:01	70	70	1	0	1
105	09:48:03	70	70	1	0	1
106	09:48:05	70	70	1	0	1
107	09:48:07	70	70	1	0	1
108	09:48:09	70	70	1	0	1
109	09:48:11	70	70	1	0	1
110	09:48:12	70	70	1	0	1
111	09:48:15	70	70	1	0	1
112	09:48:16	70	70	1	0	1
113	09:48:19	70	70	1	0	1
114	09:48:20	70	70	1	0	1
115	09:48:23	70	70	1	0	1
116	10:50:15	70	70	1	0	0
117	10:50:17	70	70	1	0	0
118	10:50:19	70	70	1	0	0
119	10:50:21	70	70	1	0	0
120	10:50:23	70	70	1	0	0
121	10:50:25	70	70	1	0	0
122	10:50:27	70	70	1	0	0
123	10:50:29	70	70	1	0	0
124	10:50:31	70	70	1	0	0
125	10:50:33	70	70	1	0	0
126	10:50:35	70	70	1	0	0

Figura 3.16. Tabla de históricos de PLC Micro 850 y ML1400

En la Figura 3.16 se muestran los históricos de los PLCs respectivos, estos históricos almacenan el tiempo y los valores de las variables configuradas, como se observa en las Tablas el tiempo de actualización de las variables será de acuerdo al establecido en la configuración, en este caso es de 2 ms para cada PLC. Y mostrará todos los cambios que se dieron en el tiempo.

3.2.2. Implementación de prototipo para Laboratorio de Redes Industriales

El prototipo desarrollado para el Laboratorio de Redes Industriales utiliza el modo de funcionamiento automático, con el proceso de fabricación del chocolate y un HMI desarrollado en InTouch. La Figura 3.17 muestra la comunicación del prototipo a implementarse. La base de datos en MySQL tendrá una tabla para cada PLC configurado, en donde se almacenarán las variables en tiempo real y tres tablas que funcionan como históricos para cada PLC configurado.

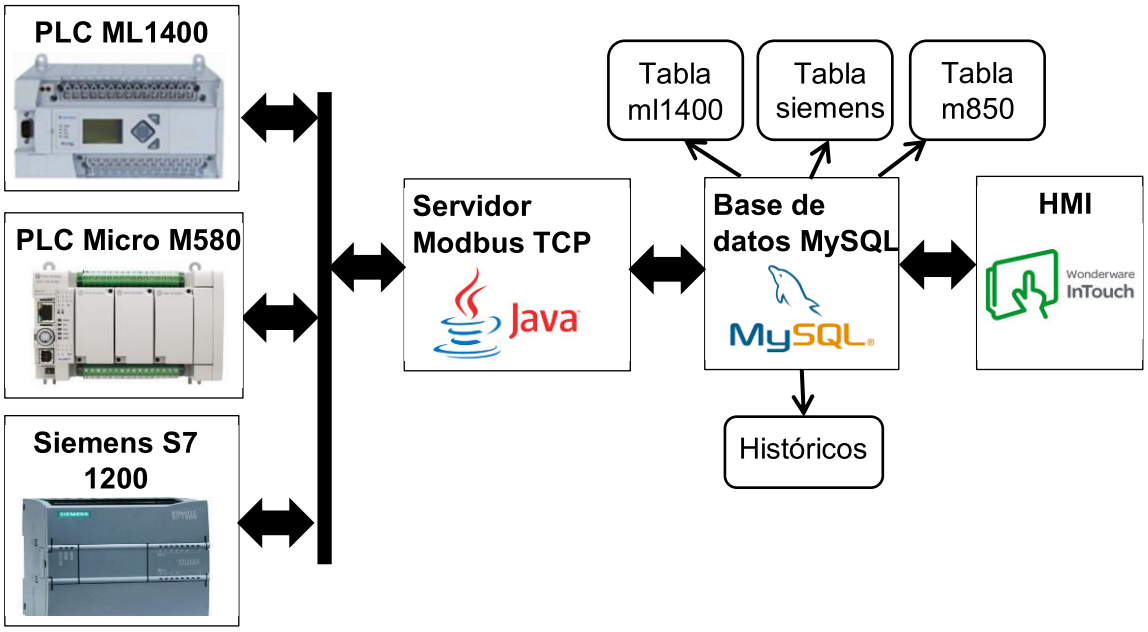


Figura 3.17. Arquitectura del prototipo

A continuación, se realiza una descripción del proceso de fabricación de chocolate y las variables que se utilizan en el mismo.

3.2.2.1. Descripción del proceso de fabricación de Chocolate

La elaboración del chocolate una vez obtenido el licor de cacao consta de las siguientes etapas: mezclado, refinado, conchado, templado y dosificado, las cuales se detallan a continuación.

Mezclado: En este subproceso se realiza una mezcla del licor de cacao con azúcar y leche. Esta mezcla se encuentra a una temperatura de aproximadamente 60°.

Refinado: La mezcla obtenido anteriormente se pasa por una refinadora la cual consta generalmente de 5 rollos colocados horizontal, los cuales giran triturando la mezcla reduciéndolo a un polvo muy fino, con una dimensión de 15 - 30 micrones.

Conchado: En esta fase se eliminan compuestos ácidos del cacao. La mezcla anterior se recubre homogéneamente con manteca, además de otros ingredientes de acuerdo al tipo de chocolate. Este proceso dura aproximadamente ocho horas y al terminar el conchado la mezcla sale a una temperatura de aproximadamente 45 ° C.

Templado: El chocolate se coloca en tanques, donde se encuentra a 45 ° C, luego se lo enfría hasta 28 °C y posteriormente se sube la temperatura hasta 29.5°C. Antiguamente se realizaba el templado manualmente mediante un maestro chocolatero.

Dosificado: Consiste en distribuir el chocolate templado en los moldes del chocolate. Estos moldes posteriormente pasaran a una etapa de enfriamiento para conseguir el chocolate sólido.

Para el control y visualización de este proceso se realiza la medición de algunas variables como el nivel de los tanques y temperatura en los subprocesos respectivos, para el control se utilizan actuadores como electroválvulas, motores y quemadores. En la Figura 3.18 se muestra los subprocesos de la elaboración del chocolate con la señalización de donde se encuentran los diferentes elementos. V representa a electroválvula, T un medidor de temperatura, C un quemador, AG un agitador y M a motor. Adicionalmente los tanques tienen medidores de nivel.

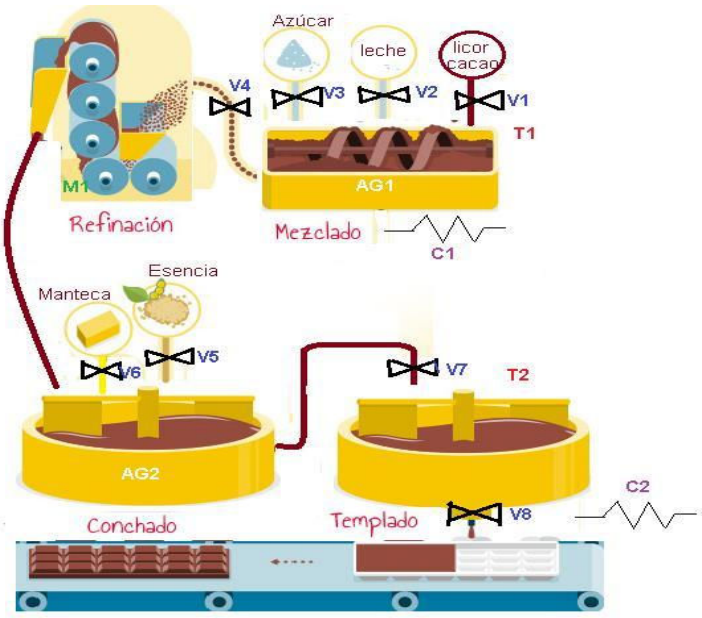


Figura 3.18. Diagrama del proceso de elaboración de chocolate

3.2.2.2. Descripción de PLCs del subproceso y variables

Para realizar las pruebas de funcionamiento se emplearon tres PLC, que manejen los cuatro tipos de variables (Coil, Contact, Input Register y Holding Register). Los PLCs empleados fueron: Micro850 de Allen Bradley para el subproceso de mezclado, el ML1400 de Allen Bradley para los subprocesos de refinado y conchado y por último el PLC S7-1200 de Siemens para el subproceso de templado.

En las tablas 3.3, 3.4, y 3.4 se detalla la variable y dirección de las variables utilizadas para cada uno de los PLC.

Tabla 3.3. Direcciones Modbus PLC micro 850

Variable	Dirección Física	Programa	HMI
Electroválvula V1	IO_EM_D0_00	000000	1
Electroválvula V2	IO_EM_D0_01	000001	2
Electroválvula V3	IO_EM_D0_02	000002	3
Quemador C1	IO_EM_D0_03	100001	100001
Tanque lleno TL1	IO_EM_DI_00	100002	100002
Velocidad AG1	IO_P2_AO_00	400001	400001
Temperatura T1	IO_P1_AI_00	300001	300001
Nivel N1	IO_P1_AI_01	300003	300003
Nivel N2	IO_P1_AI_02	300005	300005
Nivel N3	IO_P1_AI_03	300007	300007

El nivel N1 es la medición del nivel del tanque de mezclado y la variable tanque lleno TL es igual del tanque de mezclado, las mediciones de nivel N2 y N3 pertenecen al nivel de los tanques de licor de cacao y leche respectivamente.

Tabla 3.4. Direcciones Modbus PLC ML1400

Variable	Dirección Física	Programa	HMI
Electroválvula V4	O:0/0	00001	1
Electroválvula V5	O:0/1	00002	2
Electroválvula V6	O:0/2	00003	3
Estado M1	O:0/3	10001	10001
Tanque lleno TL2	I:0.0/1	10002	10002
Velocidad AG2	O:1.0	40001	40001
Nivel N4	I:1.0	30001	30001
Nivel N5	I:1.1	30003	30003

Se realiza la medición de nivel de los tanques de conchado, manteca y esencia. Igual que en el subproceso anterior se da la medición de cuando el tanque está lleno como una variable ON/OFF.

Tabla 3.5. Direcciones Modbus PLC S7-1200

Variable	Dirección Física	Programa	HMI
Electroválvula V7	%Q0.0	%Q0.0	1
Electroválvula V8	%Q0.1	%Q0.1	2
Quemador C2	%I0.0	%I0.0	10001
Tanque lleno TL3	%I0.1	%I0.1	10002
Temperatura T3	%IW.96	%IW.96	30049
Nivel N7	%IW.98	%IW.98	30051

3.2.2.3. Prueba realizada

En esta subsección se muestran las pruebas realizadas para el proceso de elaboración de chocolate mencionado anteriormente.

La Figura 3.19 muestra la configuración en el servidor para el PLC ML1400, esta configuración se la realiza para cada PLC a conectar, en la parte izquierda se encuentra la configuración de los datos del PLC como dirección IP, tiempo de muestreo, tamaño de registro, entre otros y en la parte de derecha la configuración de las variables que se van a utilizar en el proceso tanto nombre como dirección.

Figura 3.19. Configuración del PLC

A continuación se presenta la interface realizada en InTouch, en la Figura 3.20 se encuentra la pantalla con el proceso completo y la posibilidad de ir a las ventanas de los diferentes subprocesos. El botón CONECTAR inicia la comunicación con la base de datos en MySQL y el botón DESCONECTAR finaliza la comunicación con la misma. La actualización de los datos del proceso de InTouch con la Base de datos se la da cada 1 milisegundo.

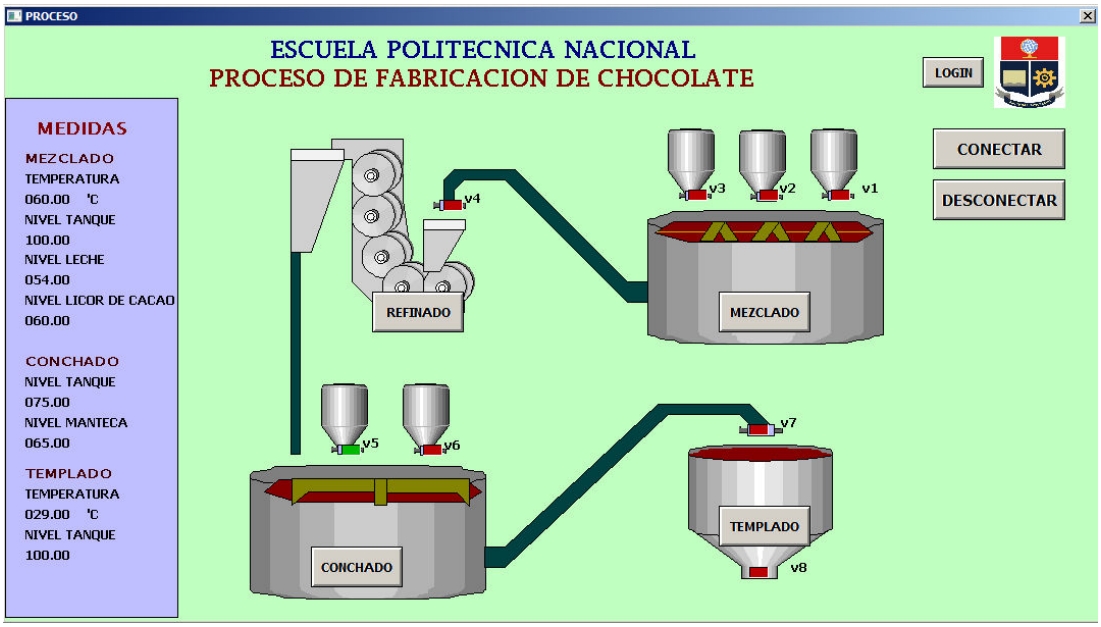
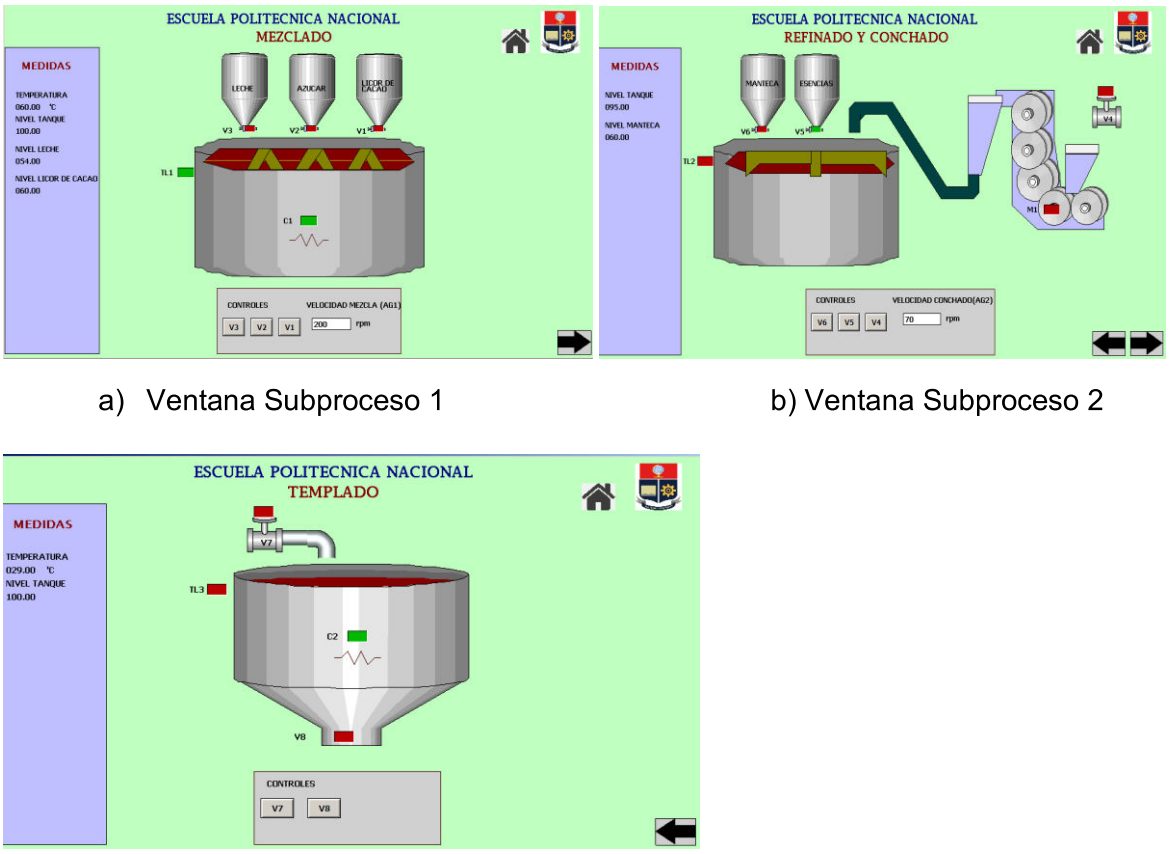


Figura 3.20. HMI en InTouch, ventana principal

La Figura 3.21 muestra las ventanas de los Subprocesos manejados por los tres PLC. Cada una de las ventanas cuenta con el gráfico que describe el subproceso, flechas de navegación, sección para visualización de las variables y variables a controlar. Además de un botón que permite regresar a la pantalla principal mostrada en la Figura 3.20.



a) Ventana Subproceso 1

b) Ventana Subproceso 2

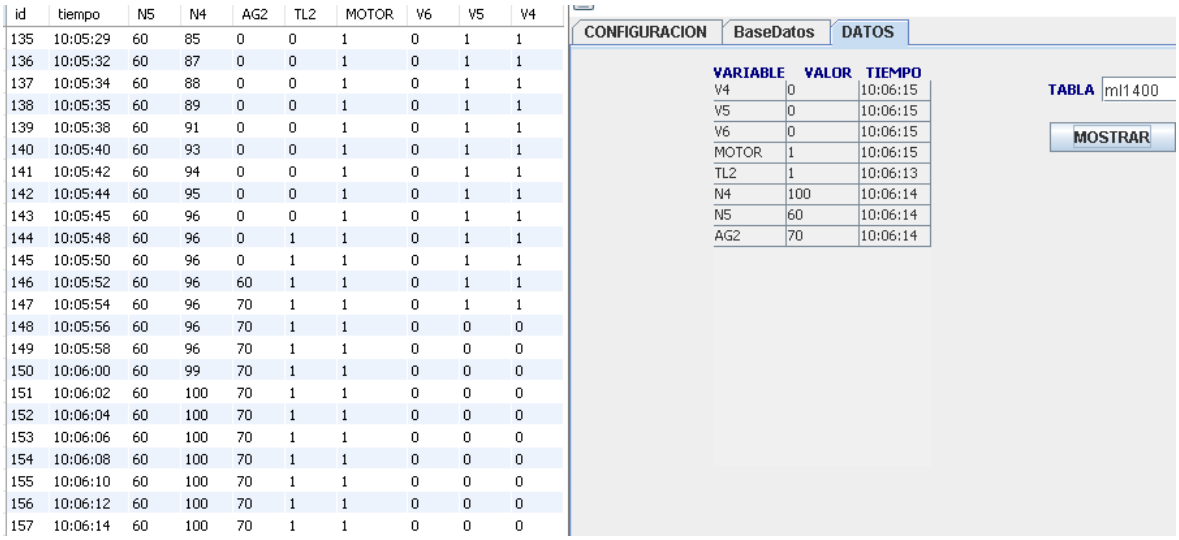
c) Ventana Subproceso 3

Figura 3.21. Ventanas de los Subprocesos

Mediante el proceso de fabricación de chocolate, se pudo realizar la implementación de un sistema SCADA y validar el funcionamiento del segundo modo, ya que tanto en el HMI desarrollado como en los PLC se obtuvieron los mismos valores. Adicionalmente se comprobó el funcionamiento de las sentencias SQL para poder realizar la comunicación entre la Interfaz realizada en InTouch con la base de datos, cabe mencionar que el empleo del lenguaje SQL se dio en un script de aplicación y mediante el uso de los Bind List se asoció el Tagname con la columna correspondiente en la base de datos.

En la Figura 3.22 se muestra, en la parte izquierda el histórico almacenado en MySQL y en la parte derecha la tabla en tiempo real en la pestaña de DATOS en la interface del servidor desarrollado en Java, ambas tablas se almacenan en MySQL, y cada PLC configurado tendrá estas dos tablas.

Al realizar la prueba se dejó funcionando por un tiempo considerable hasta comprobar que una vez almacenado los 1000 datos en la tabla de histórico, los siguientes valores se guarden en el lugar de los primeros valores, es decir el dato 10001 sustituirá al primer valor leído, el siguiente sustituirá al segundo valor y así sucesivamente, de esta manera se obtienen los últimos 1000 valores del proceso, remplazando los anteriores por nuevos.



id	tiempo	N5	N4	AG2	TL2	MOTOR	V6	V5	V4
135	10:05:29	60	85	0	0	1	0	1	1
136	10:05:32	60	87	0	0	1	0	1	1
137	10:05:34	60	88	0	0	1	0	1	1
138	10:05:35	60	89	0	0	1	0	1	1
139	10:05:38	60	91	0	0	1	0	1	1
140	10:05:40	60	93	0	0	1	0	1	1
141	10:05:42	60	94	0	0	1	0	1	1
142	10:05:44	60	95	0	0	1	0	1	1
143	10:05:45	60	96	0	0	1	0	1	1
144	10:05:48	60	96	0	1	1	0	1	1
145	10:05:50	60	96	0	1	1	0	1	1
146	10:05:52	60	96	60	1	1	0	1	1
147	10:05:54	60	96	70	1	1	0	1	1
148	10:05:56	60	96	70	1	1	0	0	0
149	10:05:58	60	96	70	1	1	0	0	0
150	10:06:00	60	99	70	1	1	0	0	0
151	10:06:02	60	100	70	1	1	0	0	0
152	10:06:04	60	100	70	1	1	0	0	0
153	10:06:06	60	100	70	1	1	0	0	0
154	10:06:08	60	100	70	1	1	0	0	0
155	10:06:10	60	100	70	1	1	0	0	0
156	10:06:12	60	100	70	1	1	0	0	0
157	10:06:14	60	100	70	1	1	0	0	0

Figura 3.22. Histórico y tabla en tiempo real del PLC ML1400

3.3. Comparación del servidor de datos industrial realizado con el software Wonderware MBTCP DAServer

En esta subsección se realiza una comparación entre las funciones que tiene el segundo modo del servidor de datos industrial realizado en Java con el software comercial Wonderware MBTCP DAServer. Las características mostradas en la Tabla 3.6 se sacaron de la Guía de usuario de MBTCP DAServer[8]. En la Tabla 3.6 se observa la comparación entre los dos programas.

Tabla 3.6. Comparación entre servidor desarrollado y Wonderware MBTCP DAServer

Descripción	MBTCP DAServer	Servidor de datos industrial desarrollado
Sistema operativo	Windows	Cualquiera
Conectividad con otras aplicaciones	Aplicaciones de Windows que actúen como cliente DDE, SuiteLink u OPC	Aplicaciones que manejen lenguaje SQL para acceder a datos en MySQL
Controladores compatibles	Controladores compatibles con Modbus a través del protocolo TCP/IP o a través de RS232/RS485 mediante un puente Modbus	Controladores Modbus a través del protocolo TCP/IP
Licencia	Licencia pagada, Modo	Gratuita

	demo permite su uso por 120 minutos	
Tamaño de dirección	4, 5 y 6 dígitos	4, 5 y 6 dígitos
Tiempo de muestreo de dispositivos	Configurable para el servidor	Configurable para cada PLC
Tiempo de espera de respuesta	Configurable, Por defecto 3 sec	10 msec
Dirección IP	Ingreso de dirección IP	Ingreso de dirección IP y permite comprobar conexión a la dirección.
Número de puerto	Por defecto 502, Configurable	Puerto 502, No configurable
Estructura de datos tipo Long	Si esta seleccionado usa el orden de datos del Software de programación	No trabaja con datos de tipo Long
Estructura de datos de tipo real	Si esta seleccionado usa el orden de datos del Software de programación	Permite elegir el orden de datos entre Big Endian y Little Endian.
Formato orden de bit	Permite elegir entre de izquierda a derecha (B1 B2…B16) y de derecha a izquierda (B16 B15...B1), por defecto está el primero	Permite elegir entre Big Endian y Little Endian.
Tipo de registro	Binario o BCD. Por defecto está Binario	Binario
Variables tipo string	Longitud total, estilo C, Pascal	ASCII
Configuración de registros	Configurable en pestaña Device Item , nombre y dirección de variable	Configurable en pestaña Base de datos, nombre y dirección de variable
Asociación de PLC con servidor	Configurable en pestaña Device group, nombre único para cada PLC	Configurable en campo Tabla de datos, nombre único para cada PLC
Visualización de datos	Visualización en consola	Visualización en pestaña DATOS
Base de datos	Base de datos en SQL Server	Base de datos en MySQL
Guardar configuración	Permite guardar la configuración de los dispositivos	Permite guardar la configuración de los dispositivos

El objetivo del trabajo realizado fue la creación de un servidor de datos industrial con protocolo Modbus TCP/IP el cual además pueda comunicarse con aplicaciones de Windows, como se observa en las pruebas realizadas el servidor cumple con los parámetros establecidos y mediante MySQL se puede conectar el servidor desarrollado con aplicaciones de Windows y cualquier otra que maneje un lenguaje SQL. Adicionalmente los dos modos permiten al usuario realizar una comunicación automática y de manera continua, así como probar individualmente los principales códigos de función del protocolo.

Finalmente, habiendo validado el funcionamiento del servidor y realizado una comparación con un software comercial se saca costos de la herramienta computacional desarrollada, mostrada en la Tabla 3.7.

Tabla 3.7. Costos de Herramienta Computacional desarrollada

Descripción	**Software**	**Costo**	**Observaciones**
Driver de comunicación	JAVA con IDE de Eclipse	$ 0	Licencia gratuita
Base de datos	MySQL 5.0	$ 0	Licencia gratuita
	Xampp	$ 0	Licencia gratuita
Interfaz de operador	Aplicación de Windows que maneje sentencias SQL	--	El costo dependerá del software que se utilice para realizar la interface de operador.
Desarrollo del software		$2400	Horas invertidas * 5 Usd.
Instalación		$50	Instalación de programas y configuración.

En la Tabla 3.7, se detalla el costo de desarrollo del driver de comunicación industrial. Entre los rubros considerados, el m[as representativo es el del tiempo empleado en el desarrollo del software, además se incluye el costo de instalación en el equipo del usuario final. En lo referente al costo de desarrollo es un valor que no se cargará al usuario final ya que es una inversión que se recuperará a largo plazo con el costo de instalación, de allí que la inversión inicial será únicamente la de instalación. Si se compara la Tabla 3.7 con la Tabla 1.1 se presenta una reducción en el costo respecto a los softwares comerciales, en donde si bien con un software comercial se tiene el soporte técnico y mejora en versiones posteriores, con el software desarrollado al poseer el código del programa se le pueden realizar modificaciones o mejoras futuras también.

4. CONCLUSIONES Y RECOMENDACIONES

De acuerdo a las pruebas realizadas del funcionamiento del servidor de datos industrial desarrollado en la sección anterior se han sacado las siguientes conclusiones y recomendaciones.

4.1. Conclusiones

- Se desarrolló un servidor de datos industrial para la comunicación de dispositivos Modbus TCP y aplicaciones de Windows, utilizando una base de datos como espacio de almacenamiento temporal, este desarrollo se realizó basado en software libre y de código abierto.

- Se realizó el estudio de la normativa del protocolo Modbus TCP, así como la estructura de los ocho principales códigos de función, mismos que permiten la lectura y escritura de los diferentes registros Modbus en dispositivos industriales con este protocolo. En el primer modo que se permitirá al usuario ingresar los parámetros necesarios para probar individualmente los ocho códigos de función, sin embargo en una comunicación automática, como es el caso del segundo modo, no fue necesario la implementación de los 8 códigos de función para una correcta comunicación sino únicamente los códigos 02, 03, 04, 05 y 16.

- La realización del servidor de datos industrial en el software libre Java tiene como ventaja la portabilidad de sus programas permitiendo su ejecución en cualquier sistema operativo y la interacción con un gestor de base de datos a través de lenguaje SQL.

- Se desarrolló un sistema de almacenamiento de los datos adquiridos durante la comunicación automática, a través de una base de datos usando el gestor de base de datos MySQL, lo cual permitió el almacenamiento en tiempo real, así como la creación de históricos. Mediante el almacenamiento de estos datos el servidor desarrollado permitirá a las PYMES mejorar sus procesos, esto justificando en que además del monitoreo de los datos en tiempo real, se dispondrá de información almacenada en un histórico dentro de la base de datos que puede ser usado para análisis fuera de línea.

- La interfaz desarrollada en Java del servidor de datos industrial permite al usuario manejar la herramienta y comprender su funcionamiento, ya que la misma se encuentra dividida en secciones que permiten ingresar los parámetros de configuración necesarios, tanto para el manejo del primer modo como del segundo

modo y además se puede observar los datos resultantes de la interacción entre el servidor de datos industrial desarrollado y el dispositivo Modbus, los cuales son en el caso del primer modo los datos leídos y en el segundo modo los datos en tiempo real visualizados en la pestaña DATOS.

- Mediante la realización de las pruebas descritas en el capítulo 3, se pudo validar el funcionamiento del servidor desarrollado, por lo que se puede afirmar que se ha implementado un servidor de bajo costo accesible para las PYMES o industrias de baja capacidad de inversión, dándoles acceso a un software similar a servidores de datos industriales comerciales, a un costo mínimo que cubra los gastos de instalación.

- Una ventaja importante del desarrollo de este driver de comunicación industrial, es que se dispone de un desarrollo local del cual se conoce su código de programación y en el que se podrán realizar futuras modificaciones o mejoras según se requiera.

4.2. Recomendaciones

- Se recomienda realizar drivers de comunicación para otros protocolos de comunicación industriales, ya que a pesar de que Modbus es uno de los protocolo industriales más utilizados existen otros protocolos tales como Profinet o Ethernet/IP que también tiene una amplia difusión

- Se recomienda realizar una aplicación en software libre en donde se realice HMIs industriales que pueda ser acoplado al trabajo realizado, para así permitir a las PYMES disminuir sus costos en sistemas relacionados con la automatización de sus procesos.

- Adicionalmente se recomienda tener en cuenta que en el caso de MySQL es importante no mantener abiertas muchas conexiones a la base de datos si no solo las necesarias, evitando sobrecargar la comunicación con la misma, lo que puede resultar en respuestas no adecuadas por parte de la base de datos.

- Se recomienda iniciar un plan piloto e instalar el servidor de datos realizado en una PYME, para así dar a conocer las ventajas de este tipo de sistemas que son factibles de implementar sin una considerable inversión económica inicial.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] H. Igor, J. Bohuslava, and J. Martin, “Proposal of communication standardization of industrial networks in Industry 4.0,” *INES 2016 - 20th Jubilee IEEE International Conference on Intelligent Engineering Systems, Proceedings*, pp. 119–124, 2016.

[2] B. Y. J. C. Warren, “Industrial Networking,” no. January, pp. 20–24, 2011.

[3] J. M. i Hernadez, “Software Libre Técnicamente Viable, Económicamente Sostenible y Socialmente Justo,” *Infonomia - Re Innovadores*, pp. 1–191, 2005.

[4] M. Hernández, “Desarrollo e implementación de una red de datos basada en Modbus y Ethernet para autómatas industriales,” Universidad de Sevilla, 2016.

[5] M. Castro *et al.*, *Comunicaciones industriales: Principios Básicos*. Madrid: UNED, 2012.

[6] KEPWARE, “Modbus Suite | OPC Server | Kepware.” [Online]. Available: https://www.kepware.com/en-us/products/kepserverex/suites/modbus-suite/. [Accessed: 14-Dec-2020].

[7] “Servidor MatrikonOPC para Modbus.” [Online]. Available: https://www.matrikonopc.es/drivers/opc-modbus.aspx. [Accessed: 17-Dec-2020].

[8] I. Invensys Systems, “MBTCP DAServer User ’ s Guide,” no. C, p. 130, 2013.

[9] Schneider Electric Software, “Wonderware Operations Integration - Supervisory Modbus MBTCP Server 4.” .

[10] R. Linares, “Evaluación de Sistemas SCADA libres,” Universidad Central Marta Abreu de Las Villas, 2014.

[11] W. Guamaní and F. Narváez, “Diseño e implementación de un sistema de monitorización para relés y medidores a través de software libre SCADA para la S/E 1 de la EERSA,” Escuela Superior Politécnica de Chimborazo, 2011.

[12] D. Aguirre, “Desarrollo de una herramienta computacional que contenga comunicación Modbus RTU y Modbus TCP para la implementación de sistemas de control supervisorio y adquisición de datos a bajo costo,” Escuela Politécnica Nacional, 2018.

[13] Modbus, “Modbus FAQ.” [Online]. Available: http://www.modbus.org/faq.php. [Accessed: 26-Mar-2020].

[14] Modbus-IDA, “MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b,” pp. 1–51, 2006.

[15] “Información Detallada sobre el Protocolo Modbus - National Instruments.” [Online]. Available: https://www.ni.com/es-cr/innovations/white-papers/14/the-modbus-protocol-in-depth.html. [Accessed: 26-Mar-2020].

[16] Organización de computadora 2014, “Punto Flotante - Estándar IEEE 754,” 2014.

[17] “La Guía del Punto Flotante - Números de punto flotante.” [Online]. Available: http://puntoflotante.org/formats/fp/. [Accessed: 06-Apr-2020].

[18] L. Yáñez, *Fundamentos de Programacion*, 1st ed. Madrid: Universidad Complutense de Madrid, 2014.

[19] J. Solano, “Lenguajes de Programación,” 2011. [Online]. Available: https://cmapspublic2.ihmc.us/rid=1NYCQ4X27-1DFZZL4-252X/lenguaje de alto nivel pdf.pdf. [Accessed: 06-Apr-2020].

[20] F. Ceballos, *El lenguaje de Programación Java*. 2001.

[21] J. Martínez, *Fundamentos de programación en JAVA*, vol. 1. Madrid: Universidad Complutense de Madrid, 2015.

[22] A. García Beltrán and J. M. Arranz, “Programación Orientada a Objetos con Java,” pp. 7–8, 2007.

[23] B. Mazón Olivo, J. Cartuche Calva, and W. Rivas Asanza, *Fundamentos de Programación Orientada a objetos en java*. Machala: Ediciones UTMACH, 2015.

[24] A. Gutiérrez, “BASES DE DATOS,” Mexico, 2010.

[25] Á. Robledano, “Qué es MySQL: Características y ventajas,” *OpenWebinars*, 2019. [Online]. Available: https://openwebinars.net/blog/que-es-mysql/. [Accessed: 09-Apr-2020].

[26] A. Sáez, “Aplicación del software Wonderware a simuladores industriales de procesos,” Universidad de Valladolid, 2017.

[27] “MySQL :: MySQL Connector/J 5.1 Release Notes :: Changes in MySQL Connector/J 5.1.47 (2018-08-17).” [Online]. Available: https://dev.mysql.com/doc/relnotes/connector-j/5.1/en/news-5-1-47.html. [Accessed: 12-Sep-2020].

ANEXOS

ANEXO A: MANUAL DE USUARIO

CONTENIDO

Contenido

A.1. INTRODUCCIÓN

El servidor de datos industrial desarrollado es una aplicación que puede ser ejecutado en cualquier sistema operativo, el cual permite realizar una comunicación con dispositivos industriales que manejen un protocolo Modbus TCP realizando una comunicación a través de un puerto Ethernet. Este servidor es compatible con dispositivos con direcciones de 4, 5 y 6 dígitos, así como dispositivos de trabajen con base 0 o base 1 en las direcciones Modbus.

El servidor de datos industrial desarrollado cuenta con dos modos de funcionamiento. El primero permite al usuario comprobar el funcionamiento de los 8 códigos de función básico del protocolo Modbus de manera individual, realizando el ingreso de los parámetros necesarios para cada código de función. Mientras el segundo modo posee una comunicación automática, misma que permite al usuario utilizar el servidor para la implementación de un sistema SCADA y poder mejorar el control y supervisión de un proceso. Este modo cuenta adicionalmente con una base de datos en MySQL la cual sirve como nexo entre el servidor desarrollado y aplicaciones en donde se realice el respectivo HMI del proceso, la comunicación con la base de datos se lleva a cabo a través de sentencias SQL, por lo cual la aplicación en donde se desarrolle el HMI debe manejar estas sentencia, como por ejemplo el software InTouch.

En las siguientes secciones del manual de usuario se presentan, los requisitos previos, se detallan las ventanas del servidor con su funcionamiento y configuración.

A.2. REQUISITOS PREVIOS

Para la utilización del modo maestro solo se requiere tener en red al servidor y el dispositivo industrial a conectar.

Para el modo automático se debe tener instalada en la máquina que funcionará como servidor los siguientes programas para la base de Datos:

- MySQL Workbench
- Xampp: permite habilitar la conexión de MySQL sin necesidad de abrir MySQL.
- MySQL Connector/ODBC 8.0

Adicionalmente es importante comprobar que los dispositivos a comunicarse se encuentren en red, así como también las máquinas en donde se encuentren el servidor y el HMI del proceso

A.3. Configuraciones para el modo automático

Para este modo es necesario realizar la configuración de la base de datos y del conector /ODBC 8.0 de MySQL

A.3.1 Creación de una nueva conexión en la base de datos

XAMPP

Para activar la conexión mediante Xampp se debe dar click en el botón Start, donde dice MySQL. Por defecto se encuentra en el puerto 3306 para MySQL, para cambiar el número de puerto realizar lo siguiente.

1. Dar clic en el botón Config. Como se muestra en la figura A.1

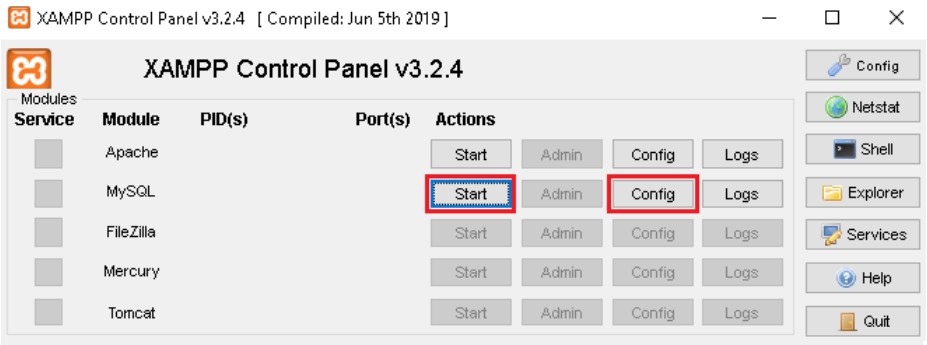


Figura. A.1 Ventana de XAMPP

2. Seleccionar my.ini y se abrira un block de notas. Se debe cambiar donde dice port por el puerto deseado. Ver figura A.2.

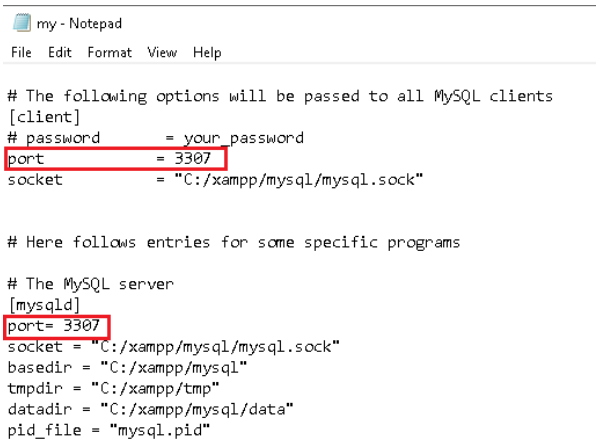


Figura. A.2 Ventana my.ini de XAMPP de MySQL

MySQL

Para realizar el cambio de puerto en MySQL Workbench se crea una nueva conección.

1. Abrir el programa de MySQL y dar click en Nueva conexión. Se abrirá la ventana mostrada en la figura A.3.
2. Llenar los campos con el nombre de la conexión, puerto, contraseña si se desea. En Hostname se puede colocar la que esta por defecto 127.0.0.1 o localhost.
3. Comprobar la conexión dando clic en Text Connection. Si el usuario root tiene contraseña pedira la contraseña.
4. Finalmente dar clic en OK.

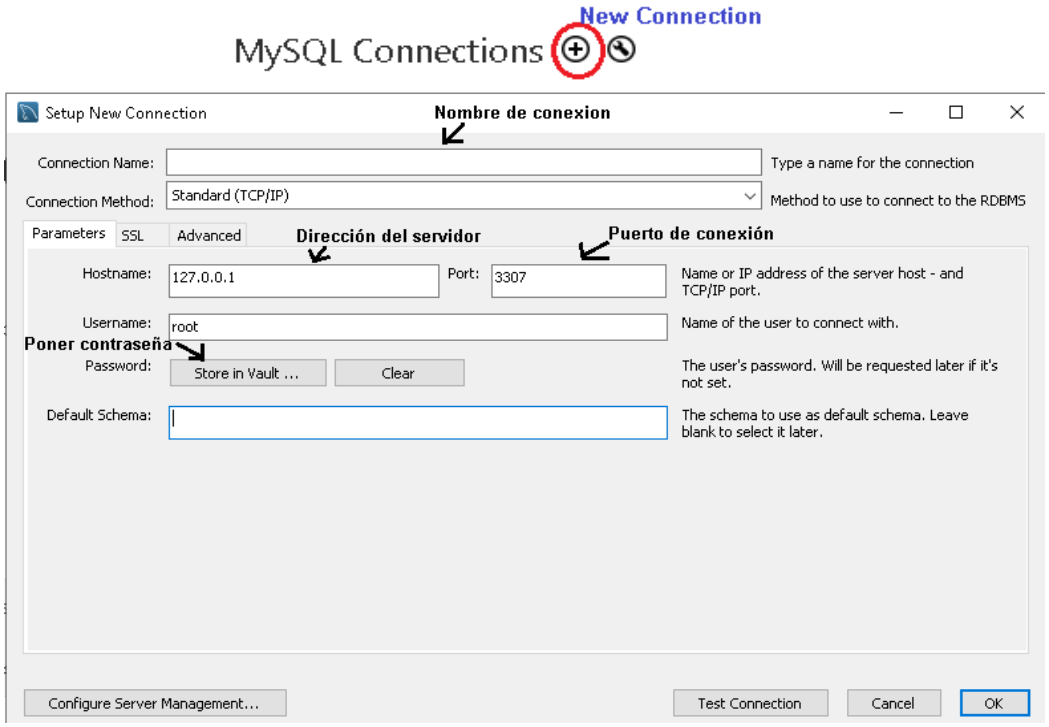


Figura. A.3 Ventana de configuracion nueva conexion

A.3.2 Base de datos en MySQL

Se debe crear en MySQL una base de datos en donde se almacenaran las tablas de los PLCs configurados en el servidor. Para la creación de la base de datos se debe dar clic en Crear a new schema, señalado en un círculo rojo en la Figura A.4, y en Name colocar **Modbus**. Finalmente se presionar el botón Apply.

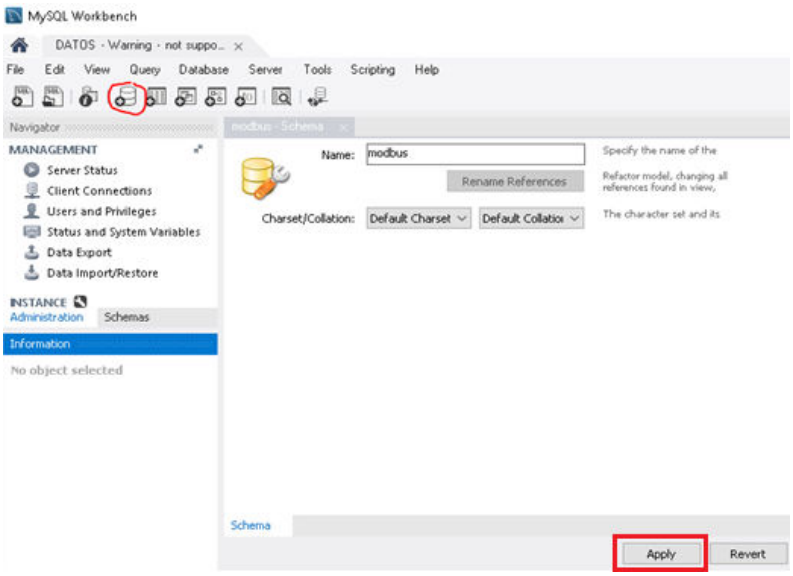


Figura. A.4 Creación de base de datos Modbus en MySQL

Nota: Para la utilización del modo automático se debe de activar previamente la conexión con la base de datos lo cual se puede realizar desde MySQL o desde Xampp.

A.3.3 Creación de usuario con permisos en MySQL

Para realizar la conexión con la base de datos desde Java y desde una aplicación de Windows, se debe crear un usuario con todos los permisos en MySQL. A continuación, se detallan los pasos a seguir.

1. Abrir MySQL como usuario root. Si se tiene problemas para abrir la conexión ver el punto A.3.4
2. Ir a **Administration** y seleccionar **Users and Privileges** como se muestra en la Figura A.5, donde se abrirá una pestaña con los usuarios de la base de datos.

Nota: Para saber que se encuentra en el usuario root, dar clic en un usuario y en la pestaña Login se podrá realizar un cambio, mostrándose en blanco los datos a cambiar, caso contrario aparecerán con fondo plomo y no se pueden modificar.

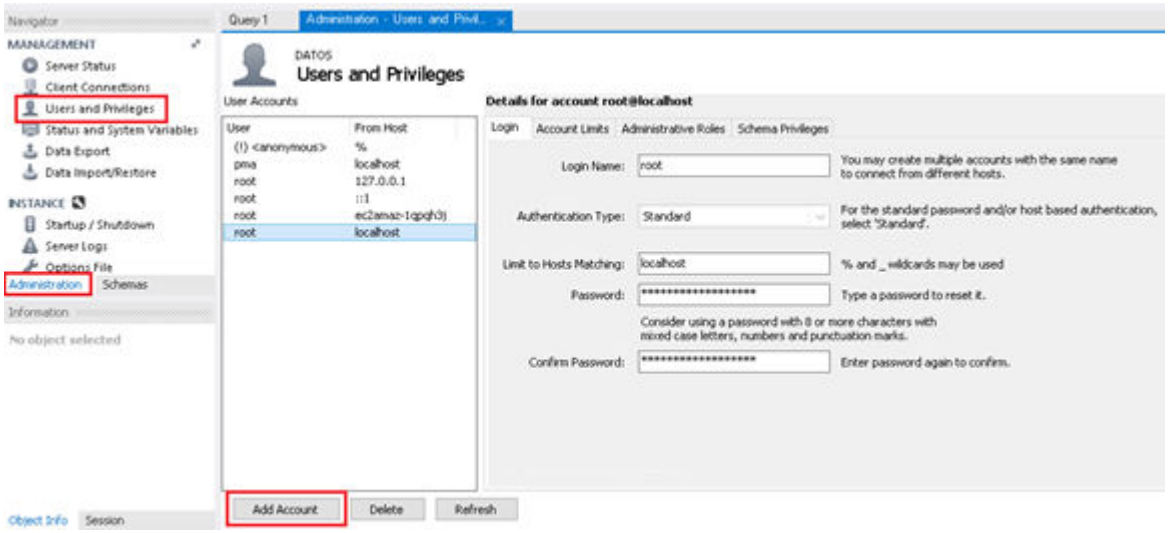


Figura. A.5 Ventana Users and Privileges

3. Dar clic en el botón **Add Account**. Se abrirá la ventana mostrada en la Figura A.6, llenar con los datos de usuario (user) y contraseña (1234) a crear.

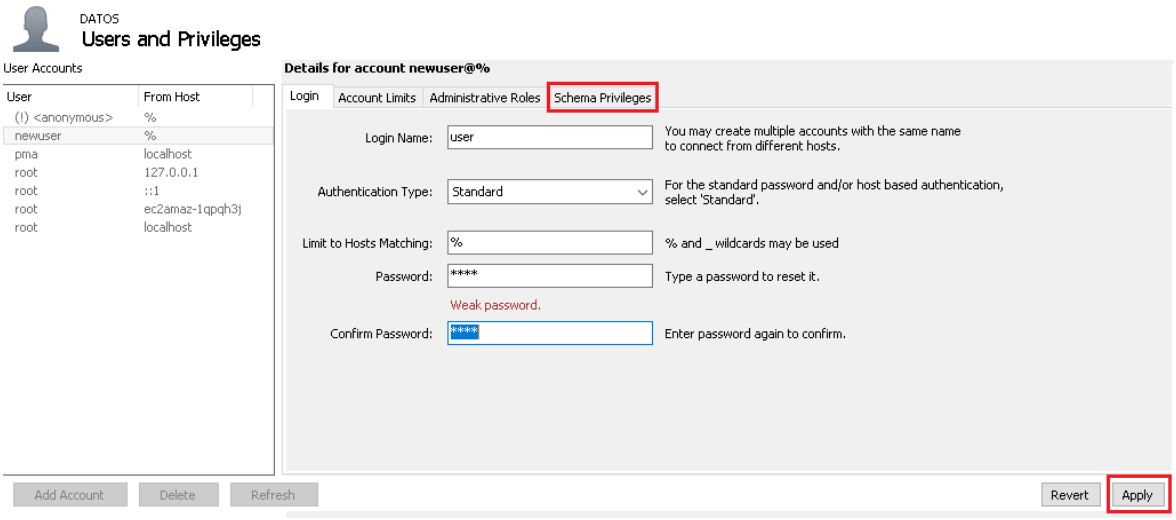


Figura. A.6 Pestaña Login, detalles de nuevo usuario

4. Dar clic en **Apply**. Con lo cual se creara el nuevo usuario

5. En la pestaña Schema Privileges, Figura A.7, dar clic en el botón **Add Entry**, se abre una pestaña dar clic en OK.

6. Dar clic en el botón **Select "ALL",** para dar todos los privilegios al usuario creado sobre los esquemas de la base de Datos. Después dar clic en el botón **Apply**, para guardar los cambios.

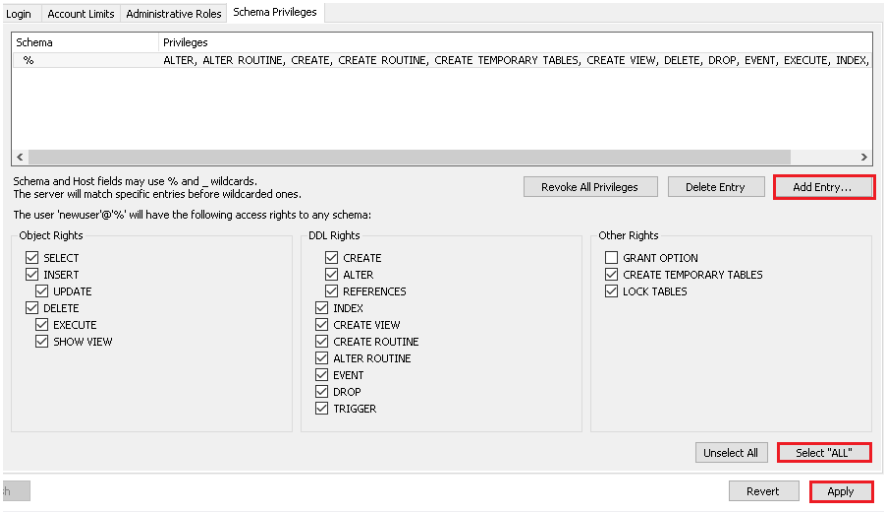


Figura. A.7 Pestaña Schema Privileges

7. Ir a la pestaña **Administrative Roles**, dar un clic sobre DBA, con lo cual se pondrá un visto en todos los casilleros. Finalmente dar clic en el botón **Apply** para guardar los cambios.

A.3.4 Solución a errores de MySQL

No se puede conectar al servidor de la base de datos.

Si al abrir la conexión en MySQL sale el error mostrado en la figura A.8 realizar lo siguiente.

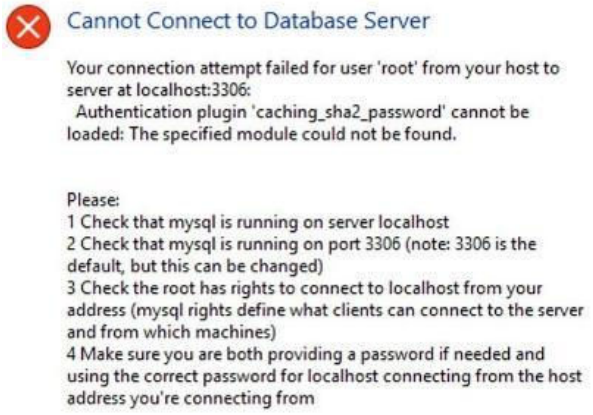


Figura. A.8 Ventana de error de conexión

1. Ir a los servicios de Windows. ⊞ Win + R (Se abrirá el cuadro de dialogo) escribir **SERVICES.MSC** para abrir los servicios de Windows.

2. Desactivar el servicio de **MariaDB**

No se puede ingresar como usuario root

Si al abrir la conexión no pide la contraseña puesta al crear la nueva conexión o se olvidó la contraseña. Realizar los siguientes pasos para cambiar la contraseña del usuario root.

1. Abrir XAMPP y detener la conexión de MySQL, luego dar clic en el botón **Config** y seleccionar **my.ini**, Figura A.1. Se abrirá un block de notas, Figura A.2.
2. En el block de notas debajo de **[mysqld]** agregar la siguiente línea y guardar.
   skip-grant-tables
3. Sin cerrar el bloc de notas regresar al panel de control de XAMPP e inicializar la conexión de MySQL, clic en el botón **Start**. Luego dar clic en el botón **Shell**.
4. Escribir **mysql -u root** en la consola. Esto establece una conexión sin contraseña.

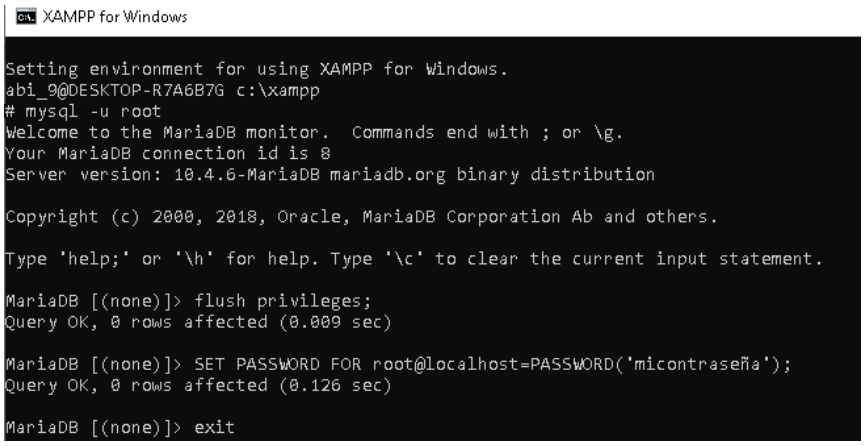


Figura. A.9 Comandos en consola para cambio de contraseña

5. Escribir el comando **flush privileges;**
6. Escribir el comando siguiente con respectiva contraseña y dar Enter.

   SET PASSWORD FOR root@localhost=PASSWORD('micontraseña');
7. Para salir de la consulta escribir en la consola **exit** y dar Enter.
8. Detener la conexión de MySQL, dar clic en el botón **STOP**.
9. Ir al bloc de notas y borrar la línea agregada en el punto 2. Guardar y cerrar.

Nota: Los pasos del numeral 5 y 6 son para las versiones desde MariaDB 10.4, para versiones anteriores se usa UPDATE mysql.user SET Password=PASSWORD(‘nueva_contraseña’) WHERE User=‘root’;

A.3.5 Configuración de conector ODBC

Para la configuración del conector en la máquina virtual de Intouch se deben realizar los siguientes pasos.

1. Instalación de MySQL Connector/ODBC 8.0
2. Ingresar a ODBC (32 bits) si no se encuentra en el administrador de datos se lo puede encontrar en la siguiente dirección. *c:\windows\sysWOW64\odbcad32.exe*
3. En la ventana de ODBC mostrado en la figura A.10 parte izquierda seleccionar MySQL y dar clic en **Add.**

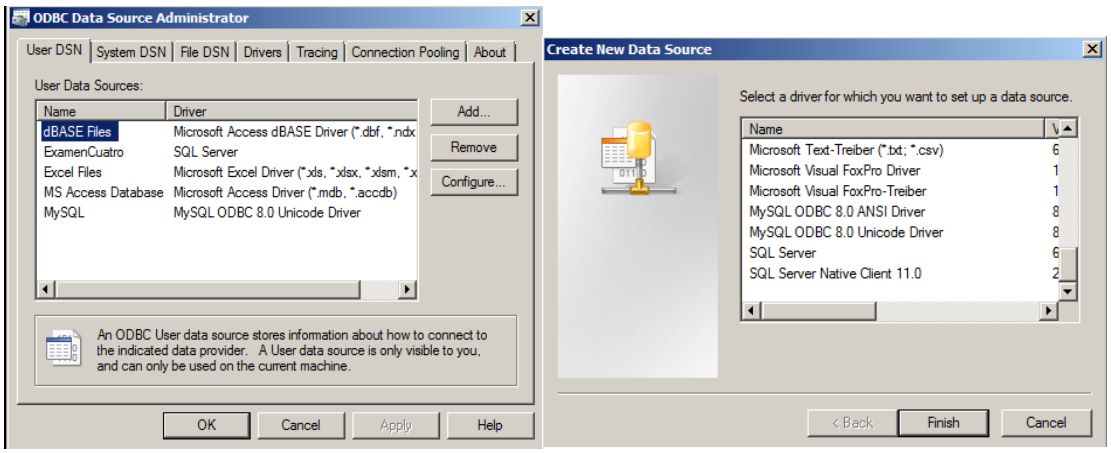


Figura. A.10 Ventanas en el Administrador ODBC

4. En la Figura A.10 parte derecha, seleccionar MySQL ODBC 8.0 Unicode Driver y presionar el botón **Finish**. Con lo cual se abre la ventana de la Figura A.11.

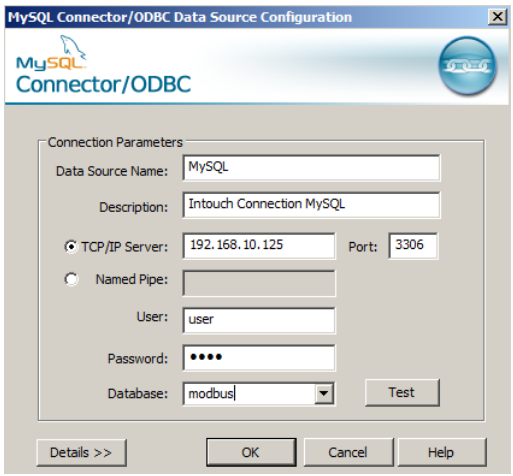


Figura. A.11 Ventana de configuración del conector

5. Rellenar los parámetros del conector, la dirección IP es la de la maquina donde está la base de Datos de MySQL y el puerto de conexión a la misma.

A.4. VENTANAS DEL SERVIDOR DE DATOS DESARROLLADO

La interfaz desarrollada tiene tres ventanas principales: Inicio, Modo Maestro, Modo Automático.

A.4.1 Ventana de inicio

Figura. A.12 Ventana de Inicio

En la ventana de inicio mostrada en la Figura A.12 se muestra una presentación de la interfaz. Además de dos botones que permiten elegir con cuál de los dos modos se va a trabajar.

A.4.2 Ventana modo Maestro

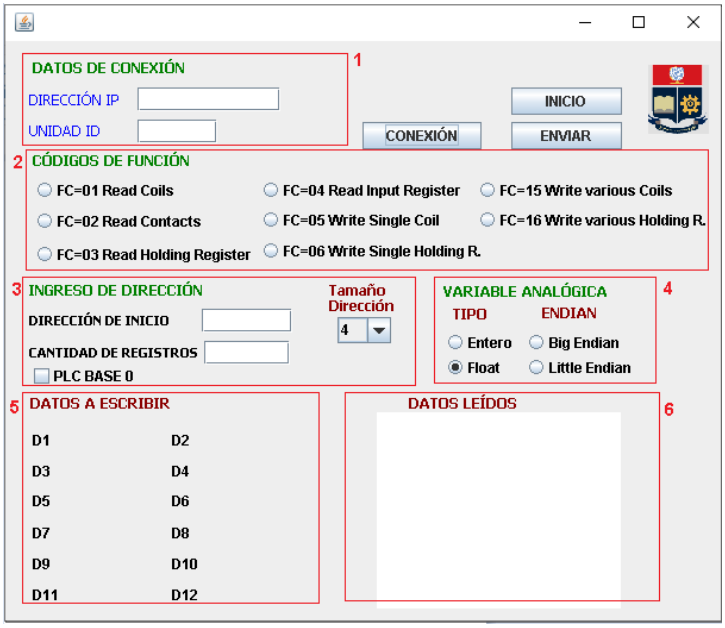


Figura. A.13 Ventana de primer modo

1. **Sección Datos de conexión:** Se realizará el ingreso de la dirección IP del dispositivo a conectar y el identificador de unidad (UNIDAD ID). La dirección IP a introducción debe mantener el formato de 4 octetos separados por un punto XXX.XXX.XXX.XXX por ejemplo 192.168.10.43. El identificador de unidad puede tener valores de 1 a 255.
2. **Sección Códigos de función:** Se puede elegir el código de función con el cual se trabajara. Los códigos de función 01, 02, 03 y 04 son de lectura mientras los códigos de función 05, 06, 15 y 16 son de escritura.
3. **Sección Ingreso de Dirección Modbus:** En esta sección se ingresará la dirección de inicio del registro Modbus, además de especificar el tamaño de la dirección, mismo que puede ser de un tamaño de 4, 5 o 6 dependiendo el dispositivo a conectarse. Adicionalmente se escoge si el PLC es de base 0 o no. En la cantidad de registro se ingresa cuantos valores se van a leer o escribir, este parámetro es necesario para todos los códigos de función a excepción de los códigos de función 5 y 6, debido a que realizan la escritura de un valor.
4. **Variable analógica:** Para los códigos de función 03, 04 y 16 que trabajan con una variable tipo Holding Register o Input Register al poder ser de dos tipos se requiere se especifique si se trata de una variable de tipo entero o flotante. Adicionalmente en caso de ser una variable tipo flotante se debe especificar el tipo de Endian del dispositivo.
5. **Sección Datos a escribir:** Una vez ingresado la cantidad de registros a escribir se habilita el ingreso de los datos para los códigos de función 15 y 16 de acuerdo a la cantidad especificada anteriormente, mientras que para los códigos de función 05 y 06 se habilita solo un cuadro para el ingreso del dato.
6. **Sección Datos Leídos:** Muestra los datos leídos con los códigos de función 01, 02, 03 y 04

Finalmente se tiene tres botones, mismos que son detallados a continuación:

- **INICIO:** Regresa a la ventana anterior para seleccionar el tipo de modo
- **CONEXIÓN:** Permite comprobar la conexión con el dispositivo cuya dirección IP fue ingresada en la sección Dirección IP.
- **ENVIAR:** Ejecuta el código de función seleccionado.

A.4.3 Ventana modo Automático

Esta ventana posee tres pestañas, mismas que son Configuración, Base de datos y Datos.

A.4.3.1 Pestaña Configuración

Figura. A.14 Pestaña de Configuración

En la pestaña mostrada en la Figura A.14 es en donde se realiza la configuración respectiva de los dispositivos a conectarse, los parámetros a ingresar son los detallados a continuación.

- Dirección IP.
- Identificador de unidad.
- Tiempo de muestreo, puede ser diferente para cada PLC.
- Tabla de datos, ingresar nombre de la tabla en la cual se almacenaran los datos de las variables a comunicación.
- Tipo de variable analógica, Se puede elegir entre Entero y Flotante, en caso de ser flotante se debe especificar el Endian.
- Tamaño de registro, pueden ser de tres tamaños 4,5 o 6 dígitos.
- Base del PLC, si esta seleccionado se trabaja con base 0 si no con base 1.

La principal función de esta pantalla es: permitir el ingreso de los datos del dispositivo a conectar, crear una tabla para guardar los datos en caso de no existir y realizar la conexión de manera continua. Adicionalmente cuenta con los siguientes botones:

- **Botón B1:** Comienza la comunicación continua con el/los PLCs agregados al servidor. Se requiere por lo menos un PLC agregado.

- **Botón B2:** Agrega la configuración realizada del PLC al servidor. La comunicación se realizará con los PLCs que hayan sido agregados al servidor.
- **Botón B3:** Permite crear dos nuevas tablas en la base de datos, una tipo histórico y otra que almacena los datos en tiempo real del PLC configurado, con el nombre ingresado.
- **Botón B4:** Permite guardar la última configuración de los PLCs que han sido agregados al servidor en una base de datos creada en MySQL.
- **Botón B5:** Carga en el servidor la última configuración guardada.
- **Botón B6:** Regresa a la ventana de Inicio

A.4.3.2 Pestaña Base de Datos

La segunda pestaña mostrada en la Figura A.15 permite el ingreso de las direcciones de los registros con las cuales trabaja en el PLC. En esta parte se ingresa el nombre del registro y la dirección. La dirección ingresada debe tener el tamaño que se seleccionó en la pestaña anterior, caso contrario al realizar la conexión mostrara que hay un error en la dirección.

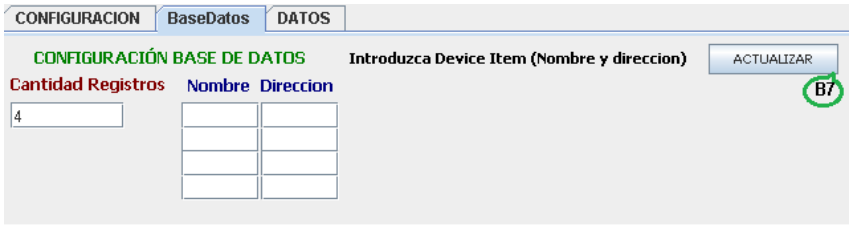


Figura. A.15 Pestaña Base de Datos

- **Botón B7:** Actualiza los datos en la base de datos correspondiente.

A.4.3.3 Pestaña Datos

En la Figura A.16 se muestra la pestaña DATOS, misma que permite la visualización de los datos en tiempo real de la tabla solicitada, estos valores se actualizarán cada 1 ms.

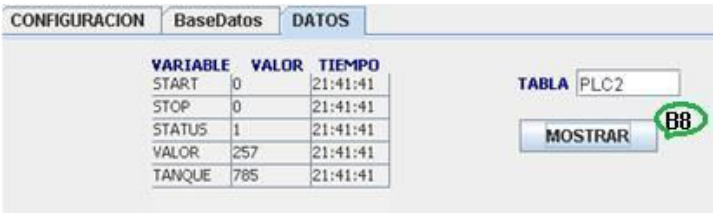


Figura. A.16 Pestaña DATOS

En la parte de TABLA se debe colocar el nombre de la tabla que se desea observar y al momento de presionar

- **Botón B8:** Al presionar empezara a mostrarse la tabla con las columnas de: nombre de la variable, valor y el tiempo de su última actualización

A.5. Comunicación de InTouch con base de datos

Para la conexión es preferible en la ventana del InTouch crear dos botones, conectar y desconectar. Para la conexión el botón realizara la siguiente acción cuando se presione

```
SQLConnect( ConnectionID,"Provider=MSDASQL;DSN=MySQL");
```

Para el botón de desconexión se realizará la siguiente acción

```
SQLDisconnect( ConnectionID);
```

Nota: Si no se crean botones de conexión y desconexión en el Script se debe de colocar al inicio la sentencia para la conexión y al finalizar la sentencia para desconexión.

1. Para la escritura crear un Blind List llamado ESCRIBIR al cual se le asociara el Tagname valor con la columna valor. Como se muestra en la figura

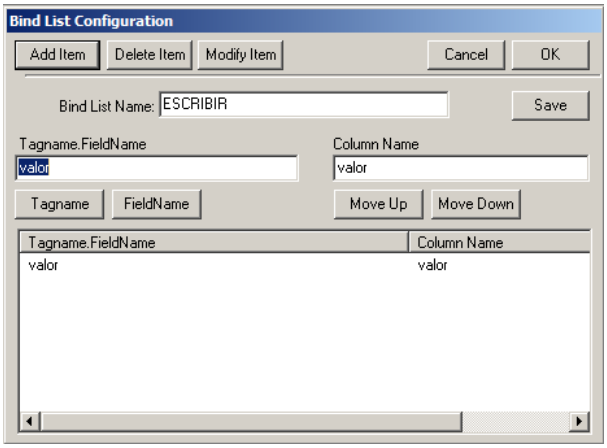


Figura. A.17 Ventana de Blind List en InTouch asociación variables de escritura

2. Crear un Blind List para cada registro que se va a leer y se asociará el Tagname de la variable con la columna valor. En la Figura A.17 en Tagname colocar el Tagname de la variable.

3. Realizar un Script de aplicación en donde se colocará las sentencias que permiten la escritura y lectura de las variables con la base de datos.

4. Crear adicionalmente un Tagname de dirección y uno de valor además del de conexión.

A continuación, en las Figuras A.18 y A.19 se muestra como se debe realizar las sentencias de escritura y lectura en el script de aplicación del HMI en InTouch.

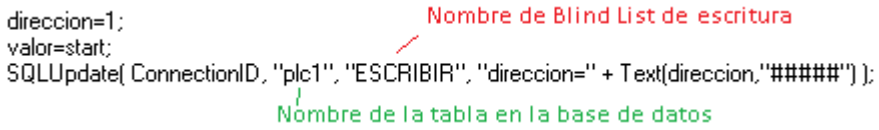


Figura. A.18 Sentencias SQL para escritura de variables con base de datos

En dirección se coloca la dirección a la cual se quiere escribir y en valor el nombre del tag que se va a escribir.

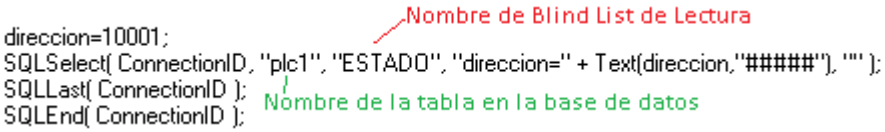


Figura. A.19 Sentencias SQL para lectura de variables de la base de datos

NOTA: Las sentencias de lectura y escritura se deben añadir al Script por cada registro que se vaya a leer o escribir. Si se quiere disminuir la utilización del Tagname dirección se puede escribir directamente en la sentencia SQLSelect el valor de la dirección con lo cual la sentencia queda del siguiente modo.

SQLUpdate(ConnectionID, "plc1", "ESCRIBIR", "direccion=1");

Como el ejemplo que se muestra a continuación en la Figura A. 20.

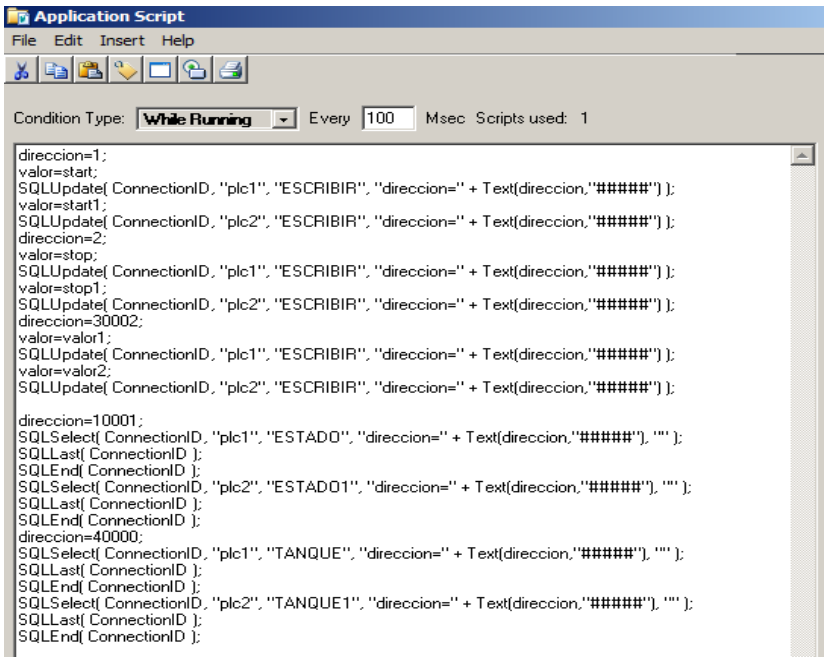


Figura. A.20 Ejemplo de Script de aplicación para la lectura y escritura de dos tablas donde se tiene START, STOP, STATUS, y la lectura y escritura de una analógica.