ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE PROTOTIPO DE DRIVER DE COMUNICACIÓN TIPO PROFINET I/O PARA LECTURA Y ESCRITURA DE DATOS BOOLEANOS, ENTEROS Y FLOTANTES USANDO SOFTWARE LIBRE

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

GABRIEL JOSIAS SANTOS RIVERA

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

CODIRECTOR: MBA. ANA VERÓNICA RODAS BENALCÁZAR

Quito, Junio 2022

AVAL

Certificamos que el presente trabajo fue desarrollado por Gabriel Josías Santos Rivera, bajo nuestra supervisión.

NOMBRE DIRECTOR

Dra. – Ing. Silvana del Pilar Gamboa Benítez

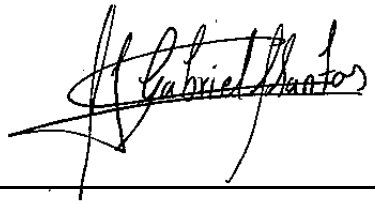
NOMBRE CODIRECTOR

MBA. Ana Verónica Rodas Benalcázar

DECLARACIÓN DE AUTORÍA

Yo Gabriel Josías Santos Rivera, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado 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.



GABRIEL JOSÍAS SANTOS RIVERA

ÍNDICE DE CONTENIDO

RESUMEN

El mercado actual de softwares industriales, que son utilizados para sistemas de control supervisorio, presenta un abanico de opciones que no son asequibles económicamente para la pequeña y mediana empresa. Esto limita la capacidad de trabajo de este tipo de compañías en los sistemas de automatización, orientados a control por software industrial.

El presente trabajo desarrolla un servidor de datos industrial que utiliza el protocolo Profinet TCP, un software libre tipo Java para la aplicación y un servidor tipo MySQL para la gestión de la base de datos. El prototipo final se constituye en una opción viable, de bajo costo, para uso dentro de la pequeña y mediana empresa.

Se implementó una interfaz para el servidor de datos industrial con 3 modos de funcionamiento, subdivididos en escritura y lectura. El primero propone un sistema de pruebas simple para lectura y escritura de datos, permitiendo la validación de la conexión con el PLC; el segundo habilita la configuración de parámetros para uso de la base de datos con el PLC y una comunicación de prueba para cerciorarse de que el funcionamiento es correcto; el último modo permite el acceso a una pantalla de usuario final con los datos de la configuración previa.

Se realizaron una serie de pruebas que validaron el funcionamiento y operatividad del servidor.

PALABRAS CLAVE: Comunicación, PLC, MySQL, Java, Profinet, InTouch

ABSTRACT

The current market for industrial software, which is used for supervisory control systems, presents a range of options that are not economically affordable for small and medium-sized companies. This limits the work capability of these in automation systems, oriented to control by industrial software.

The present work develops an industrial data server that uses the Profinet TCP protocol, a Java-type free software for the application, and a MySQL-type server for the management of the database. The final prototype is a viable, low-cost option for use within small and medium-sized companies.

An interface for the industrial data server was implemented with 3 operating modes, subdivided into writing, and reading. The first proposes a simple test system for reading and writing data, allowing the validation of the connection with the PLC; the second enables the configuration of parameters for use in the database with the PLC, and a test communication to make sure that the running of the program is right; the last mode allows access to a user screen with the data of the previous configuration done.

A series of tests were carried out and validated the server's performance and operability.

KEYWORDS: Communication, PLC, MySQL, Java, Profinet, InTouch

1. INTRODUCCIÓN

Actualmente, están en desarrollo las tecnologías orientadas hacia sistemas de automatización de nivel industrial. Este tipo de tecnologías de información requieren de un proceso de estandarización, que está determinado por las características de los protocolos de nivel industrial utilizados para su aplicación en campo. Dentro de la rama de redes industriales, se considera como una de sus principales aplicaciones el control supervisorio aplicado en sistemas de tipo SCADA. La comunicación entre equipos es esencial e indispensable para llegar al objetivo de la estandarización en el proceso de control supervisorio en un sistema de redes industriales. Se debe tener en cuenta que en la actualidad este tipo de inversión solamente es accesible para empresas con un alto ingreso, lo cual excluye a empresas pequeñas y medianas; la inversión en software requerida para implementar sistemas de automatización complejos orientados a procesos industriales es considerablemente alto.[1][2]

Por esta razón actualmente la implementación de sistemas de software libre que puedan solucionar este tipo de inconvenientes, capaces de proporcionar de forma barata y confiable sistemas que sean adaptables a la comunicación entre controladores y sistemas de supervisión para la implementación de redes industriales, se encuentran en desarrollo y en auge [3]. Debido a que los sistemas de software libre no tienen costo alguno, la única inversión requerida para su implementación, es el desarrollo de las aplicaciones para su uso específico lo cual también es una ventaja, ya que la implementación de este tipo de sistemas está basada en función de los requerimientos del proceso.[4]

El protocolo Profinet se utiliza para la implementación e instalación de sistemas de tipo control supervisorio de la marca Siemens y sus marcas asociadas. Esto debido a que la fundación Profinet es financiada en su mayoría por Siemens, y este protocolo; aparte del protocolo Profibus, conforman el repertorio principal de comunicación existente para PLCs de tipo Siemens. La Fundación Profibus-Profinet determina las características del estándar en Profinet, incluyendo el desarrollo de objetivos y actualización de sistemas de seguridad orientados hacia la industria, basados en los avances tecnológicos actuales. Profinet forma parte del estándar de Ethernet industrial para sistemas de automatización y tecnología.[3]

El presente trabajo de titulación desarrolla la aplicación de un servidor de datos de tipo industrial para el protocolo Profibus, utilizando el software de Java. El objetivo de este driver es su aplicación como suplemento de software industrial orientado al uso en software libre. Adicionalmente, se implementa una base de datos para almacenamiento, lectura y

escritura, siendo un puente de comunicación entre la interfaz hombre máquina y el controlador lógico programable.

1.1 OBJETIVOS

El objetivo general de este Proyecto Integrador es:

- Implementar un prototipo de driver de comunicación tipo industrial Profinet para lectura y escritura de datos booleanos, enteros y flotantes, usando software libre.

Los objetivos específicos de este Proyecto Integrador son:

- Estudiar la normativa y la base teórica relacionada con la implementación del protocolo Profinet para comunicación entre dispositivos industriales.
- Establecer los requerimientos funcionales que debe cumplir el driver de comunicación para la implementación de lectura y escritura de datos booleanos, enteros y flotantes.
- Buscar y seleccionar las herramientas de software libre para el desarrollo del driver propuesto.
- Implementar la versión demo del driver de comunicación propuesto
- Validar los drivers desarrollados con equipos industriales comerciales, así como con una interfaz implementada en PC en base a un software industrial comercial

1.2 ALCANCE

Se estudiará los métodos que se pueden utilizar para la conexión TCP entre dos dispositivos.

Se determinará de acuerdo con la normativa del protocolo Profinet, la estructura de la trama de datos y proceso de comunicación entre transmisor y receptor.

Se seleccionará el software para el desarrollo del driver de comunicación propuesto. Se implementará un programa dentro del software de programación establecido para la escritura y lectura de datos de tipo byte desde una computadora hacia un controlador lógico programable usando el protocolo Profinet.

Se implementará un programa para la escritura y lectura de datos de tipo entero y flotante desde una computadora hacia un controlador lógico programable usando el protocolo Profinet.

Se desarrollará el servidor de datos en el software libre seleccionado.

Se realizará una interfaz de usuario para la configuración y diagnóstico del servidor de datos, en donde se ingresen los parámetros de configuración de este.

Se realizará las pruebas del funcionamiento del servidor de datos con PLCs Siemens con puerto Ethernet.

Se desarrollará la programación necesaria para la creación de una base de datos relacional en la que se guardarán los resultados de la adquisición, misma que se comunicará con el servidor implementado.

Se realizará una interface de operador en software industrial comercial, para la validación de la interoperabilidad entre el servidor propuesto y aplicaciones de Windows.

Se realizará las pruebas del funcionamiento de las interfaces gráficas realizadas y la comunicación del servidor de datos implementado con PLCs Siemens. Por una parte, se probará el desempeño global del sistema para validar la interoperabilidad del driver desarrollado con equipos y software industrial comercial. También se evaluará, el funcionamiento individual de las funciones del driver tales como la interacción del driver con los PLCs, la interface de monitoreo y configuración del driver, además de lectura y escritura hacia la base de datos.

Se validará el funcionamiento del programa mediante la revisión del estado de las variables del controlador lógico programable (PLC) con la herramienta de programación de éste y la comparación del estado de las variables del controlador lógico programable.

Se realizará pruebas comparativas para verificar el correcto funcionamiento del software desarrollado en comparación con un servidor de datos industrial comercial, así como el cumplimiento de características de conexión y comunicación

1.3 MARCO TEÓRICO

Esta sección considera los conceptos teóricos acerca de los sistemas de recepción y análisis de datos, en las redes industriales, y el protocolo Profinet; además de información relevante para la comprensión del presente trabajo.

1.3.1 SOFTWARE COMPUTACIONAL PARA IMPLEMENTACIÓN DE SISTEMAS DE ADQUISICIÓN Y RECEPCIÓN DE DATOS, EXISTENTES EN EL MERCADO.

La validación del driver desarrollado en base a un software industrial comercial forma parte de lo establecido en los objetivos del proyecto. El cumplimiento de este requiere el conocimiento de las características generales de programas ya existentes comercialmente, que puedan establecer una comunicación con elementos industriales. Se realizará el análisis de 5 programas comerciales, estos están presentados en la tabla 1.1; siendo los más manejados en la industria para comunicación con elementos de la familia Siemens; que utilizan mayormente el protocolo Profinet. Estos drivers trabajan con el protocolo OPC, mientras que el prototipo desarrollado utiliza el protocolo Profinet.

El análisis desarrollado permite establecer el estado del mercado de los drivers comerciales, y posteriormente definir las ventajas del uso del prototipo; principalmente en el campo de la accesibilidad económica.

Tabla 1.1. Drivers de comunicación Profinet comerciales

Desarrollador	Software	Detalle	Licencia
Wonderware	DASSIDirect DAServer	El servidor DAServer es una aplicación de Microsoft Windows. Provee aplicaciones que sirven para acceder a información de la familia de PLC S7-200/300/400[5]	Existe una versión de demostración que tiene 120 minutos, para posteriormente solicitar la compra de una licencia, y dependiendo de la cantidad de tags, el costo puede variar. El precio del servidor de InTouch de 500 tag en la versión 2020 es aproximadamente de 3329.43$[6]
Matrikon	MatrikonOPC Server for Siemens	Es un servidor de OPC para uso de PLC Siemens con la familia de S7-200,300,400,1200,1500[2]	Existe una versión gratuita demo, pero para uso continuo la licencia por un año es de

			USD1,200 y se añade un costo adicional por el servidor OPC de 2,300$.
National Instruments	NI OPC Server for Siemens	Servidor de NI para OPC, orientado a Schneider pero tiene adaptabilidad a Siemens.[2]	Se entrega el servidor OPC para Siemens como un complemento para LabView versión completa con un costo de 3925$ por año.
Kepware	Kepserverex OPC Siemens Suite	Basado en Siemens específicamente, incluye aplicaciones de OPC para cliente.	Existe una versión gratuita de uso limitado y una licencia para uso completo con costo de 475$.

Este tipo de sistemas solamente representan a los servidores de datos, se requiere a su vez del programa para desarrollo de HMI. En algunos casos estos se incluyen en los sistemas, como es el caso de National instruments, pero existen otros en los que se requiere un software externo como es el caso de Intouch, WinCC para Siemens o IGSS o Modicon para desarrollar una interfaz de hombre máquina.

Hoy en día, existen trabajos realizados en la Escuela Politécnica Nacional en software libre para la implementación de un SCADA, utilizando los protocolos de comunicación Modbus. El análisis de trabajos previos realizados, orientados al desarrollo de un prototipo de comunicación en un protocolo industrial, establece un precedente para la relevancia que el proyecto puede tener dentro de la universidad y la comunidad científica. Estos proyectos establecen los pilares para el desarrollo del prototipo y a futuros programas dentro de esta rama de la investigación. Establecen la viabilidad de la implementación de drivers de comunicación industrial basados en software libre.

1.3.2 PROTOCOLO PROFINET

Profinet nace del protocolo Profibus DP, los mismos forman parte de la organización internacional *Profinet Foundation*. Profinet se encuentra estandarizado en las normativas IEC61158 e IEC61784. Este está orientado a tener más capacidades y desarrollarse más

que Profibus DP, esto debido a sus características, así como la capacidad de Profinet de ser 100% Ethernet, lo cual permite una interoperabilidad amplia. Esto propone un precedente para desarrollo de tecnología basado en protocolos Ethernet. Citando a Peter Wenzel "*PROFIBUS DP ist gut, PROFINET kann einfach mehr*"[3], que se traduce como "*PROFIBUS DP es bueno, Profinet puede ser fácilmente mejor* ".

Las expectativas para Profinet en la industria 4.0 incluyen comunicación proxy y con interruptores APL (capa física avanzada), Profinet siendo utilizado como un medio de comunicación entre los controladores y los elementos en campo. La figura 1.1 presenta una ejemplificación de la expectativa de aplicación de Profinet a largo plazo según la fundación Profinet.

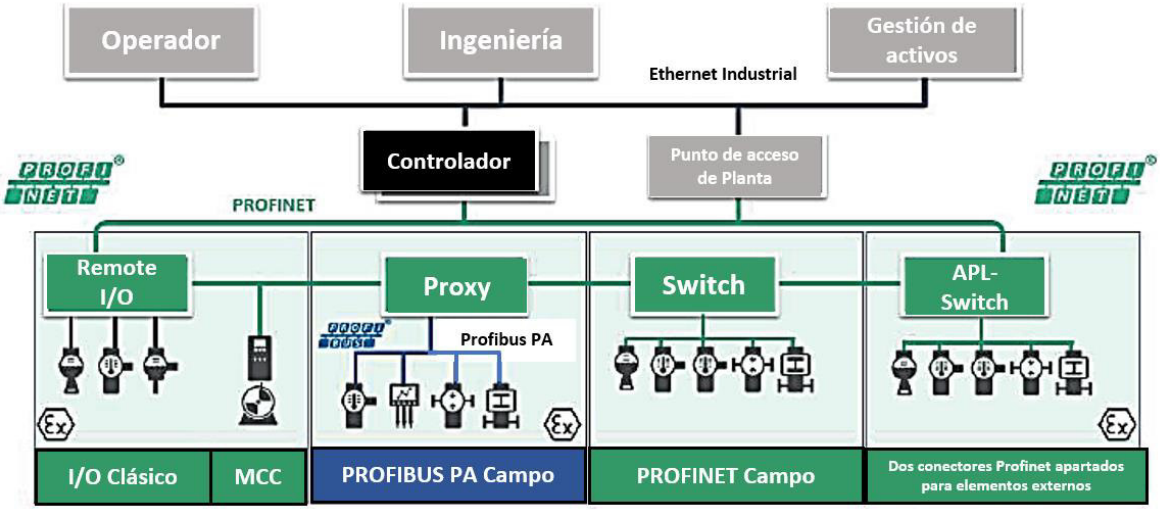


Figura 1.1. Profinet y Profibus PA, en el campo derecho futuro con APL [7]

1.3.2.1 Estado del Arte Trama Profinet

El análisis de la estructura de la trama Profinet es requerido para el desarrollo del driver; estableciendo los encabezados a utilizarse, la función de cada uno de estos y su respectiva configuración, por lo que la inspección de la trama parte por parte permite entender los valores establecidos en estos registros, ya que posteriormente deben ser replicados en la programación para establecer de forma exitosa la comunicación.

En el análisis de la trama se considera el programa Wireshark. Este se encuentra orientado al análisis de protocolos y paquetes enviados en una red, se puede observar cada una de las partes de la trama enviada, el proceso de inicialización de la trama o en su defecto de la comunicación realizada entre 2 elementos. El análisis realizado por el proceso de una trama únicamente está delimitado para sistemas en los que la computadora no inicia el

proceso de comunicación, sino que acepte el mismo tanto para la recepción y el envío de datos.

En la figura 1.2. se puede observar el entramado Profinet RT, como se mencionó anteriormente, este entramado se basa en el envío sincronizado de datos en tiempo real. En las siguientes secciones se muestra el funcionamiento de cada una de las partes del entramado así como sus configuraciones respectivas para el funcionamiento de Profinet.

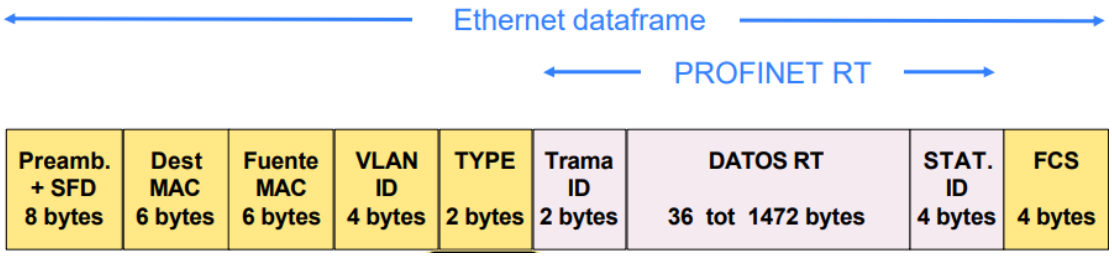


Figura 1.2. Trama Profinet para su clasificación RT [8]

Para la compresión del entramado Profinet se debe tener en cuenta el protocolo Ethernet, el protocolo ipv4, aparte del protocolo TCP. Esto tiene que analizarse a partir de las direcciones MAC, así como de las direcciones IP, los puertos utilizados en cada uno de los elementos y las banderas encendidas en función de las características del mensaje.

1.3.2.2 Trama en el Modelo OSI

Cada uno de los encabezados de los protocolos anteriormente mencionados presenta un nivel específico del modelo OSI, cuya vinculación con el protocolo se ejemplifica en la figura 1.3.

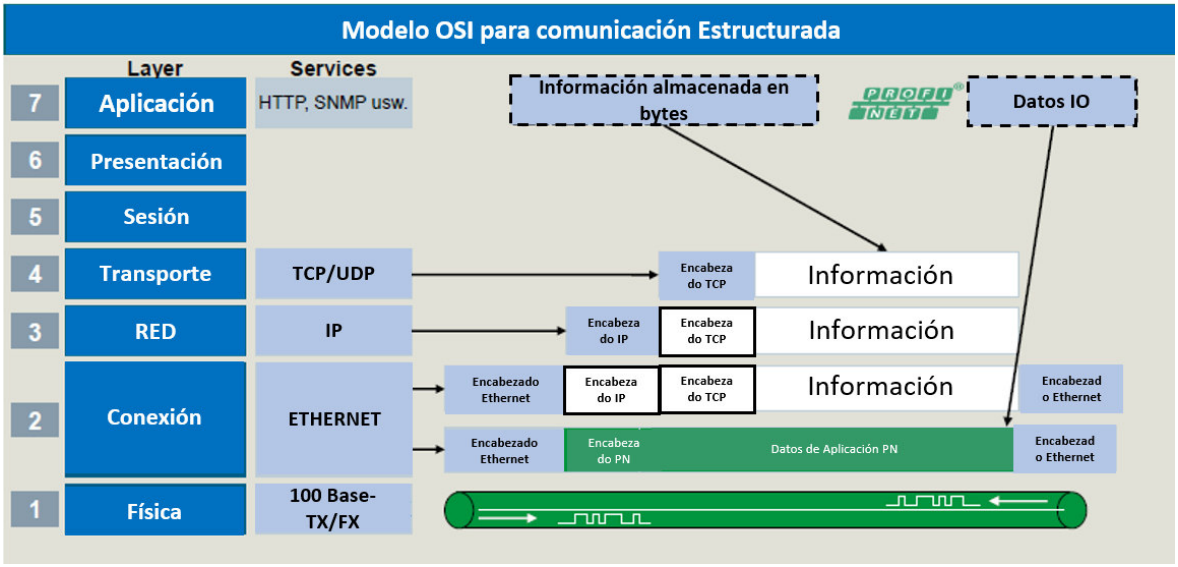


Figura 1.3. Trama Profinet segmentada y aplicada al Modelo OSI[9].

Los encabezados se muestran de la siguiente forma:

Capa física: No tiene un elemento referente a la trama.

Capa de Enlace de Datos: Encabezado Ethernet, en este caso Ethernet 2.

Capa de Red: Comunicación basada en la dirección IP. En este caso encabezado IPv4.

Capa de Transporte: Comunicación TCP. Establecimiento de los puertos de comunicación en el encabezado TCP.

1.3.2.3 Encabezado Ethernet 2

El encabezado Ethernet inicia con la dirección MAC del destinatario del mensaje, esto significa 6 pares de bits que presentan las características únicas del dispositivo a enviar el mensaje, en los que se ocupan los bits LG e IG, que sirven para determinar características individuales de la dirección MAC. Pasado de informar la dirección MAC del destinatario, se ingresa la dirección del origen del mensaje, que de la misma forma que en el anterior direccionamiento, ocupa 6 pares de bits. Posteriormente se determina el tipo de protocolo IP que para este caso es Ipv4 y el *padding* (relleno de datos para completar un encabezado de 32 bits). En la figura 1.4. se presenta un ejemplo simple del encabezado Ethernet.

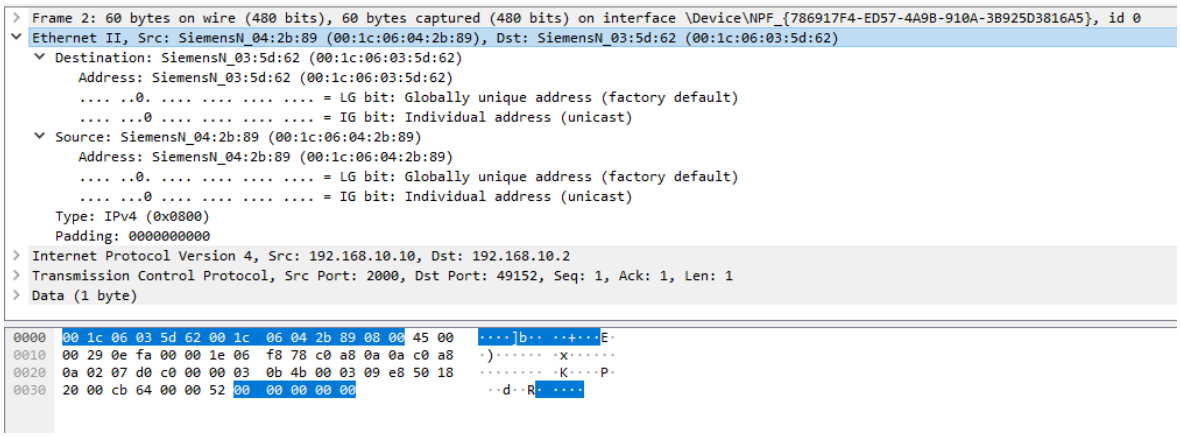


Figura 1.4. Encabezado Ethernet obtenido de Wireshark. Comunicación entre PLCs en protocolo Profinet. Se resalta el encabezado Ethernet.

Se puede observar en la figura 1.4. el estado de los bits GL e IG de cada una de las direcciones MAC, se encuentran en 0 por default; en la comunicación tipo Profinet se espera que su estado se mantenga de esta forma. La información referente a esta encabezado es en su mayoría ocupada por los dispositivos de ruteo, ya que la configuración de este empieza desde el protocolo IPv4.

1.3.2.4 IPv4

El encabezado de internet, en este caso es IPv4, utiliza múltiples identificadores, que permiten el reconocimiento de las direcciones IP y sus características. Primero se menciona el tipo de versión que se ocupa, existiendo actualmente dos versiones: IPv4 y IPv6. La versión IPv4 que es la que se utilizará para el desarrollo del proyecto, contempla una dirección de 4 bytes. Se debe mencionar que, posterior de la versión del protocolo IP, se encuentra la longitud del encabezado. Existe un campo de servicios diferenciados que se encuentra en default ya que no es requerido para este proceso. Se indica la longitud total y el número de identificación. Se menciona: el número de banderas, el offset del fragmento, tiempo de vida, tipo de protocolo utilizado; en este caso este es el protocolo TCP. Finalmente se tiene el chequeo de errores y las direcciones IP de origen y de destino, como se observa en la figura 1.5.

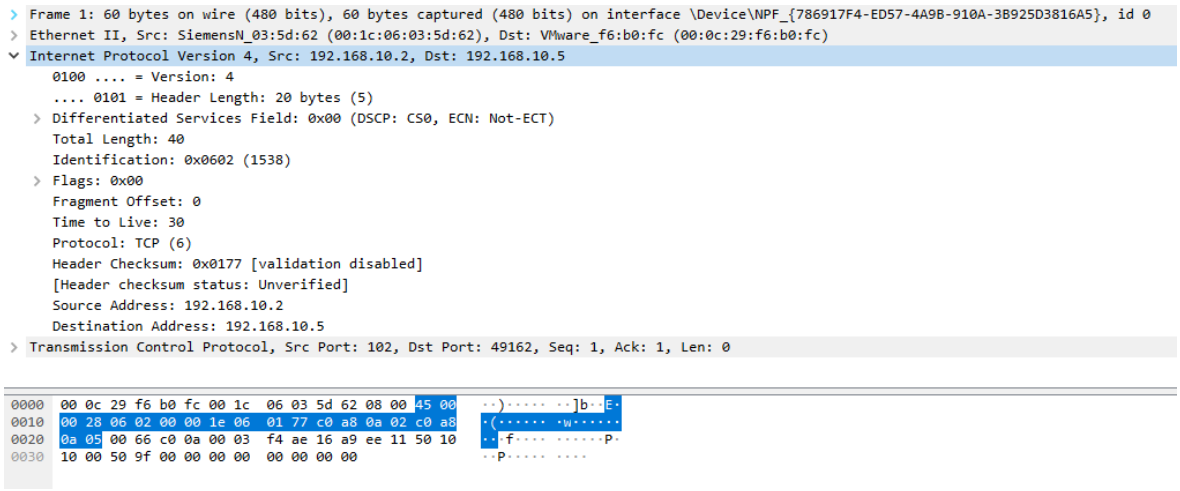


Figura 1.5. Encabezado IPv4 obtenido de Wireshark. Comunicación entre PLCs en protocolo Profinet. Se resalta el encabezado IPv4.

1.3.2.5 TCP

TCP es conocido como el protocolo de transporte orientado a conexión a internet. Existe una diferencia considerable con respecto a un sistema TCP con respecto a un sistema UDP, para el caso de Profinet se utiliza un encabezado tipo TCP en su clasificación RT para Profinet. El encabezado TCP trabaja con el puerto de origen y el puerto de destino, a continuación, se muestra cada una de las partes del protocolo: puerto de origen, puerto de destino, índice de trama, longitud del segmento, número de secuencia, acknowledge number, longitud del encabezado, banderas, tamaño de la ventana, chequeo de errores, análisis de secuencia de reconocimiento, estampas de tiempo, TCP Payload. Estas partes

pueden ser observadas en la figura 1.6. que ejemplifica la organización de los datos en el encabezado.

Figura 1.6. El formato de un segmento TCP con un encabezado TCP seguido de información [10].

Como se observa en la figura 1.6. la organización de los bits del encabezado TCP se organiza en grupos de 16 bits. Cada uno de los bloques de información pueden ser revisados en la herramienta de Wireshark, así como sus estados en una aplicación con el protocolo Profinet. En la figura 1.7. se desglosa de forma comprensible cada una de las partes del encabezado así como los valores existentes en este caso, la imagen es únicamente referencial para el entendimiento del funcionamiento del encabezado TCP.

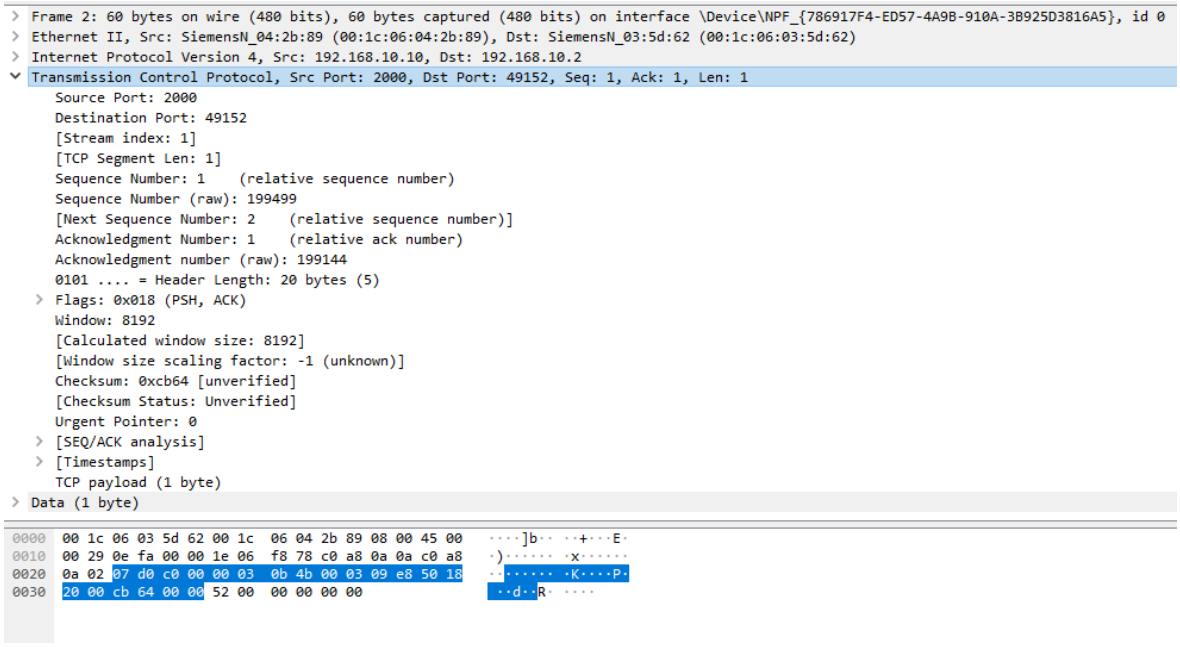


Figura 1.7. Trama enviada de PC a PLC en red. Imagen tomada de herramienta Wireshark. Se hace énfasis en el encabezado TCP.

Se observa en la figura 1.7. el estado de cada una de las partes del encabezado. Siendo lo más importante para el desarrollo del driver los puertos utilizados para la comunicación, el tamaño del segmento TCP así como las banderas encendidas en el protocolo.

1.3.3 PROFINET EN EL MODELO OSI

Profinet Foundation toma como referencia el modelo OSI para explicar el funcionamiento, así como los tipos de conexiones y posibilidades existentes en Profinet. La información mostrada para la ubicación de Profinet en el Modelo OSI fue obtenida de un manual ofrecido por Profinet Foundation presentando los requerimientos y características generales del protocolo, que son requeridos para la implementación de un elemento que pueda ocupar Profinet en una red industrial. [11]

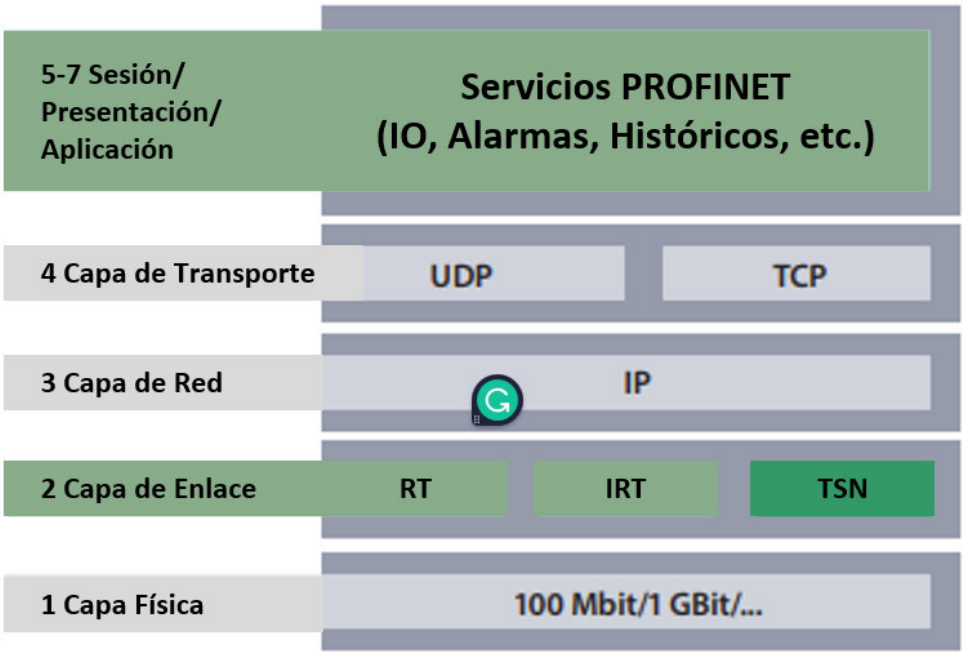


Figura 1.8. Profinet en el Modelo OSI[11].

En la figura 1.8. se toma en cuenta las 4 primeras capas para diferenciar el modelo OSI de otros protocolos y es importante resaltar las características de la segunda capa del modelo, que es la capa de en enlace de datos, en la cual se muestra diferentes tipos de comunicación Profinet; esto se debe a los diferentes tipos de comunicación desarrollados por Siemens en este protocolo.

1.3.3.1 CLASIFICACIONES DE PROFINET

Para entender el tipo de Profinet ocupado para el desarrollo del proyecto, se debe comprender las diferentes clasificaciones consideradas por Siemens existentes para este protocolo. Se consideran 3 clasificaciones: según el tipo de canal utilizado en red Profinet, aplicación del protocolo y según desarrollo del elemento de aplicación.

Tipo de canal utilizado en red Profinet:

Esta clasificación está basada en la segunda capa del modelo OSI, capa de en enlace de datos, en la cual según se observa en la figura 1.8. existen 3 tipos de Profinet existentes en esta capa. Estos tres tipos son IRT (Tiempo Real isócrono), NRT (Tiempo no Real) y RT (Tiempo Real). Cada uno de estos tipos se caracteriza por el tiempo requerido para el intercambio de información, así como la importancia de los datos existentes en estos canales.

Dentro de los datos que se pueden comunicar en Profinet, existen aquellos que requieren de un envío cíclico que puede ser o no sincronizado, así como aquellos que tienen una comunicación acíclica. El tipo IRT y RT son datos cíclicos donde IRT es un tipo de dato isócrono y RT es un dato síncrono. Siendo los datos RT los más comunes en aplicaciones industriales, mientras que los sistemas que requieren de IRT son aquellos que necesitan de aplicaciones de alta velocidad ya que los ciclos de envío de información son menores a 1ms y no todos los dispositivos son capaces de ocupar este tipo de datos[8].

El canal NRT se ocupa para requerimientos acíclicos. Este se requiere para sistemas que solamente necesiten envío de información una vez en caso de existir un cambio relevante en la información, como es el caso de alarmas o diagnósticos[9].

En la figura 1.9. se puede observar la clasificación de los canales en una red Profinet.

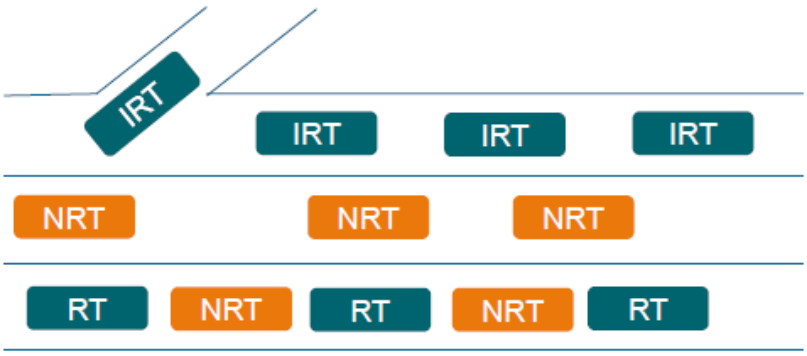


Figura 1.9. Distribución de los canales en una Red de comunicación Profinet [9].

Para el desarrollo del proyecto no se toma en cuenta de forma específica ninguno de los tipos de datos antes ejemplificados, sino que se toma una mezcla de los dos datos tipo cíclico antes mencionados. Esto debido a que la modificación de la base de datos presenta un inconveniente de comunicación que impide ocupar un sistema isócrono total, otra razón que presenta un inconveniente es las capacidades del PLC ocupado ya que la familia S7-1200 no está adaptado para trabajar con sistemas isócronos.

Clasificación según Aplicación de Protocolo:

Esta clasificación es la más conocida para este protocolo. Los dos tipos de comunicación existentes son Profinet IO y Profinet CBA. Siendo el primero ocupado para comunicación con los dispositivos de campo, como sería el caso de un variador de velocidad. El segundo tipo se encarga del sistema en su nivel superior, esto quiere decir que este tipo de comunicación está orientada al intercambio de datos entre controladores[8].

El driver desarrollado está basado en la comunicación Profinet, su proceso de prueba se realizó en la clasificación CBA; pero su aplicación está orientada a Profinet I/O y CBA. Su uso en un entorno tipo Profinet IO se propone siempre y cuando se conozca de forma clara el funcionamiento de los datos en el elemento de campo.

Clasificación según elemento de desarrollo:

Profinet tiene la ventaja de permitir a cualquier persona o empresa el desarrollo de su propio elemento de comunicación. Profinet Foundation determina que existen tres tipos de desarrollos posibles en función de las capacidades del elemento desarrollado: Conformance Class A, Conformance Class B y Conformance Class C [11].

Estas clases determinan los requerimientos necesarios en función de las características del campo en el cual el elemento Profinet va a ser ocupado. Estos requerimientos son los requeridos para la obtención de la certificación Profinet, para la comercialización de este tipo de productos.

Este proyecto no requiere el cumplimiento de estas especificaciones. Profinet no está orientado a la suplantación de sistemas OPC, por lo que el resultado obtenido solamente funciona en forma de prototipo. Sin embargo, si es posible clasificar las capacidades del driver en las clases existentes.

CC-A: Conformance Class A. Requiere de una herramienta de ingeniería para su aplicación y usa la infraestructura de una red de Ethernet existente. Incluye las funciones básicas de Profinet[11].

CC-B: Conformance Class B. Comprime las funciones de CC-A y tiene una interfaz amigable con el usuario sin la necesidad del uso de una herramienta de ingeniería. Soporta diagnósticos de funciones de red como es el caso de mensajes de estado del puerto.

CC-C: Conformance Class C. Comprime todas las funciones de CC-B. Soporta alta precisión y transmisión de información de forma determinística, incluida aplicaciones isócronas [11].

Según lo revisado anteriormente el prototipo desarrollado se encuentra en la clasificación de CC-A, ya que en este caso se necesita de una herramienta que requiere conocimientos previos externos al uso del driver para el funcionamiento del sistema, en este caso MySQL. Aunque, ya se encuentra utilizando una interfaz de usuario que facilita el uso del sistema, lo cual permitiría buscar a largo plazo características similares a la CC-B.

1.3.4 MANEJO DE DATOS EN PROFINET

El manejo de datos en el protocolo en los PLC Siemens forma parte fundamental del desarrollo del proyecto. Es necesario reconocer la programación y los elementos requeridos, a partir del cual los bloques y los controladores funcionan en este protocolo. Se debe entender el manejo de la memoria para los controladores de la familia Siemens así como los tipos de datos existentes para los PLCs S7-1200 que fueron utilizados para el desarrollo del proyecto.

1.3.4.1 Bloques de Programación

El funcionamiento de los bloques de programación de los controladores determina cuáles son las características, entradas y valores que deben analizar cada uno de los PLCs para ser capaces de determinar cuál debe ser el formato de la trama a aplicarse en la red para el envío de la información. La comunicación en el protocolo Profinet está orientada a utilizarse entre controladores tipo PLC, o en otros casos con actuadores o sensores especializados.

Para el desarrollo de la programación inicial se debe determinar los tipos de PLC a trabajar, en este caso siendo dos PLC de la familia S7-1200, se debe realizar una conexión entre ambos; como se muestra en la figura 1.10.

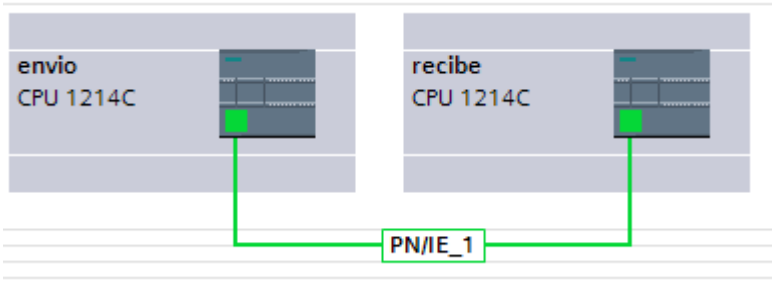


Figura 1.10 Conexión entre PLC en el TIA Portal.

Pasando esta estructuración básica mostrada en la figura 1.10, se procede a la configuración individual de cada uno de los controladores. Se establecen las direcciones en la red de cada uno, así como características internas. Durante el proceso de configuración se deben activar las marcas de tiempo de cada uno de los controladores, en la programación interna se va a requerir el uso de estas.

1.3.5 MANEJO DE LAS VARIABLES EXISTENTES EN EL PLC

Dentro de la planificación del proyecto está el análisis de variables tipo entero, booleano y punto flotante. Es importante aclarar que los PLCs S7-1200 no manejan solamente estos tres tipos de datos, sino que existe un abanico completo de tipos datos con los que pueden trabajar, estos pueden variar dependiendo la generación de PLC y características del firmware.[12]

1.3.5.1 Estándar IEEE754

El formato de punto flotante, que fue estandarizado por IEEE en 1985, tiene 2 representaciones: precisión simple y precisión doble. El formato de precisión simple consiste en 32 bits, y el de doble precisión tiene un formato de 64 bits. En este trabajo de investigación se trabajará únicamente con el formato de 32 bits. El formato de precisión de 32 bits incluye un bit de signo, 8 bits del valor de exponente, y 23 bits de un valor fraccionario. La representación del sistema de precisión de punto flotante simple usado se muestra en la figura 1.11.[13]

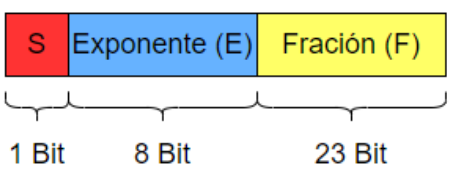


Figura 1.11 Representación del sistema de punto flotante en 32 bits.[13]

Se puede obtener el valor de punto flotante del sistema mediante la siguiente ecuación:

$$(-1)^{s} x\ 1.F\ x\ 2^{E} - 127 \qquad (1.1)[13]$$

El sistema de precisión doble expone un bit que representa el signo, 11 bits que representan el exponente y 52 bits denominado mantisa. El exponente tiene un desplazamiento de 1023 referente a sus 11 bits. La mantisa tiene un bit implícito y 52 bits fraccionarios.[6]

1.3.5.2 Método de Conversión Para Datos Enteros

Para el trabajo en el software se requiere que los datos que se reciben o envían sean compuestos o descompuestos en su1s partes binarias, como directamente en su composición de bytes. Esto se logra mediante un método que permita la descomposición de un valor entero en sus componentes.

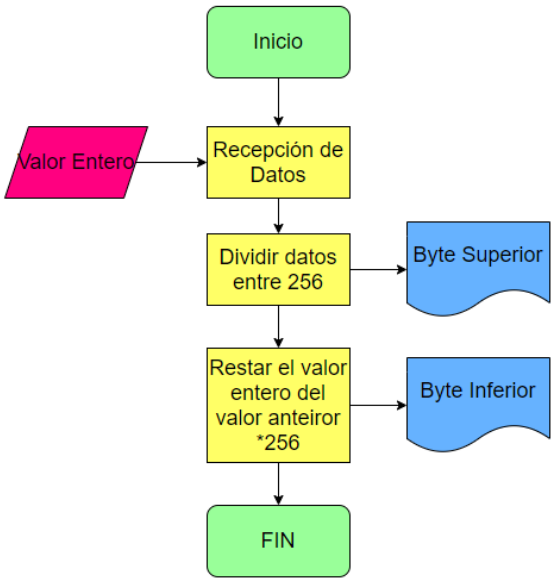


Figura 1.12 Diagrama de Flujo del método de descomposición.

En la figura 1.12 se expone el diagrama de flujo básico del método de conversión a bytes de un dato entero. Se inicia con la recepción del dato, y se divide este entre 256, al valor íntegro ingresado se le resta el valor anterior por 256 y se obtiene el byte inferior, mientras que el dato dividido entre 256 presenta el byte superior. [6]

1.3.6 JAVA

Partiendo de un análisis de los lenguajes de programación existentes que son capaces de manejar encabezados TCP, que se mostrará posteriormente, se establece a Java para el

desarrollo del prototipo. Este lenguaje contempla la capacidad de modificar los registros de los encabezados según se requiera dentro del protocolo, e incorpora estamentos de programación direccionados al manejo de bases de datos.

Java es presentado como un lenguaje de programación en computadoras de alto nivel, esto quiere decir que se encuentra bastante alejado a la programación directa del microprocesador de la computadora, lo cual se considera programación de bajo nivel. En Java se desarrollan estamentos de programación que son similares a una oración en un idioma, esos estamentos llevan las tareas del pseudocódigo y se conocen como comandos.[14]

En el momento de desarrollo de un programa en Java, este se convierte en lenguaje de bajo nivel, gracias al compilador de lenguaje, que se conoce como lenguaje de máquina. El lenguaje de máquina determina las acciones individuales del procesador y se puede analizar este como la combinación de unos y ceros en grupos específicos que ordenan al microprocesador las acciones a realizar en el mismo. El compilador de lenguaje en este caso debe estar orientado a trabajar con JAVA.

1.3.6.1 Programación Orientada a Objetos

La programación orientada a objetos es una extensión de la programación procedural. Las operaciones individuales utilizadas en un programa de computadora se conocen como procesos, siendo este el origen del término de programación procedural.

Escribir programas de computadora que tienen una orientación a objetos tiene las siguientes características:

- Creación de clases
- Creación de objetos
- Creación de aplicaciones que pueden manipular o utilizar los objetos

En programación orientada a objetos se define a una clase como un grupo, colección de elementos con características similares. Se puede comparar con un plano que determina las características de una pieza. La definición de una clase existe antes de que los objetos sean creados. La definición de la clase determina los atributos que los objetos tendrán, los atributos son las características que determinan al objeto, o en otras palabras las propiedades del objeto.

Un objeto es una instancia específica de una clase. Los valores contenidos en las características del objeto diferencian a estos de los otros objetos en la misma clase, se toma como ejemplo el caso de los tipos de vehículos, en los cuales existe diferentes marcas

años modelos cada uno de estos vehículos es formado a partir de la misma clase pero tienen diferentes valores en sus características.

La encapsulación se define con la inclusión de información y métodos en el objeto, permite al usuario; o al programador, trabajar con todos los métodos de un objeto y características de este como una entidad individual.[14]

1.3.6.2 Eclipse IDE

Eclipse IDE, es conocido como un entorno de desarrollo integrado para Java. Actualmente es el líder en entornos de desarrollo para Java con un aproximado del 65% del uso para estas aplicaciones. De igual forma que el resto de los programas antes mencionados, eclipse está creado por una comunidad de código abierto y se usa en diferentes áreas, como las de desarrollo de aplicaciones para computadora y Android [15].

Una herramienta ampliamente usada para el desarrollo de aplicaciones en eclipse es el Windows maker o Windows builder. Esta herramienta permite la creación de pantallas, la modificación de estas, e inserción de botones selectores y algunos otros elementos comúnmente utilizados en interfaces de usuario, de tal forma que pueda ser utilizada por un operador. Este operador no requiere de conocimientos avanzados de programación para utilizar la aplicación desarrollada en Windows Builder. La herramienta de eclipse se usa para la creación de la interface hombre máquina que se desarrollará en el proyecto.[16] A continuación, se presenta un ejemplo de la interfaz del Windows maker utilizada para la creación de la pantalla de operador, en la figura 1.13.

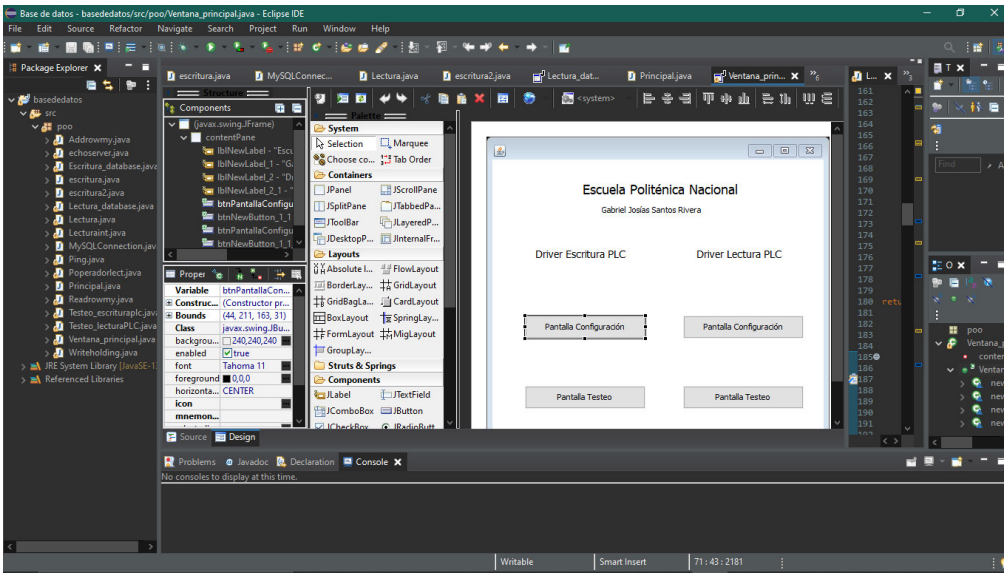


Figura 1.13. Pantalla para modificación de interfaz en Java en eclipse IDE

Uno de los objetivos del uso en JAVA es la capacidad de manejar información en una red de comunicación LAN, de modo que la programación sea capaz de comunicarse con el PLC mediante Profinet. Véase objetivos sección 1.1.

1.3.7 MYSQL DATABASE

El objetivo de este proyecto es la creación de un driver de comunicación capaz de conectar un software industrial, para interfaz de usuario y un controlador lógico programable, a través del protocolo Profinet. Debido a las características de este driver se requiere un intermediario para establecer la comunicación entre la interfaz y el PLC, esto es necesario ya que la interfaz hombre máquina no está desarrollada para trabajar con el protocolo Profinet, sino con OPC, un ejemplo claro de esto son los drivers ya existentes en la industria, mencionados anteriormente, para más información revisar tabla 1.1.

Uno de los objetivos del desarrollo del proyecto la implementación del driver mediante herramientas de software libre, a partir de este objetivo y el requerimiento de una base de datos relacional se seleccionó el uso del programa MySQL Workbench. Este programa de la familia MySQL permite la creación, modificación y uso de una base de datos relacional, con su respectiva comunicación en red. Esta aplicación de MySQL permite la creación de bases de datos de forma gratuita y la modificación de estas de manera sencilla, ya que al realizar cambios en las bases de datos se requiere conocer acerca de la programación en MySQL, sino que existe una interfaz en el programa que es amigable con el usuario, permitiendo la alteración de la información y configuración de las bases de datos [17].

La interfaz de usuario de MySQL Workbench, toma las instrucciones del operador para la base de datos y las adapta al lenguaje MySQL. Antes de ejecutar las instrucciones en el lenguaje MySQL permite observar los estamentos de programación, como se observa en la figura 1.14., siendo esto una herramienta importante para el desarrollo del método utilizado en el driver.

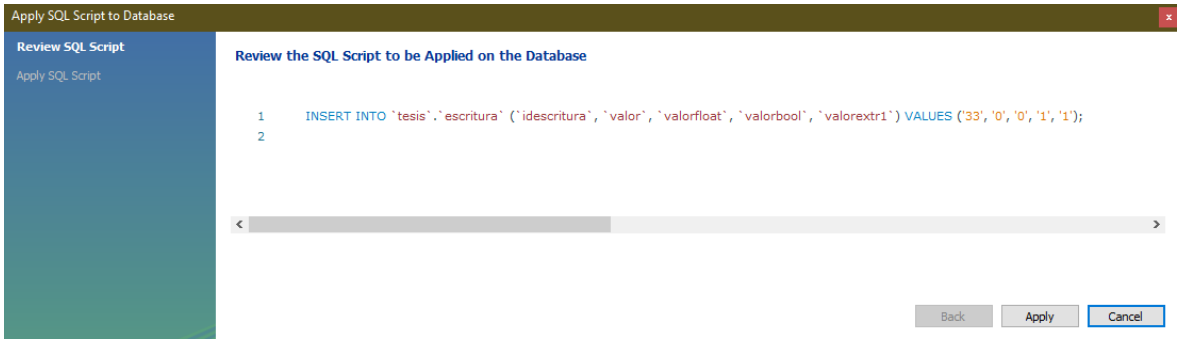


Figura 1.14. Pantalla previa a la aplicación de un estamento de MySQL en el programa MySQL Workbench.

1.3.7.1 Tipos de Datos en MySQL

Las características del proyecto desarrollado proponen la necesidad de conocer los tipos de variables que son manejables en MySQL. En la aplicación se consideran tanto datos booleanos, flotantes y enteros, pero como se ha mencionado anteriormente el abanico de tipos de información manejables en MySQL no se limita a estos. Se presentan los tipos de información existentes en MySQL en la tabla 1.2.

Tabla 1.2 Tipos de variables existentes en MySQL.[17]

Variable	Tipo	Bytes
Tinyint	Integer	8-bit
SmallInt	Integer	16-bits
MediumInt	Integer	24-bit
Int, Integer	Integer	31-bits
Big Int	Integer	64- bits (8 Bytes)
Float	Floating Point	4 bytes (8 place precision)
Double	Floating Point	8 bytes (16 place precision)
Decimal	Fixed Point	2 bytes
Date	Date	3 bytes
Time	Time	3 bytes
DateTime	Date- Time	8 bytes
Year	Year	1 byte
Timestamp	Date – Time	4 bytes

Char (n)	String	n bytes (maximum 255 characters)
VarChar (n)	String	n +1 bytes
TinyText	String	n+1 bytes
Text	String	n +2 bytes
Mediumtext	String	n +3bytes
LongText	String	n+4 bytes
Tinyblob	Binary Data	255 bytes

1.3.7.2 Sentencias Comunes en MySQL

Aparte del conocimiento de las variables en MySQL y sus características, es requerido tener en cuenta las sentencias que se ocupan en el lenguaje SQL, ya que durante el desarrollo del proyecto, Java debe ocupar estas para lectura, así como para escritura en la base de datos. A continuación, se presentan las sentencias principales y el uso:

- Select: Consulta de datos
- Insert: Insertar datos
- Update: Modificar datos de una tabla
- Delete: Eliminar datos de una tabla
- Order By: Se utiliza para ordenar los resultados
- Distinct: Eliminar los duplicados
- Where: Condición para ubicación de datos a consultar
- Create Table: Creación de tablas
- Drop Table: Eliminación de tablas
- Truncate: Elimina las filas de una tabla
- Alter Table: Modificar características de una columna en una tabla [6]

1.3.8 WONDERWARE

Wonderware es reconocida por su software de desarrollo de interfaces de operador para sistemas industriales, mejor conocida como Intouch. Esta herramienta de desarrollo de interfaces es muy utilizada en el mercado siendo, según Wonderware, aplicada en 1/3 de las plantas industriales a nivel mundial. Wonderware fue fundada con la idea, de que los operarios puedan monitorear las operaciones de una fábrica con mayor efectividad, como si las herramientas fueran más fáciles de utilizar, similar a la interfaz de un videojuego. En 1987 con la presentación de InTouch, Wonderware consiguió el primer interfaz hombre máquina basada en el sistema operativo Microsoft Windows [5].

1.3.8.1 Intouch

Intouch fue desarrollada como una herramienta para la creación de interfaces de usuario de tal forma que éstas pueden ser utilizadas para el monitoreo y control de una planta industrial. InTouch presenta una serie de herramientas y ventajas que lo caracterizan y lo convierten en uno, de los softwares de desarrollo más usados en la industria, entre estas ventajas se incluyen: la facilidad de incorporar gráficos, su versatilidad para trabajar con diferentes tipos de servidores de datos, bases de datos para el almacenamiento y manejo de información. Intouch tiene una cantidad considerable de herramientas para trabajar con bases de datos de MySQL, por lo que su adaptabilidad también favorece al desarrollo del proyecto, aunque su aplicación está limitada a la verificación de resultados en el mismo. En la figura 1.15 se puede observar la pantalla general de configuración de Intouch.

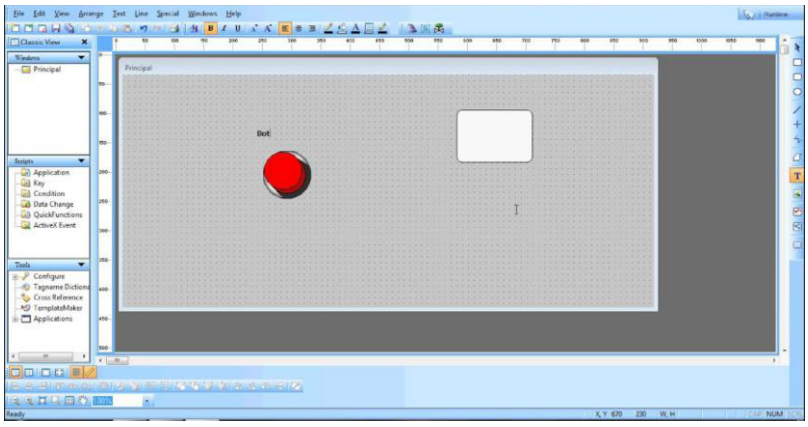


Figura 1.15. Pantalla para configuración y manejo de la aplicación e Intouch en la creación de interfaces industriales.

Como se ha mencionado antes, véase tabla 1.1, InTouch tiene una versión de prueba la cual permite la configuración y modificación de pantallas de operador en una interfaz, con

una limitación temporal. De la misma forma, se requiere un servidor de datos para realizar la aplicación en el protocolo OPC en un PLC, siendo el más utilizado para el entorno de Intouch el programa Archestra, que se está conformado por un DAServer [18].

1.3.8.2 Archestra

Archestra es la arquitectura tecnológica desarrollada por Wonderware, basada en .NET, para Microsoft Windows. Este puede conectarse con la herramienta de Intouch y a su vez con los PLC de diferentes fabricantes, a través de diferentes protocolos. Aunque, se encuentra limitado debido a las licencias que manejan tanto la herramienta de desarrollo de interfaz, así como la herramienta de programación de controlador, haciendo referencia a la cantidad de tags que se permitan trabajar en las licencias de software. Archestra, no es un servidor de datos, sino es el conjunto de los servidores de datos para diferentes protocolos y fabricantes que se encuentran desarrolladas por Wonderware.[5]

El desarrollo del proyecto toma como referencia el servidor DASSIDirect de Wonderware, perteneciente a Archestra, que se encuentra orientado específicamente a su uso en la familia de PLC S7 de Siemens. El objetivo del uso de este servidor en el proyecto es la comparación de las características de este servidor, con la del driver desarrollado en Java.

2. METODOLOGÍA

Teniendo en cuenta el estado del arte referente al entorno de trabajo para el desarrollo del driver, en este capítulo se explica el diseño, el procedimiento y el método utilizados para la programación del sistema, así como para el desarrollo de pruebas de funcionamiento previas a los resultados finales.

En este capítulo se realiza un análisis de la trama en el protocolo Profinet, a través de la herramienta Wireshark. Se considera la configuración requerida y los métodos necesarios para el desarrollo adecuado de un sistema de recepción y envío de información a un controlador, el manejo de los datos, las librerías requeridas y los métodos relacionados.

Se analiza el diseño del servidor y su implementación en el software libre Java. Se considera la arquitectura y su requerimiento de una base de datos relacional, así como la interacción existente entre la base de datos, la interfaz de usuario y el driver.

Se realiza un sistema de pruebas, que sea capaz de verificar el funcionamiento de los sistemas de comunicación con el PLC y con la base de datos. Este sistema de pruebas inicial no está orientado a su uso con un interfaz de usuario, sino solamente con código; es decir que ocupa únicamente estamentos relacionados a la comunicación mas no a la

creación de un HMI. Se utiliza posteriormente como una subfunción accesible desde cualquier parte de la programación que se utilizará para la implementación final del driver.

2.1. ARQUITECTURA DEL SISTEMA

El sistema está desarrollado de tal forma que puede intercambiar datos con cualquier aplicación que tenga acceso a una base de datos tipo MySQL. Se propone una arquitectura que ocupe la base de datos como un intermediario, permitiendo la conexión y el paso de información desde el controlador hacia la interfaz del operador. El driver toma los datos de la interface desarrollada para la configuración de la comunicación, tanto para el PLC como para la base de datos, y utilizando las capacidades de intercambio de información programadas, guarda o recupera registros de la base de datos siendo estos los requeridos para ser escrita en el PLC o leídos desde el PLC.

El driver está siendo desarrollado para reemplazar a un servidor OPC. Al momento de programar el controlador se requiere la integración, en la programación de red de TIA Portal, a un PLC, que va a ser reemplazado por el driver. En la imagen 2.1 se muestra la arquitectura planteada para el desarrollo del driver.

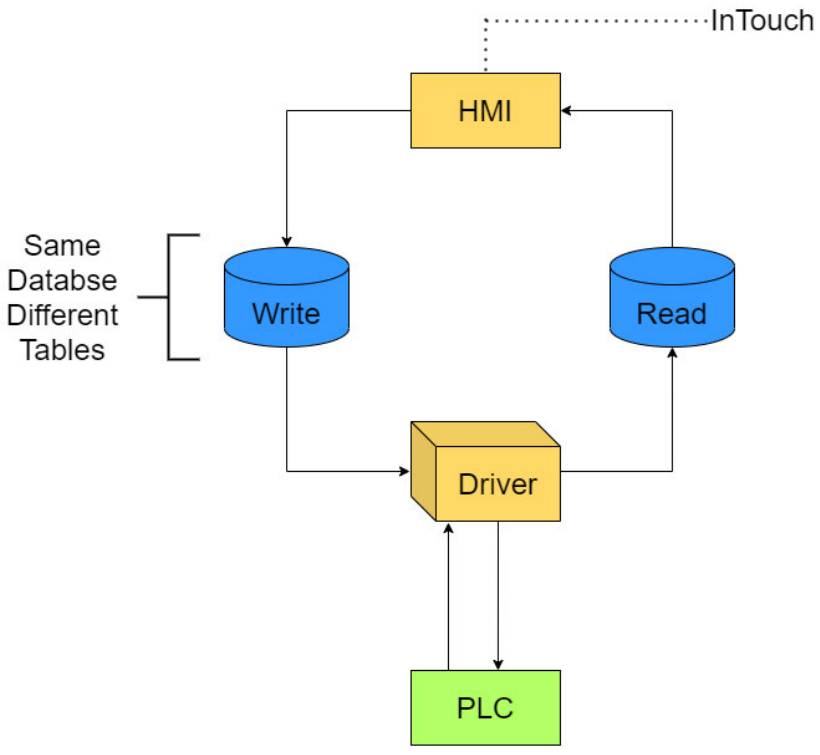


Figura 2.1. Arquitectura propuesta para el desarrollo del driver

En cuanto a la arquitectura, el alcance del proyecto se centra en el desarrollo del driver en el lenguaje de programación Java, así como en la configuración y adaptación de las tablas en la base de datos, que sirven como intermediario entre el controlador industrial y el conector ODBC que envía la información a la interfaz del operador.

El flujo de datos del sistema en un sentido inicia con la información existente en el PLC. Esta se encuentra almacenada en un tag específico que está configurado en un bloque de envío, TSEND_C del PLC. El bloque envía los datos mediante el protocolo Profinet, a través de la red de comunicación, hacia el driver en Java. El objetivo del driver es ser capaz de aceptar la comunicación y con el método interno, adaptar la información obtenida de tal forma que esta pueda ser entendida por la base de datos. La segunda responsabilidad del driver es escribir en la base de datos, utilizando las sentencias apropiadas en las librerías de Java, los registros adaptados obtenidos, de tal forma que se pueda recopilar los mismos desde la herramienta de desarrollo de HMI, en este caso InTouch.

En sentido contrario los datos se originan en la interfaz del operador y gracias a la configuración estos son guardados en la base de datos. El driver tiene la responsabilidad de recoger la información requerida desde las tablas de la base de datos y traducirla para que esta pueda ser procesada por el PLC, en este proceso el driver es el responsable de generar la trama Profinet. El bloque de recepción del PLC obtiene la información y la misma es utilizada según los requerimientos de la aplicación.

2.2. REQUERIMIENTOS DEL SISTEMA

Siguiendo lo establecido en los objetivos, se debe desarrollar un prototipo orientado al protocolo Profinet, desarrollado en software libre; que sea capaz de ser comparable con un driver de comunicación Industrial. El prototipo desarrollado debe ser capaz de manejar protocolos de comunicación basados en el modelo OSI; esto incluye: Ethernet 2, TCP, IP. El software utilizado para realizar la programación debe tener la habilidad de manipular y modificar estos encabezados, específicamente los puertos utilizados en el protocolo TCP.

Recapitulando los tipos de Profinet considerados por Profinet Foundation, el prototipo a implementar debe cumplir con la CC-A, en una red de Profinet CBA o I/O, y orientado a Profinet RT.

A partir del análisis realizado del funcionamiento de los drivers industriales OPC existentes en la industria, se establece el requerimiento de una base de datos relacional que pueda trabajar con un sistema en red así como el lenguaje de programación establecido para la

aplicación. La base de datos debe ser un sistema de software libre, siguiendo con los objetivos del proyecto.

2.3. SELECCIÓN DE SOFTWARE

Teniendo en cuenta los requerimientos del sistema basados en los objetivos del proyecto, se selecciona un software para programación, así como un sistema de base de datos relacional. Las opciones consideradas tanto para el software de programación así como para la base de datos dependen de los conocimientos previos del desarrollador del sistema, y de la accesibilidad a la información necesaria para su implementación en cada una de las opciones.

2.3.1 SOFTWARE DE PROGRAMACIÓN

Se consideran diferentes lenguajes y opciones para el sistema de programación, priorizando la aplicación en software libre. Se tomó en cuenta lenguajes ya revisados para este tipo de aplicaciones, así como conocimientos previos en cada uno de estos; el resumen de las opciones se encuentra en la tabla 2.1.

Tabla 2.1. Lenguajes de programación considerados

Lenguaje	Manejo Encabezados	Manejo Bases de Datos	Software Libre	IDE
C	Existencia de librerías	Existencia de librerías	Cumple con el requerimiento de software libre	Notepad++ No orientado a interfaz de usuario
Java	Existencia de librerías y trabajos externos con su uso [6]	Existencia de librerías	Cumple con el requerimiento de software libre	Eclipse IDE Orientado a Objetos y

				capacidad de interfaz
Matlab	Existencia de librerías, limitación de manejo.	Existencia de librerías.	Costo por licencia de manejo.	Matlab IDE
LabView	Existencia de librerías, limitación de manejo	Existencia de librerías	Costro por licencia de manejo	Labview IDE

Como se puede observar en la tabla 2.1, Java presenta el lenguaje de programación más adecuado para la implementación del prototipo. Siendo este un lenguaje orientado a objetos, que tiene las herramientas necesarias para el cumplimiento de los objetivos del driver. Este lenguaje tiene la capacidad de manejar de forma independiente los encabezados en TCP, siendo este el requerimiento más importante para la comunicación Profinet, considerando a su vez que se tiene conocimiento de sus librerías con manejo de lenguaje MySQL para modificación de bases de datos.

2.3.2. BASE DE DATOS

La selección de base de datos toma en cuenta las opciones existentes en MySQL, teniendo en cuenta que las librerías consideradas en la selección de software en Java manejan este lenguaje de programación de base de datos. En el abanico de aplicaciones de MySQL existe una variedad considerable de herramientas las cuales tienen un funcionamiento similar, esto nos orienta a seleccionar la GUI de trabajo en función de la experiencia de manejo de esta. Por esta razón se toma en cuenta MySQL Workbench gracias a su interfaz gráfica amigable con el usuario y a la experiencia previa existente con el manejo de esta herramienta.

2.4. ORGANIZACIÓN DE SOFTWARE A IMPLEMENTAR

Java, como ya se mencionó anteriormente, utiliza clases para su programación, estas deben estar organizadas para cada una de las acciones y subrutinas existentes en el

método a implementar. Para el desarrollo del prototipo se tiene en cuenta la organización de cada una de las pantallas, las diferentes rutinas existentes en cada una de las clases y cómo se usan cada una de estas para acceder a la base de datos o al controlador.

2.3.1 MANEJO DE LAS PANTALLAS

Las pantallas presentan las diferentes capacidades del driver. Cada una de estas tiene sus características y necesidades, las cuales se presentarán más adelante en el desarrollo de sus interfaces de usuario y métodos a utilizar. Se presenta el manejo de las pantallas, como se accede a cada una de estas, el nombre y el objetivo general de cada una. Las pantallas tipo JFrame, se encuentran clasificadas en dos tipos: el primero es las pantallas de envió de información hacia el PLC y está su contraparte que es la recepción de datos desde del PLC.

En la figura 2.2 se puede observar la organización de las pantallas. Siendo la pantalla principal aquella que permite el acceso a las demás. El programa inicia en una clase que permite el acceso a la pantalla principal, que esta implementada en otra clase. Con esta pantalla principal el usuario puede acceder a las dos pantallas de prueba y a las pantallas de configuración del driver. Las pantallas de prueba tienen como objetivo la verificación de las características de comunicación entre Java y el PLC, de tal forma que la configuración posterior del driver se realice de forma adecuada. Esta verificación de comunicación establecida en las pantallas del driver ocurre únicamente entre este y el PLC.

Las pantallas de configuración son la antesala a las pantallas de operador que son el resultado final del driver. Estas pantallas tienen la capacidad de probar el sistema de comunicación con la base de datos y el PLC simultáneamente, de tal forma, que se pueda comprobar el correcto funcionamiento del driver antes de la puesta en marcha y corregir errores en los datos proporcionados al driver referentes a las características de envío de datos entre las diferentes partes del sistema. Finalmente en la pantalla del operador, únicamente se puede comprobar el funcionamiento del driver, sin modificar sus parámetros y a su vez, el dato que pasa a través de este.

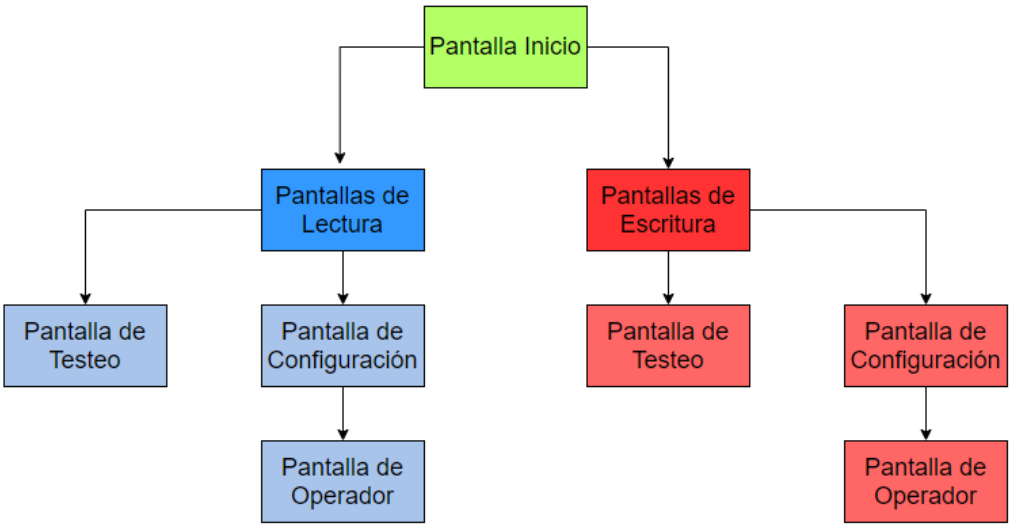


Figura 2.2. Organización de las Pantallas en el driver.

2.3.2 USO Y MANEJO DE LAS CLASES

Partiendo de esta organización de pantallas, se tiene un uso de clases y de subrutinas existentes en cada una de estas, según los requerimientos de la pantalla. Estas subrutinas a su vez estas llaman a otras dependiendo de la organización del método, que trabaja a favor del cumplimiento de las tareas específicas del driver.

En la figura 2.3, se presenta las subrutinas llamadas por cada una de las pantallas, siendo en total un número de 6 pantallas. Para la figura 2.3 no se tienen en cuenta las pantallas de operador, ya que estas ocupan las subrutinas que sus pantallas de configuración.

Más adelante se explicará cada método con detenimiento, enfatizando en aplicabilidad, manejo de la base de datos y programación requerida en el lenguaje MySQL.

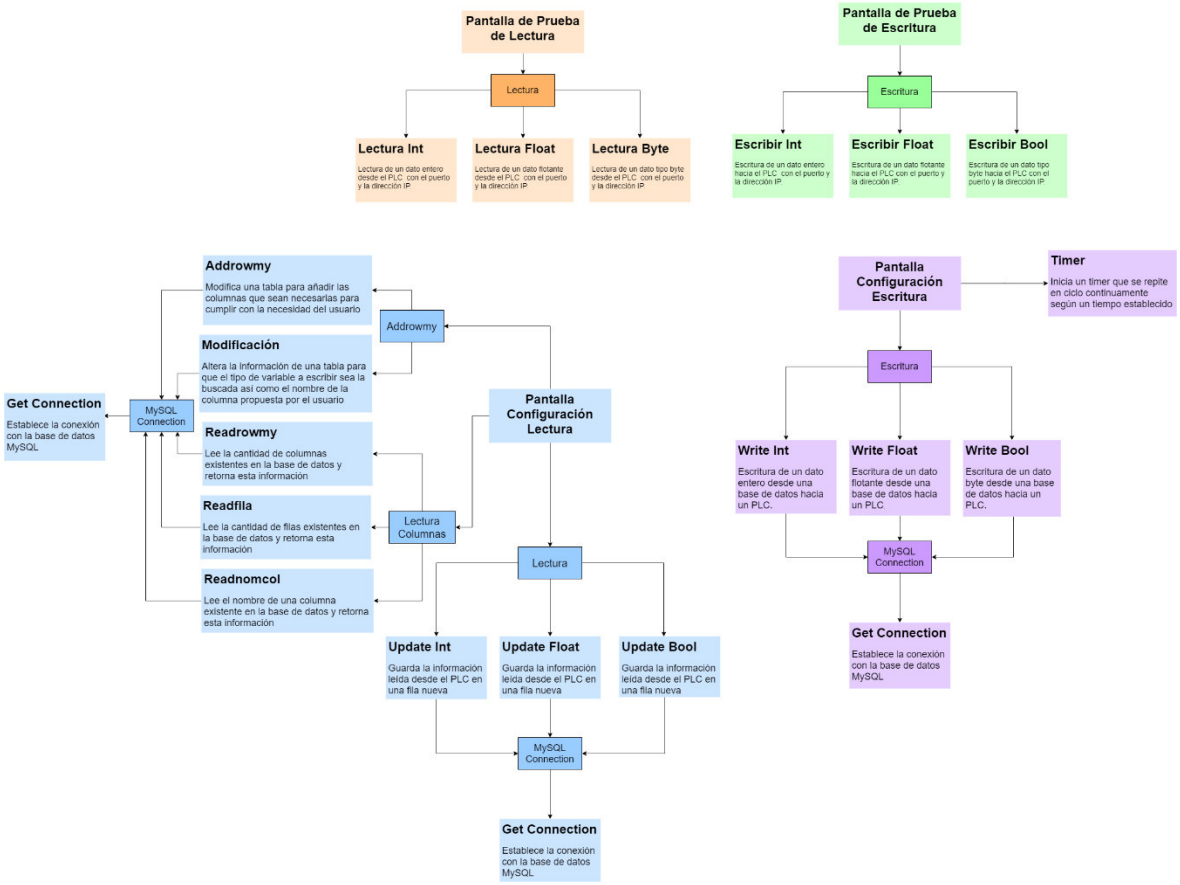


Figura 2.3. Organización de las clases y subrutinas de cada una de las clases del driver, véase figuras 2.4, 2.5, 2.6 y 2.7 para información detallada.

La figura 2.3 presenta la configuración y el uso de cada una de las clases existentes en el driver para el manejo de protocolo de Profinet.

El diagrama se encuentra dividido en dos grandes grupos, las pantallas de lectura y de escritura. Siendo estas representadas por un bloque solamente con su nombre y en letra negrilla. Las clases a las cuales se acceden se encuentran representadas por los bloques a su alrededor siendo las dos clases principales: *Lectura y Escritura.* De estas clases se derivan las principales subrutinas que permiten la lectura y escritura; en la base de datos y en el PLC.

Las subrutinas pueden llamar a otras clases en función de los requerimientos del método. Un ejemplo claro es el uso del bloque de conexión de MySQL, que es usado comúnmente en la escritura y lectura de la base de datos.

2.3.2.1 Clases y Subrutinas en Pantalla de Prueba de Escritura

Para empezar el desglose de información referente al manejo de las clases, se analiza la clasificación del uso de estas, considerando las pantallas que las requieren. La pantalla de

prueba de escritura se maneja con 3 subrutinas existentes en la clase *escritura*, el uso de estas depende del requerimiento de la aplicación. Véase figura 2.4. y 2.3.

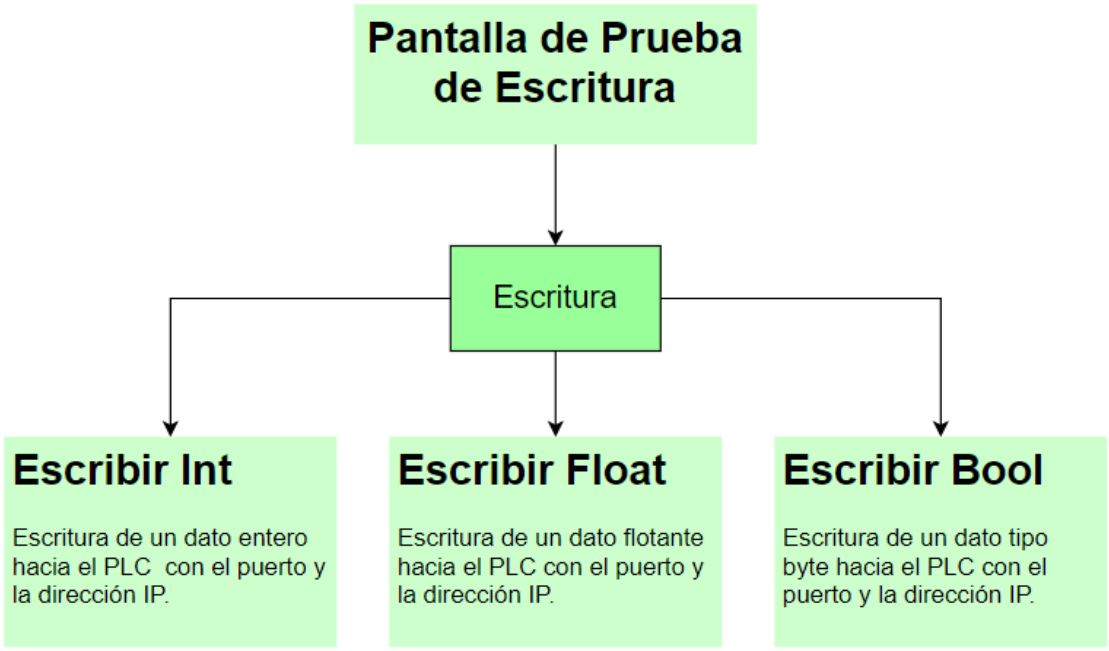


Figura 2.4. Organización de las clases y subrutinas utilizadas en la pantalla de prueba de escritura en el driver.

Dentro del driver las principales clases son las de escritura y lectura. En estas se encuentran implementadas todas las subrutinas de trabajo con el PLC y la base de datos, esto incluye las requeridas para el uso de la pantalla de prueba. El objetivo de esta pantalla es la verificación de los parámetros de comunicación con el PLC, en este caso la escritura de variables en el controlador.

Las subrutinas creadas en la clase escritura, que se ocupan en la pantalla de prueba se diferencian entre ellas por el tipo de dato con el cual pueden trabajar, siendo en total 3: entero, punto flotante y booleano.

2.3.2.2 Clases y Subrutinas en Pantalla de Prueba de Lectura

De la misma forma que se tiene un conjunto de subrutinas desarrolladas para la pantalla de escritura, se implementaron subrutinas gemelas para la pantalla de lectura. Estas fueron creadas con el objetivo de leer información desde el PLC sin intervención de ningún programa externo y se diferencian entre sí por el tipo de dato con el cual pueden trabajar, véase figura 2.5.

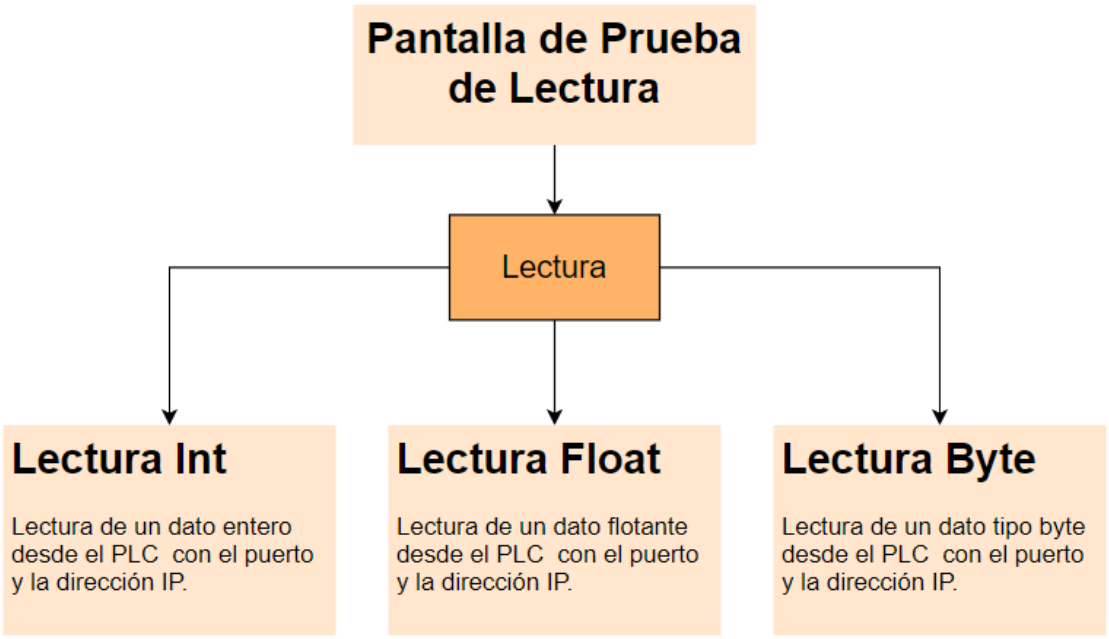


Figura 2.5. Organización de las clases y subrutinas utilizadas en la pantalla de prueba de lectura en el driver.

Las subrutinas creadas se encuentran en la clase lectura; una de las clases principales en el programa, en la cual se encuentran implementadas todas las subfunciones de lectura del PLC que utiliza el driver. La clase lectura es capaz de manejar datos tipos booleanos, enteros y flotantes.

2.3.2.3 Clases y Subrutinas en Pantalla de Configuración de Escritura

Entre las pantallas de escritura y de lectura, las pantallas de escritura en PLC presentan la configuración más simple de clases y subrutinas, en comparación a los sistemas de lectura. Debido a las complicaciones existentes para la escritura de datos de forma adecuada en la base de datos, la pantalla de lectura utiliza una cantidad considerablemente mayor de subrutinas y de clases.

Las pantallas de configuración, así como las pantallas de operador utilizan un proceso cíclico debido al tipo en enlace de datos utilizado para la comunicación, para más información véase clasificaciones de Profinet tema 1.3.3.1. Se utiliza un temporizador que permite la repetición del proceso de escritura en un tiempo determinado. Este temporizador utiliza el llamado de una subfunción llamada *"Timer"* en una librería existente en el software de programación Java.

La pantalla ocupa subfunciones de la clase escritura que se clasifican dependiendo del tipo de dato a manejar. Estas subrutinas utilizan estamentos de obtención de información de base de datos, los datos a enviar se originan desde el HMI, son pasados a la base de datos para ser leídos por el driver y enviados al PLC.

Los estamentos de lectura de información de base de datos requieren de una conexión con la misma. La clase "*MySQL GetConnection*" es la encargada de establecer esta, por lo que es utilizada por cada una de las subrutinas ocupadas en la escritura de datos, véase figura 2.6 donde se presenta la configuración de clases y rutinas en la pantalla de configuración.

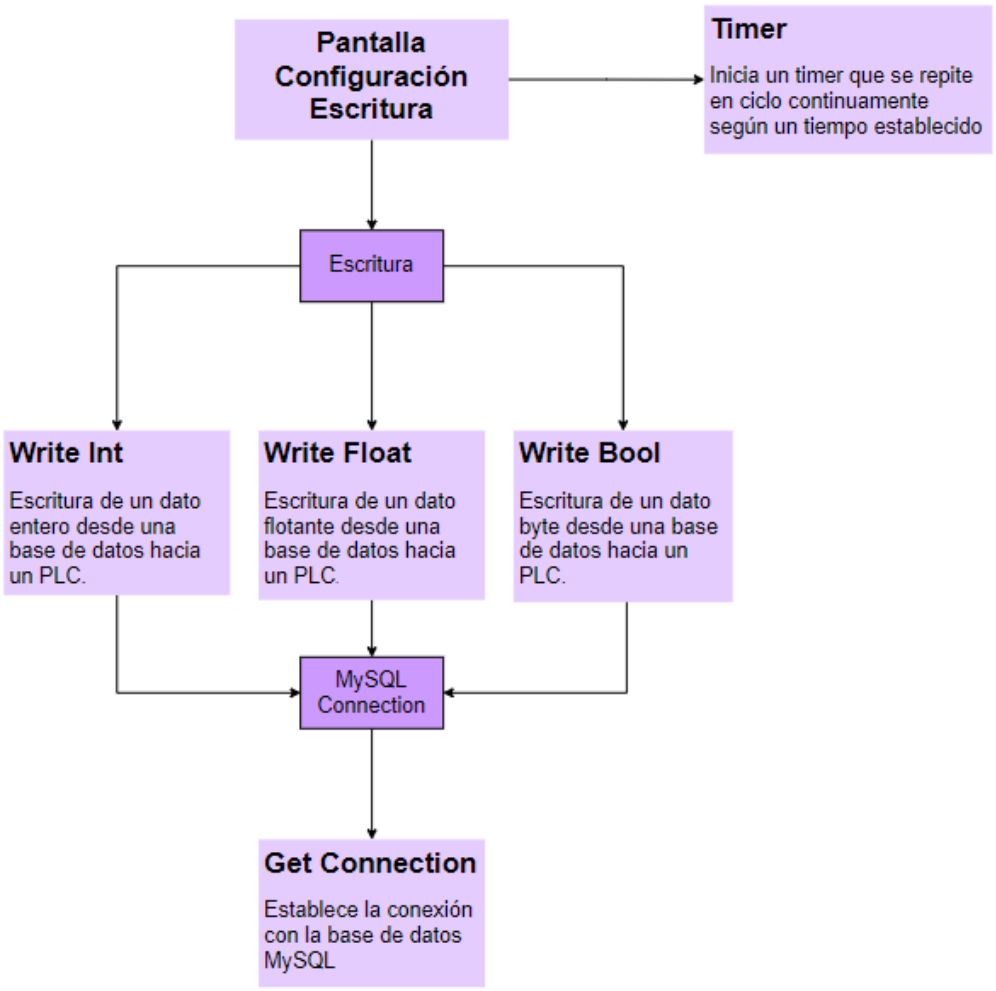


Figura 2.6. Organización de las clases y subrutinas utilizadas en la pantalla de configuración de escritura en el driver.

2.3.2.4 Clases y Subrutinas en Pantalla de Configuración de Lectura

Las pantallas de configuración y de operador de lectura, son aquellas que ocupan una cantidad mayor de clases y subrutinas. El método de escritura en la base de datos requiere de una cantidad considerable de información de la configuración preexistente en MySQL,

partiendo de esta necesidad existe un grupo de clases orientada al propósito de obtención de información.

Las clases utilizadas para obtener información de configuración de bases de datos son: "*Addrowmy*" y "*Lectura de Columnas",* véase figura 2.7. En estas clases se encuentran implementadas 2 subfunciones, las cuales son: "*Addrowmy*" y "*Modificación*"; que se encargan de preparar la base de datos para recibir la información obtenida desde el PLC. Las funciones que se encargan de obtener información de la configuración de la base de datos son: "*Readrowmy", "Readfila"* y *"Readnomcol"*.

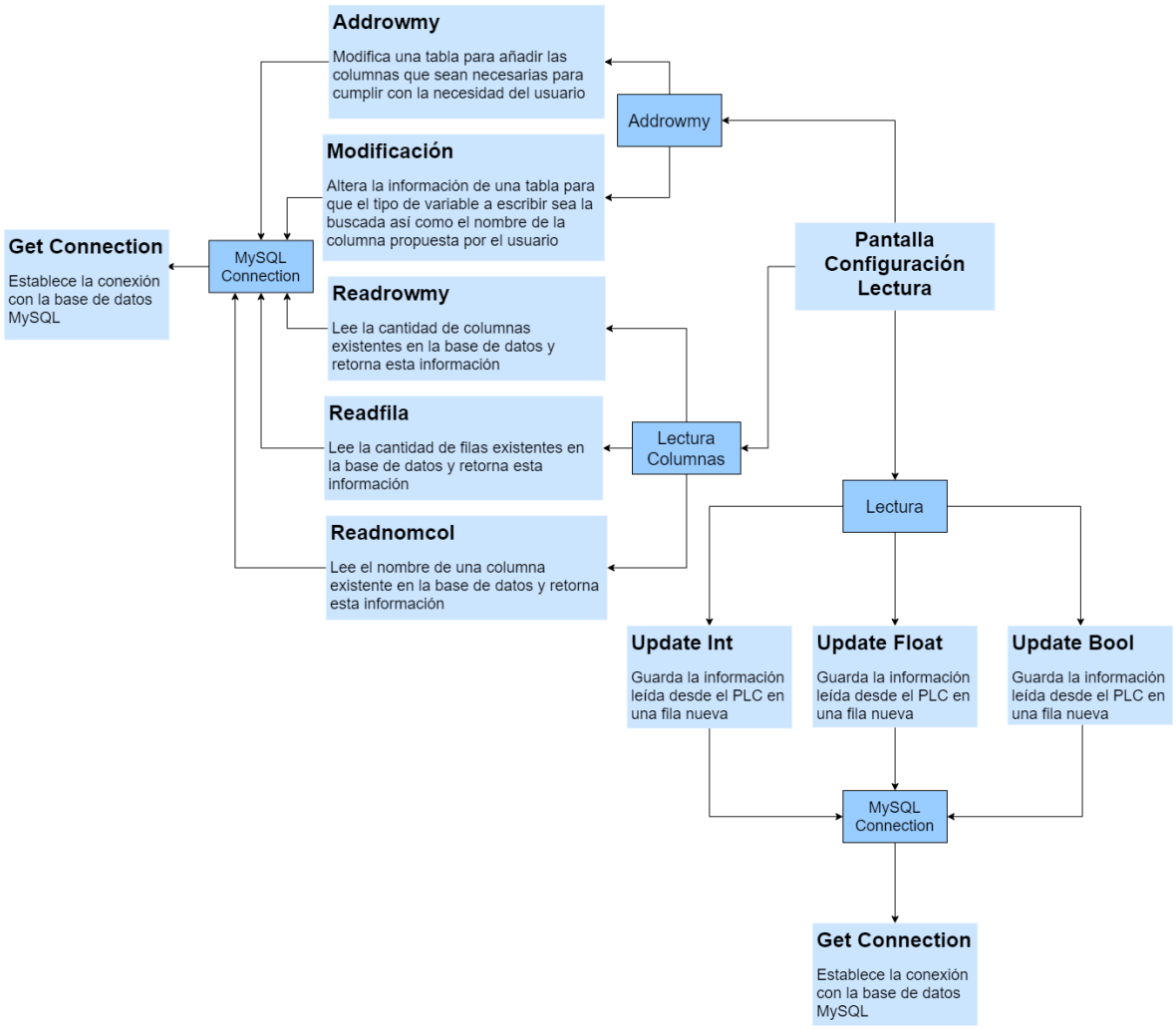


Figura 2.7. Organización de las clases y subrutinas utilizadas en la pantalla de configuración de lectura en el driver.

2.5. PROGRAMACIÓN DE LOS BLOQUES INTERNOS DE ENVÍO Y RECEPCIÓN DE DATOS

Para el correcto funcionamiento del driver tienen que estar programados los bloques de comunicación para Profinet en el PLC. Estos bloques son diferentes para ambos procesos de escritura y lectura. Al momento de insertar un bloque de comunicación se genera automáticamente una base de datos interna en el PLC, que sirve para el almacenamiento de la información correspondiente al estado y características del bloque. Gracias a la ayuda existente en el programa de desarrollo, TIA portal, se puede programar de forma individual cada uno de los terminales de entrada y salida de los bloques de envío y recepción de información. Se presenta el bloque de envío, su configuración y la información correspondiente referente a cada una de las partes de este en la figura 2.8.

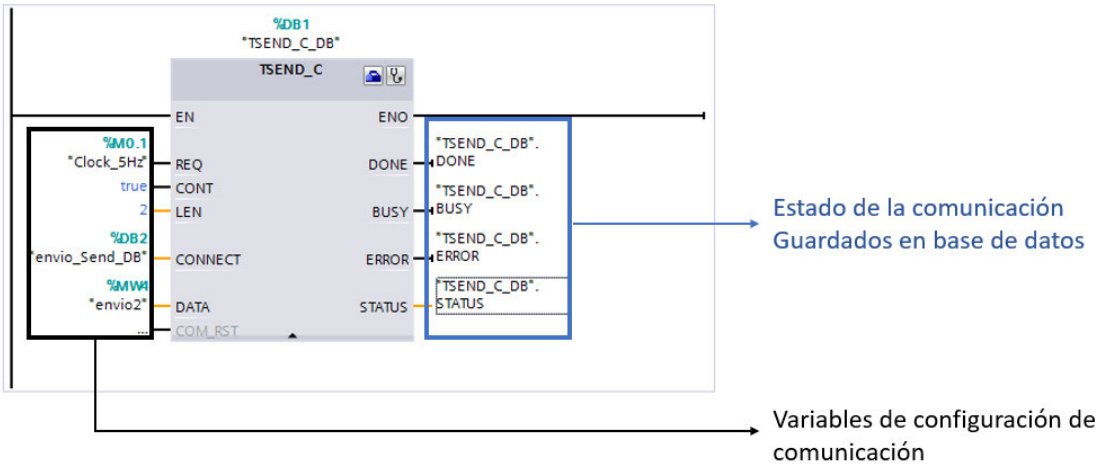


Figura 2.8 Bloque de envió de datos. Se diferencia las variables que presentan el estado del bloque y aquellas que sirven para su configuración.

Para empezar la configuración del bloque de envío de datos, se tiene en cuenta la entrada de activación de requerimiento, “REQ”. Esta entrada se activa mediante un flanco de subida y activa el bloque completo e inicia el requerimiento de información, puede compararse con la apertura del puerto para inicio de recepción datos. Posterior a esta entrada del bloque tenemos la de control con las siglas “CONT”, la misma está orientada a realizar un control de la conexión de comunicación, usualmente se recomienda que este valor se encuentre siempre en uno. Es requerido determinar la longitud en bytes del mensaje enviado, establecido en la entrada de longitud “LEN”, se recomienda siempre ocupar el número de bytes máximo existente en el dato enviado. Se puede aumentar este número de bytes, para envío de conjuntos de datos más complejos, pero debido a las características de la trama

las posibilidades de fallo son considerablemente mayores. El terminal data se encarga de receptar la información que se desea enviar, que puede ser una entrada del PLC o una marca, siempre y cuando se encuentre nombrada en los tags del controlador. El último terminal de entrada sirve para un reinicio de la comunicación, usualmente no se requiere la activación del mismo aunque, pueden existir aplicaciones en las cuales su uso sea necesario.

Al lado derecho del bloque existen los terminales de salida, revisar figura 1.11, que envían información a zonas específicas de la base de datos, generada, como antes se mencionó; paralelamente a la creación del bloque, estos datos generados determinan las características de la conexión y estado del bloque. En primera instancia, se encuentran los terminales de finalización, de ocupado, o de error; cada uno de estos terminales se representa mediante un valor tipo bit y como sus nombres indican, muestran el estado de la comunicación. Para entender de forma más específica cuál es el estado de la comunicación y del bloque en un momento determinado, se utiliza el terminal *STATUS,* que mediante un código entregado, el cual debe ser revisado en la literatura entregada por Siemens referente a la configuración del bloque, en una variable tipo Word, permite determinar cuál es el estatus de la comunicación y la causa de un posible error, en caso de existir uno.

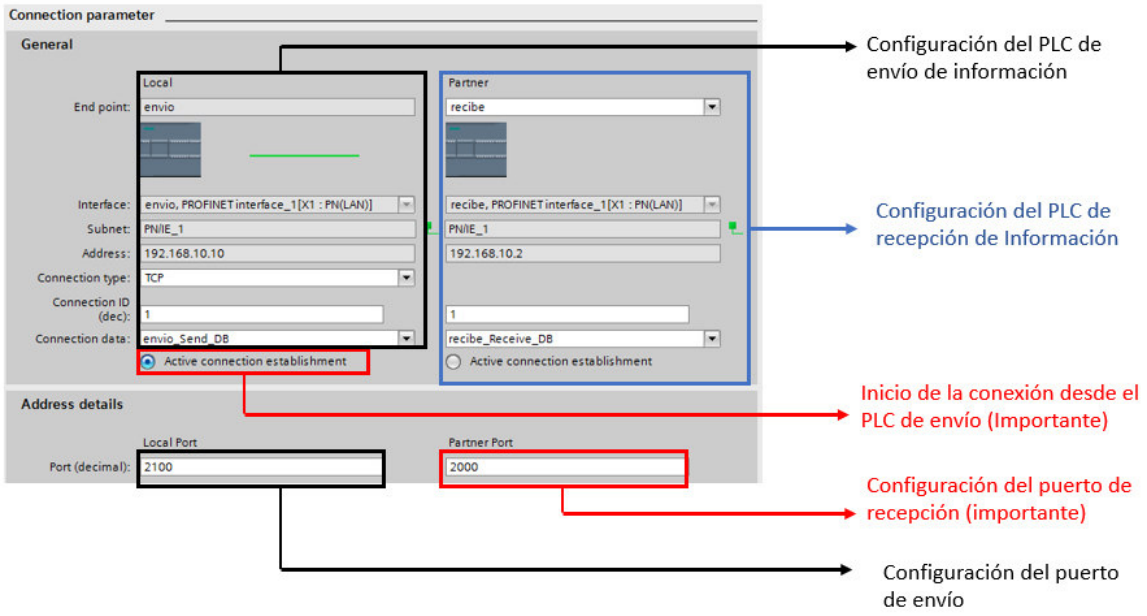


Figura 2.9 Pantalla de configuración de comunicación entre PLC en Profinet. Se enfatiza las partes de configuración importantes para el funcionamiento del sistema.

Partiendo de esta configuración previamente existente se desarrolla la configuración interna del bloque, la misma se puede observar en la figura 2.9 La configuración interna del bloque permite seleccionar el PLC al que se espera enviar los datos. Gracias a las características de la programación de TIA portal, al seleccionar el PLC en el que se quiere recibir los datos, se rellena de forma automática la información referente a sus características de red. A partir de esta preconfiguración se da un identificador de conexión y también un nombre de conexión, que se escriben automáticamente en uno de los terminales de entrada al cerrar esta pantalla de configuración, véase figura 2.8.

Es necesario seleccionar que la comunicación sea iniciada por el controlador que envía la información, esta opción se encuentra directamente debajo del nombre de la conexión. Es requerido seleccionar los puertos tanto en el PLC de recepción como en el envío, en los que se trabajará para realizar esta conexión específica. En la figura 1.12 se pueden observar estos dos puntos de configuración marcados en rojo, debido a que son indispensables para el correcto funcionamiento del driver.

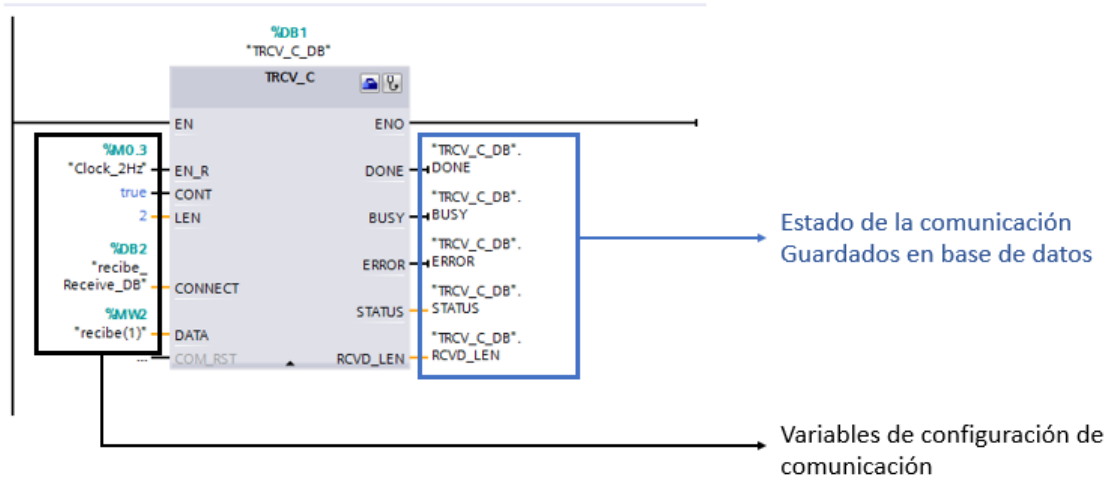


Figura 2.10 Bloque de recepción de datos. Se diferencia las variables que presentan el estado del bloque y aquellas que sirven para su configuración.

La configuración del bloque de recepción de datos, que se encuentra en la figura 2.10, contempla la activación del bloque, este terminal debe ser activado por una marca de tiempo que se habilita previamente en el controlador. El siguiente terminal es el de control, que ya fue explicado en la anterior sección; que sirve para el encendido o apagado del sistema y se recomienda tenerlo activado para el funcionamiento adecuado. Posteriormente se encuentra el terminal de la longitud del dato a enviar en bytes, se

recomienda que longitud sea exactamente la del campo data. El terminal de data recepta la información que va a ser utilizada en la comunicación.

Los terminales de salida contemplan 3 valores, de la misma forma que con el bloque de envío. Estos representan: la finalización del envío, la ocupación del bloque y la existencia de un error, en caso de existir uno. Existen 2 terminales más, que presentan las características del estatus del bloque al momento de realizar la comunicación y la cantidad de bytes recibidos.

Se configura, igual que su contraparte de envío, el bloque de recepción de datos, en este caso se puede hacer referencia a la figura 2.8 en la cual se presenta a la configuración básica para el bloque de comunicación. En el caso del bloque de recepción de datos se debe seleccionar que este no genere el inicio de la comunicación sino que acepte esta.

2.6. MANEJO Y ESTRUCTURA DE LAS TRAMAS EN PROFINET EN TCP

La característica principal del driver es su manejo del protocolo Profinet sobre TCP. El driver debe ser capaz de manejar el protocolo, su funcionamiento y trabajar con las diferentes variables en este. La trama de envió se encuentra compuesta por encabezados por varios protocolos, siendo uno de ellos el protocolo TCP.

2.6.1 OBTENCIÓN DE LA TRAMA PROFINET

El desarrollo del driver considera inicialmente el análisis de la trama Profinet. Para revisar la trama se tiene en cuenta que, a menos que un mensaje este dirigido directamente a la dirección de la computadora; esta no es capaz de leer los bytes de la trama; la razón de esto es que el switch, que genera la red LAN y que conecta al sistema el PC y PLC, utiliza la dirección MAC para el enrutamiento de la información. Los datos enviados desde un PLC a otro no pueden ser leídos por el programa Wireshark, ya que este se encuentra en una computadora con una dirección MAC diferente.

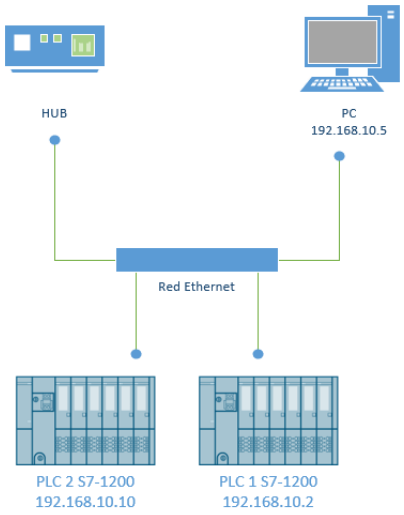


Figura 2.11. Conexión en red de PLC con el HUB para análisis de Profinet.

El diseño del driver requiere la revisión del entramado Profinet, se utilizaron herramientas de red que no ocupen la opción de conmutación, o que permitan el análisis del tráfico de datos en un elemento de conmutación, como es el caso del switch. Se reemplazó el switch por un HUB, conociendo que este elemento de red tiene la característica de siempre enviar todos los paquetes de datos a todos los elementos de la red sin distinción de su dirección IP o su dirección MAC. En la figura 2.11 se presenta la configuración de red utilizada para el cumplimiento del objetivo de análisis de la trama Profinet antes presentado.

Posterior a haber realizado la conexión, se configura una comunicación de un byte entre ambos PLC, de tal forma que esta pueda ser leída por el programa Wireshark. En la figura 2.12 se presenta una captura tomada de un entramado, en la que se puede identificar, de forma general, las configuraciones de los protocolos antes mencionados, para más información véase Protocolo Profinet 1.3.2. Wireshark muestra las configuraciones generales de cada encabezado, centrándose en los datos de identificación del receptor y el emisor.

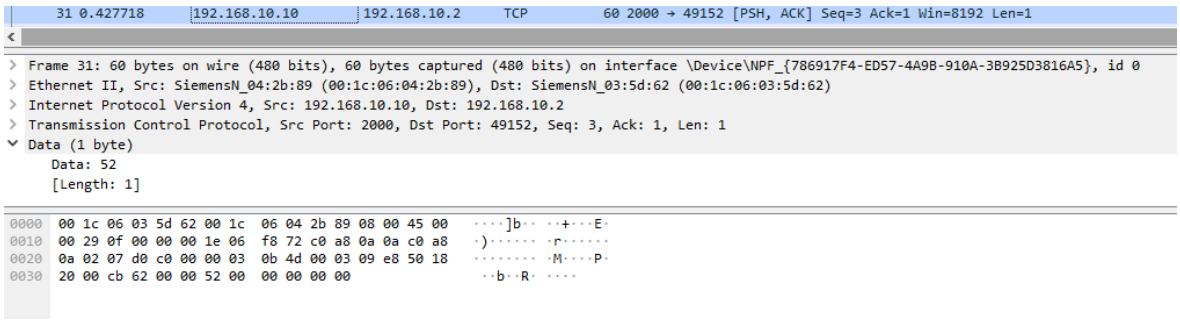


Figura 2.12. Entramado Profinet obtenido de Wireshark de PLC a PLC 1 byte

2.6.2 INICIO DE COMUNICACIÓN PROFINET

La característica esencial en el proceso de comunicación es el conocimiento de cuál es el elemento que la inicia, esto se puede observar en la herramienta Wireshark, al inicio de la comunicación entre 2 elementos mediante un protocolo Profinet. Se presenta un análisis del inicio de la comunicación, para el entendimiento del funcionamiento del protocolo Profinet en la red de datos, entre 2 PLCs.

2.6.2.1 Inicialización de Comunicación

Se considera el PLC que envía la información como PLC1 y al PLC que recibe la información como PLC2, en favor de la explicación del funcionamiento del entramado Profinet.

El elemento que inicia la comunicación, PLC1, desarrolla un mensaje de broadcast cuando el puerto se encuentra abierto, este mensaje de broadcast pregunta a todos los elementos en red, cuál es la dirección MAC de una dirección IP definida. La respuesta se entrega desde el PLC2 de recepción de datos, únicamente mencionando su dirección MAC y gracias a las características del encabezado de la trama se puede confirmar la dirección IP requerida para el inicio de la comunicación.

Después de que el PLC1 que inicia la comunicación conoce cuál es la dirección MAC y la existencia del PLC2 que recibe la información, se envía un mensaje de sincronismo. Este mensaje no tiene información en la trama enviada, sino que se caracteriza porque en el encabezado tipo TCP se activa una bandera llamada SYN, bandera de sincronismo, indicando que la trama enviada es comprueba el inicio de la comunicación entre los controladores. Para identificar esta bandera se utiliza la herramienta de Wireshark, existe la facilidad que trama está marcada con la bandera de activación de mensaje de sincronismo (SYN).

Gracias a las características del protocolo TCP, como respuesta de un mensaje enviado, se responde con un reconocimiento de este, el destinatario tiene que reenviar un mensaje hacia el origen de la información indicando que los datos fueron recibidos de forma satisfactoria, como se presenta en la figura 2.13, este proceso se conoce como *"timeout retransmission*"[10]. Ejemplificando, posterior al envío de un mensaje de sincronismo para inicialización de la comunicación, existe un reconocimiento de este que es enviado directamente desde el destino, que se caracteriza porque las banderas de sincronismo y de reconocimiento se encuentran levantadas (ACK y SYN).

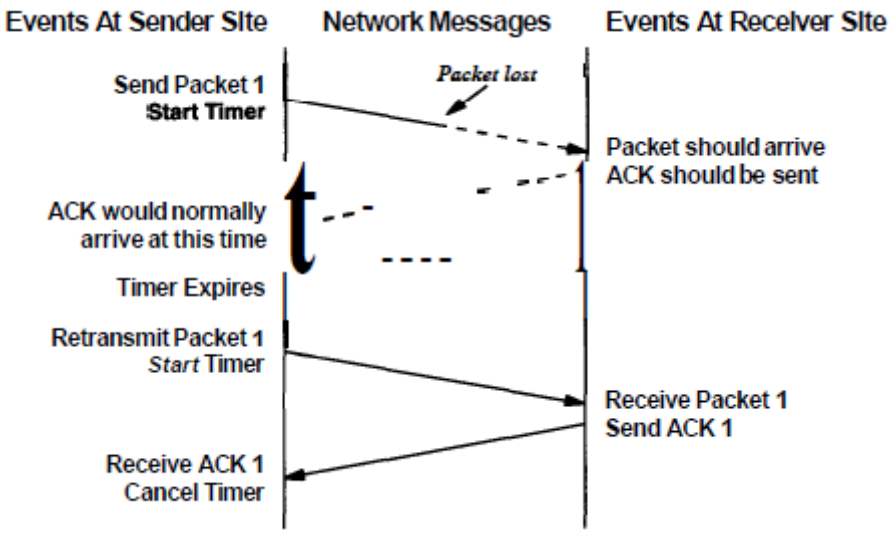


Figura 2.13. Sistema de reconocimiento por tiempo fuera existente en el protocolo TCP[10].

2.6.2.2 Comunicación y envío de datos

Ya entablada la comunicación a partir de estos mensajes existe otro mensaje de reconocimiento que se envía directamente desde el origen (PLC1) hacia el destino (PLC2), este mensaje tiene las mismas características que el reconocimiento del mensaje de sincronismo, sólo que en este caso existe un byte de tipo secuencia que se encuentra en el encabezado TCP. Este byte antes mencionado se inicializará dando como resultado el origen de la comunicación. Cabe recalcar que en esta trama no se encuentra activada la bandera de sincronismo solamente la de reconocimiento (ACK).

Partiendo de este mensaje, se envía el primer paquete de comunicación. Este primer paquete tiene encendidas las banderas de reconocimiento y push, el dato que se espera enviar y el valor de secuencia en la trama, que para este caso es uno. Posteriormente aparece el reconocimiento de este, siguiendo los lineamientos del protocolo TCP; ver imagen 2.10., enviado desde el destinatario hacia el origen, el cual no tiene datos, sino solamente activa la bandera de reconocimiento y ésta se encuentra en un valor de 2.

Partiendo de lo anteriormente mencionado, se puede concluir que los mensajes de envío de información siempre tienen encendidas las banderas de push y reconocimiento. Siendo importante, que en este caso la secuencia de datos indicará cuántas veces se ha enviado la información desde el inicio de la comunicación. Las tramas de reconocimiento de estos paquetes se envían sin ningún dato más allá de los encabezados, en los que se encuentra el mismo valor de secuencia, pero solamente se encuentra activada la bandera de

reconocimiento. Todo este proceso puede verse organizado y ejemplificado en la figura 2.14.

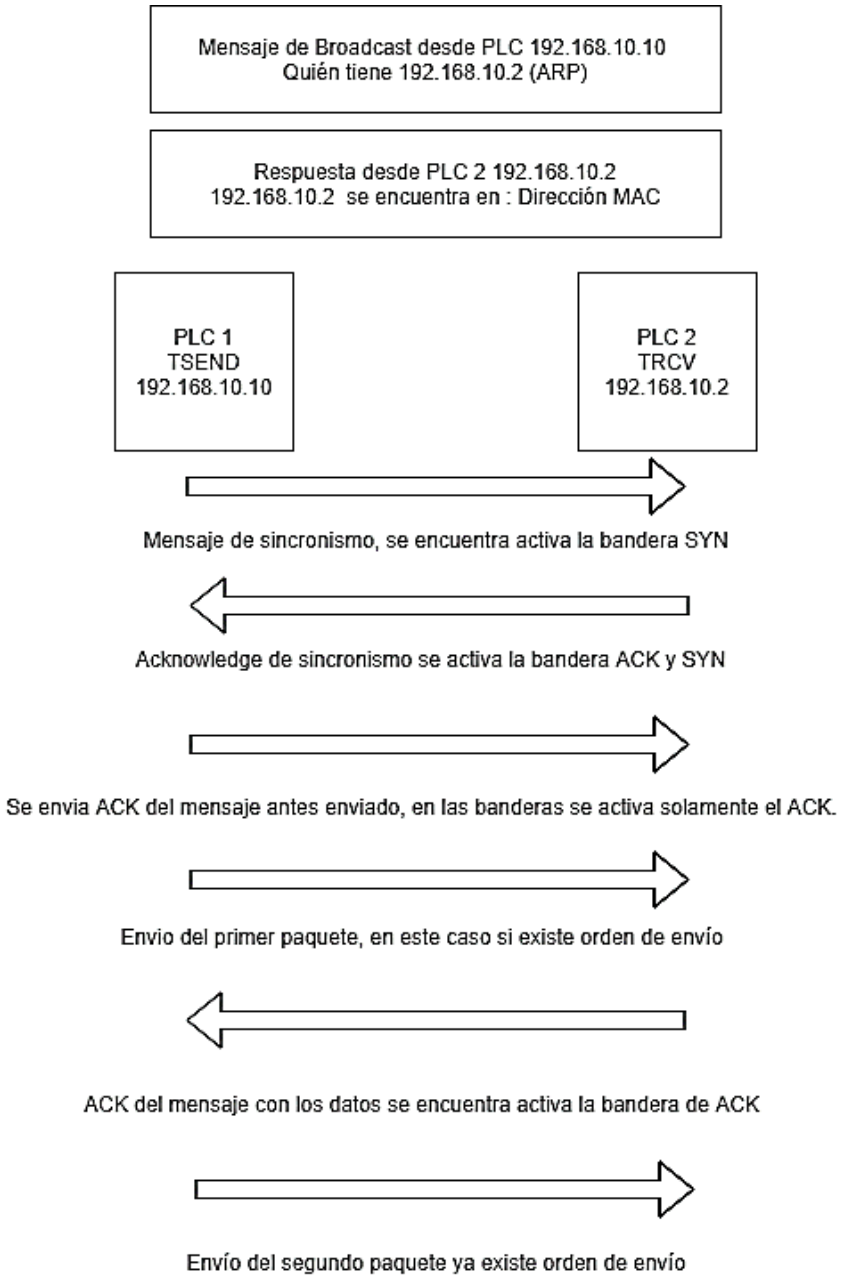


Figura 2.14 Diagrama representativo de inicio de comunicación entre 2 PLC para Profinet.

2.6.3 RECONSTRUCCIÓN DEL ENTRAMADO

El objetivo del proyecto de titulación consiste en el manejo de tramas Profinet de tipo entero, flotante y booleano. La diferencia entre estas tramas en el protocolo Profinet,

solamente es la configuración de la longitud de la trama y la configuración de los bytes según las características del dato.

La reconstrucción del entramado comprende el análisis del protocolo TCP y la configuración requerida para hacer posible él envió de datos desde y hacia el PLC. El envió, la recepción de datos a partir del protocolo y las herramientas requeridas para el manejo de cada uno de estos, son diferentes.

2.6.3.1 Construcción del Entramado de Escritura

Se consideran las librerías de creación de sockets y la librería *InetAddress* para el manejo que las características del protocolo IP. La librería *InetAddress* está orientada al manejo de la información de red de la computadora y de variables que contengan esta información. La librería *sockets* permite la creación de sockets que ocupen puertos de comunicación en la computadora y sean viables para receptar información en la programación. Usando estas librerías se creó un programa para la escritura de un dato en el PLC. Es importante aclarar que para el desarrollo de estas pruebas no se implementó una interfaz de usuario, los datos eran enviados desde el código de programación.

Utilizando las librerías se configura una dirección IP, que es la dirección del PLC, se configura un tiempo fuera, que determina la cantidad de tiempo que debe demorarse el PLC en responder al mensaje antes de que se cancele la comunicación y se establece el puerto que se utilizará para enviar el mensaje. Al momento de la creación del socket, se le asigna a este la IP antes mencionada y el puerto que es el configurado en el PLC. A este socket ya creado se le da la instrucción de conectarse, y se le entrega el tiempo fuera, que determina la finalización de la conexión en caso de existir un fallo, y a su vez activa la excepción, que permite identificar cuál fue el problema en la conexión.

En el establecimiento del entorno propicio para el envío de datos, se configura una variable especial en la librería socket, esta variable es el *SocketAdress*; establece el puerto y la configuración IP a utilizarse. Esta variable se establece en un ServerSocket al que se le otorga la función de *connect*.

El envió de datos utiliza la subfunción, ya existente en las librerías, "*getOutputStream",* esta instrucción permite la escritura con la instrucción de *write(),* en que se selecciona la variable a transferir por la red LAN hacia el PLC. El dato enviado debe ser un vector de bytes, los que deben de estar configurados según las características del dato. El proceso completo del funcionamiento de la librería en la aplicación de prototipo se muestra en la figura 2.15.

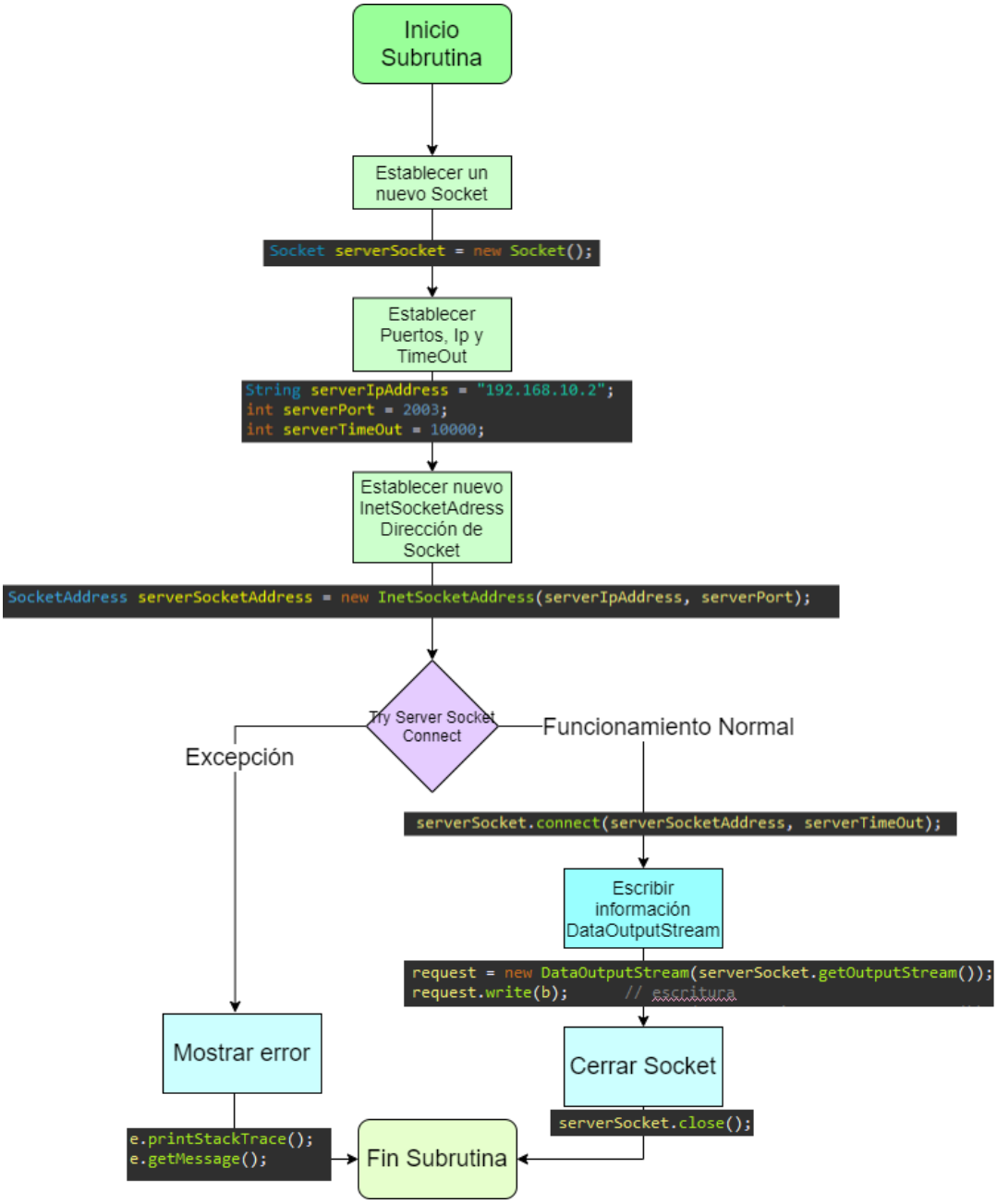


Figura 2.15. Diagrama de flujo para el uso de librerías para escritura de datos en el PLC.

Las características de la trama Profinet, permiten que el envió de múltiples datos a través de un paquete de bytes, sea posible. Siendo necesario a su vez modificar la programación del PLC y el método de manejo de bytes en JAVA. Las posibilidades de esta aplicación, de múltiples datos en un solo paquete, requieren que la programación del PLC y de Java; derive en un sistema muy complicado y conflictivo. Se puede considerar para aplicaciones específicas, ya que de esta forma los recursos ocupados en el PLC y el driver se reducen considerablemente.

Partiendo de lo antes mencionado se construye la trama. La figura 2.16 se presenta la trama generada desde una computadora hacia un PLC.

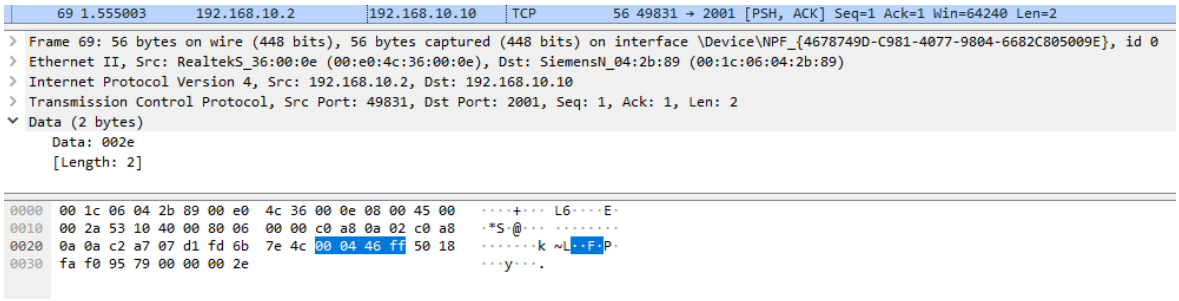


Figura 2.16. Entramado Profinet obtenido de Wireshark de PLC a PC 1 byte

2.6.3.2 Construcción del Entramado de Lectura

Se analiza el entramado de lectura. En este caso se considera que la comunicación no la inicia la computadora, sino el PLC, esto quiere decir que la computadora se encarga de receptar y aceptar esta comunicación de tal forma que el paquete puede llegar.

Se creó un programa de prueba para receptar la información enviada desde el PLC. Este utilizaba las mismas librerías antes mencionadas en el código de prueba del entramado de escritura.

La lógica del programa funciona solamente en su inicio, de la misma forma que en el entramado anterior, véase figura 2.17, ya que se debe crear un socket con características específicas, existiendo un cambio de datos en comparación con la configuración del entramado de escritura. Donde anteriormente existía un tiempo de ruptura de conexión, se encuentra un backlog que presenta el número de conexiones pendientes que la cola tendrá.

Se configura la IP del cliente, en este caso la IP de la computadora y el puerto que se ocupará en la conexión. Este puerto debe ser configurado, según los requerimientos de la comunicación, en el PLC.

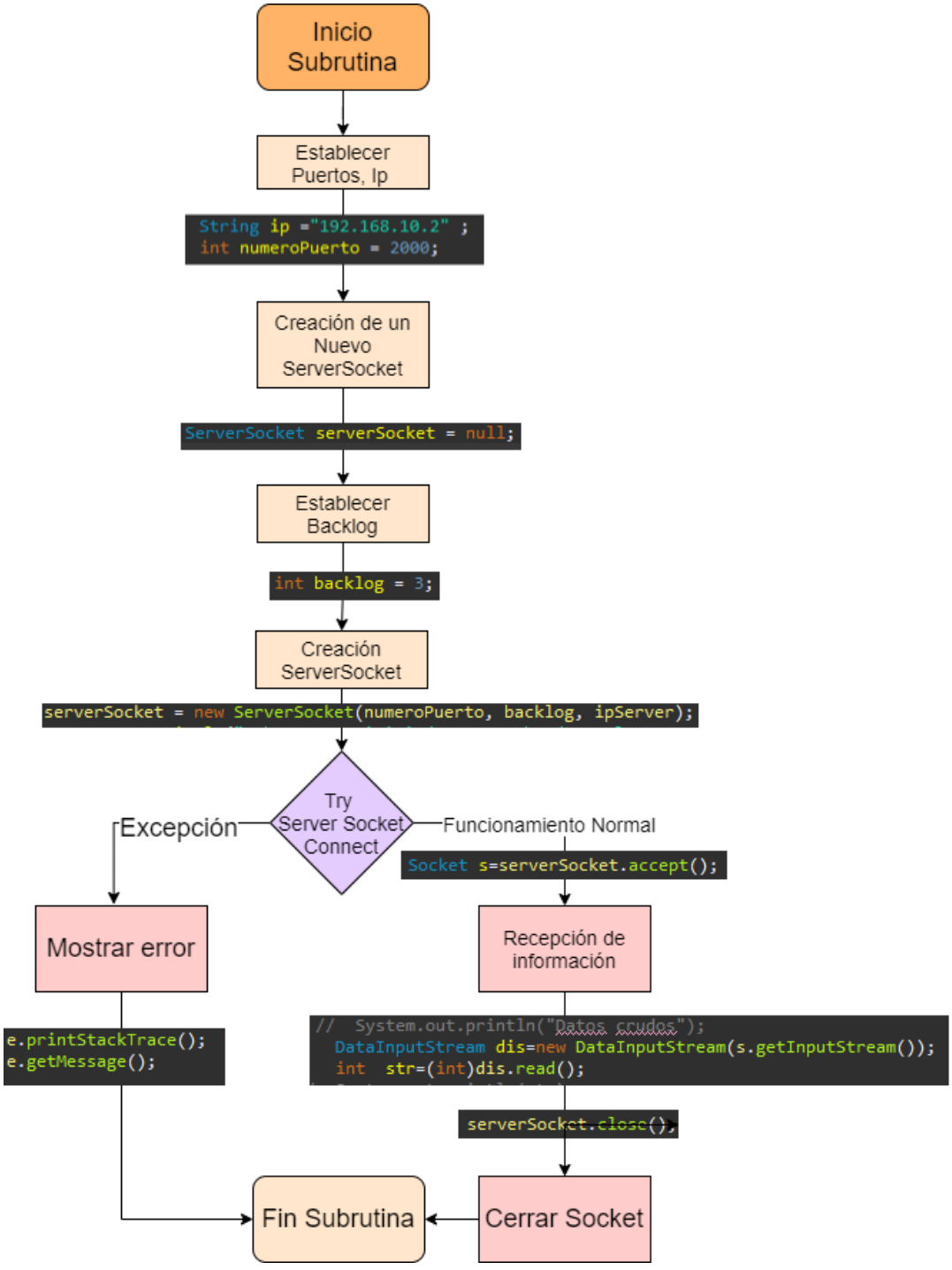


Figura 2.17. Diagrama de flujo para el uso de librerías para lectura de datos en el PLC

Posterior a la creación del "*ServerSocket",* se le ordena ejecutar la rutina de "*accept()"* que sirve para la recepción de datos de cualquier elemento que cumpla con las características de la configuración de trama TCP.

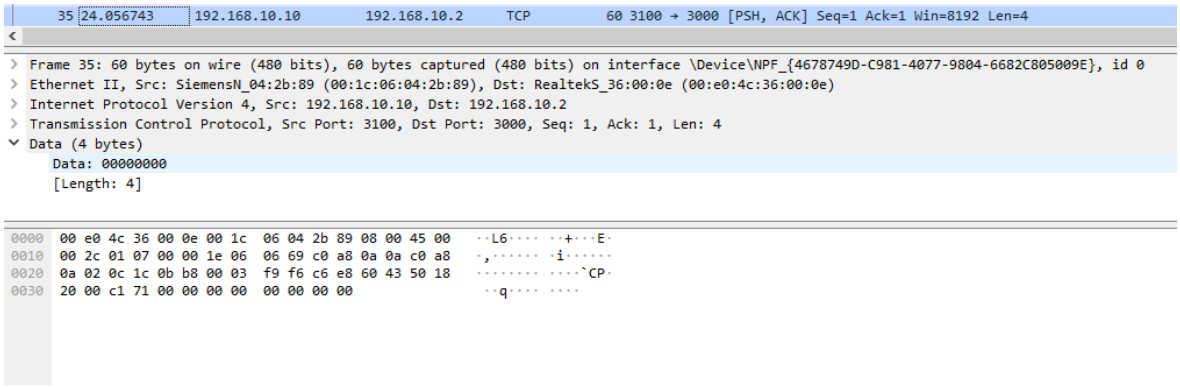


Figura 2.18. Entramado Profinet obtenido de Wireshark envío de paquete desde PLC a PC

La figura 2.18 se presenta un entramado de recepción de datos en profundidad desde un PLC hacia la computadora. La dirección MAC del elemento de recepción; es reconocida por el programa de Wireshark como un adaptador, demostrando así que el envío es desde un controlador hacia la computadora.

El driver está configurado de tal forma que pueda aceptar la comunicación desde el PLC. Esto quiere decir que el PLC es aquel que debe buscar los datos del receptor para enviar los registros, de tal forma que antes de que se pueda realizar la comunicación, el PLC se encuentra constantemente enviando un mensaje de broadcast en la red, que está buscando la información de la dirección MAC del receptor a partir de su dirección IP. La figura 2.19 se presenta este proceso.

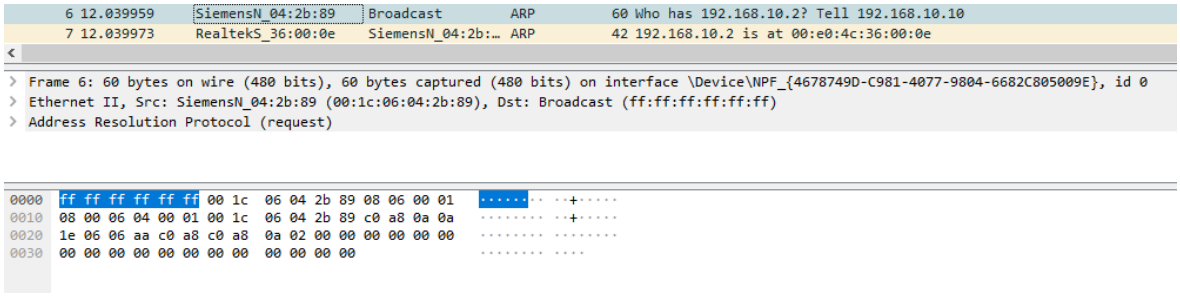


Figura 2.19. Mensaje de búsqueda tipo broadcast en la red

2.7. MANEJO DE DATOS Y MÉTODOS DE CONVERSIÓN

Como se menciona durante el marco teórico se deberá desarrollar 2 métodos para el manejo de cada uno de los bytes recibidos durante la trama Profinet. Esto se debe a que la organización de los bits recibidos en la red por sí solos no son suficientes para ser comprendidos ya sea por el operador o por la base de datos. Se requiere de métodos de

conversión de tal forma que, estas matrices de datos en bytes puedan ser convertidas en variables comprensibles y manejables en la programación.

El objetivo del driver es el manejo de datos enteros, flotantes y booleanos. Las características tanto de la trama de Profinet, así como del almacenamiento en MySQL, requieren que los datos booleanos sean almacenados en bloques de bytes, estos no requieren de una manipulación de datos para ser comprendidos por el operador; al contrario de los datos tipo flotante y entero.

2.7.1 MÉTODO DE MANEJO DE DATOS ENTEROS

Los datos enteros en Java se almacenan en bloques de 16 bits, o en 2 bytes. El caso de la trama Profinet, propone que estos bytes vienen separados. Teniendo en cuenta que el objetivo del driver es la capacidad de escribir y leer datos en un PLC, es requerido un método que sea capaz de realizar una conversión bidireccional entre un entero y sus componentes en bytes. A continuación, se presenta el pseudocódigo de la conversión desde un dato entero a sus componentes en bytes.

Método 1: Conversión de entero a bytes

1. DEFINE INT byte superior, byte inferior
2. IF valor entero < 256
3. byte inferior → valor entero adaptado a byte
4. byte superior → 0
5. ELSE
6. byte superior → Valor entero dividido para 256
7. byte inferior → Valor entero menos byte superior por 256
8. END IF

El método mostrado ocupa las características de las variables de tipo byte en el lenguaje Java, eliminando los valores decimales en una división con un residuo. A su vez, se requiere de las herramientas de conversión de variables que existen en JAVA, estas permiten el uso de un dato tipo entero como uno tipo byte, contando únicamente los datos correspondientes al último byte del valor entero.

Para la conversión en sentido inverso se requiere de otro método, que ya se encuentra implementado en las funciones internas de la librería de creación de sockets; ya que al momento de obtener el dato del entramado Profinet, la librería recupera la información por conjuntos de 16 bits.

2.7.2 MÉTODO DE MANEJO DE DATOS FLOTANTES

Se requiere de un método que sea capaz de manejar datos flotantes. En este caso, el método es más complejo que en el método de datos enteros. Para la obtención del valor flotante se requiere de analizar los datos recibidos por bits ya que los mismos se encuentran en formato de precisión simple (Véase marco teórico manejo de datos Profinet).

2.7.2.1 Método de Transformación de un Dato Flotante en Array de Bits

Como se explica en el marco teórico en el manejo de datos Profinet literal 1.3.4, un dato flotante de precisión simple se encuentra conformado por cuatro bytes, en los cuales existe una mantisa, un exponente y un byte específico para el signo correspondiente. Para el manejo de este dato se utiliza una librería ya creada en Java, esta requiere que el dato se encuentre descompuesto y organizado de forma específica en un vector de bytes. A continuación, se muestra el pseudocódigo utilizado para la conversión de un dato flotante en un vector de bytes.

Método 2: Conversión De Flotante A Array De Bytes

1. DEFINE BYTE ARRAY datos []
2. DEFINE FLOAT valor flotante
3. DEFINE INT auxiliar
4. Auxiliar → Float.floatToIntBits(valor flotante) // función floatToIntBits
5. Datos [] → (byte) (auxiliar >> 24), (byte) (auxiliar >> 16), (byte) (auxiliar >> 8), (byte) (auxiliar) // se desplaza el resultado obtenido a partir de la función

Partiendo de los requerimientos que se tiene para la obtención del vector de bytes, se puede suponer que el método interno de la función *"floatToIntBits"* es realizar un desplazamiento de cada uno de los bits del valor flotante, tomar el valor del bit que determina el signo, separar la mantisa del exponente y calcular el valor final.

2.7.2.2 Método De Transformación De Array De Bits En Un Dato Flotante

Para convertir la información obtenida desde el PLC y traducirla de tal forma que esta pueda ser trabajada por JAVA; para consecuentemente ser ingresada por la base de datos en MySQL. Se maneja un método diferente, ya que la subrutina de conversión existente, *"intBitsToFloat"*, tiene otros parámetros de entrada para la obtención del valor flotante. A continuación, se muestra el pseudocódigo utilizado para la conversión de un vector de bytes en un dato flotante.

Método 3: Conversión de Array de Bytes a Flotante

1. DEFINE BYTE ARRAY datos []
2. DEFINE FLOAT valor flotante
3. DEFINE INT auxiliar
4. datos [] → {(byte) primer byte obtenido TCP, (byte) segundo byte obtenido TCP, (byte) tercer byte obtenido TCP, (byte) cuarto byte obtenido TCP}

   // creación del array de bytes con los datos obtenidos de la comunicación
5. auxiliar [] → {datos [0]<<24 | (datos[1]&0xFF) << 16 | (datos[2] & 0xFF) <<8 | datos[3]&0xFF}

   // los datos del array son modificados según se muestra para crear el array requerido por la subrutina
6. Valor flotante → Float.intBitsToFloat(auxiliar)

Para preparar al array de bytes, de tal forma, que este sea manejable por la subrutina y se obtenga el resultado deseado, se mueve cada una de las partes del array original según se muestra en el pseudocódigo del método 3. La operación de AND mostrada se utiliza para cerciorar al sistema de que los datos son de 16 bits cada uno ya que la obtención de datos se hace a través de enteros.

2.8. GESTIÓN DE BASE DE DATOS

El objetivo de la base de datos es conectar el driver con la interfaz del operador, de tal forma que el driver tiene la tarea de escribir o leer los datos según se requiera. Para realizar la tarea de conexión con la base de datos el driver debe ser capaz de utilizar las librerías de conexión MySQL, que son *"java.sql"* y *"java.sql.preparedstatement*". Se ocupa un conector/J para MySQL, este es necesario para comunicar MySQL con JAVA.

2.8.1 REQUERIMIENTOS

En la configuración requerida para el funcionamiento del driver, en un entorno de pruebas, se propone la creación de 2 tablas en una base de datos; en estas se requiere el establecimiento de una KeyID para el funcionamiento adecuado del driver. En estas deben crearse 3 columnas para los diferentes tipos de datos: entero, flotante y booleano. Para MySQL estos son: Int, Float y Tinyint, respectivamente. El prototipo creado es capaz de modificar la base MySQL según los requerimientos de la aplicación, pero se recomienda la

creación de la tabla según las especificaciones propuestas para reducir los tiempos de escritura de datos.

2.8.1.1 Conexión MySQL

Para realizar la conexión a la base de datos se debe conocer la dirección del conector JDBC, o conector/J para MySQL. Se requiere también de la dirección URL de la dirección de los datos, esta se encuentra fácilmente en la configuración de la base de datos. Se requiere a su vez la configuración adecuada del usuario y la contraseña, ya que para el acceso a la información se requiere de un proceso de autenticación en la base de datos.

Para la conexión MySQL se genera una clase para su realización. Esta tiene un funcionamiento similar a las clases generadas para la comunicación con la base de datos, ya que tienen las sentencias de *try* y *catch*; en caso de existir un problema con la conexión la sentencia de excepción permite identificarlo y dejar la comunicación hasta que el problema sea resuelto. Con los requerimientos antes mencionados, como la dirección de datos, así como el usuario y la contraseña; se permite el acceso a la base de datos en la subrutina. A continuación, se presenta el pseudocódigo de la clase para la conexión con la base de datos.

Método 4: Conexión a MySQL

1. DEFINE CONNECTION connection
   // gracias a las librerías se pueden crear estas variables para establecer la configuración de las conexiones
2. TRY
3. Class.forName (“com.mysql.jdbc.Driver”)
   // dirección del conector JDBC
4. DEFINE STRING url, user, password
5. url → "jdbc:mysql://localhost:3306/tesis"+"?useTimezone=true&serverTimezone=UTC&useSSL=false"
   // dirección de la base de datos
6. User → root
7. Password → password
   // Usuario y contraseña
8. DriverManager.getConnection (url,user,password)
   // Intento de conectar con la base de datos
9. CATCH
10. Imprimir Errores

Como se puede observar, durante el método de conexión con la base de datos, el driver utiliza una información predeterminada durante la programación, la cual le permite acceder a la base de datos gracias al driver JDBC. Se ocupa el terminal para observar si la conexión con la base de datos es exitosa o no, pero este método se ocupa solamente durante el desarrollo del driver, ya que no es viable para el momento de la aplicación del resultado final, el terminal no es un elemento de uso común en una interfaz de usuario. En las pantallas de aplicación, en el caso de existir una falla en la conexión con la base de datos se muestra una alerta, para informar al usuario de la existencia del error.

2.8.2 ESCRITURA Y LECTURA EN BASE DE DATOS

Para la escritura y lectura en base de datos se utilizan estamentos específicos, es necesario conocer cómo trabajar con estos estamentos pero en la programación de Java. Gracias al driver JDBC es posible, mediante un código específico, trabajar con el lenguaje MySQL, aunque es importante recalcar que, la programación no es exactamente la misma, ya que al existir variables que dependen de situaciones ajenas al driver; estas posibles variaciones deben ser consideradas también y esto requiere modificaciones en la programación.

2.8.2.1 Estamentos Utilizados en Escritura

A continuación, se presentan los estamentos que se usaron durante la programación para escribir en la base de datos, otros estamentos fueron utilizados a la par para obtener información desde la base de datos por lo cual su clasificación se encuentra en los estamentos de lectura que se revisarán más adelante, véase literal 2.9 Funciones principales. A su vez, se presenta las variaciones en programación que tienen estos estamentos al ser utilizados en Java.

- UPDATE: "UPDATE `"*+nombretabla+*"` SET `"*+nombrecolumn+*"` = """ WHERE (`"*+lecturaidcol+*"` = '"*+id+*"')"

El uso del estamento *update* es para modificar los registros de una tabla en la base de datos, este es el más utilizado en la programación del driver. Ende la representación de la aplicación del estamento existen variables que deben ser insertadas. A continuación, se explica de dónde provienen estas variables.

- nombretabla: como su nombre lo indica aquí se inserta el nombre de la tabla en la base de datos.

- nombrecolumn = Nombre de la columna en la base de datos
- lecturaidcol = Nombre de la columna donde se encuentran los Key ID (revisar marco teórico)
- id= Número esperado en la columna del Key ID
- INSERT: "INSERT INTO nombtabla VALUES (?,?,?,?,?)"

El uso del estamento *insert* es para crear una nueva fila en la base de datos. Este estamento no se utiliza de forma continua en la programación del driver, sino que su uso está relacionado a la etapa de desarrollo del driver. Se utiliza la función *set*, ya que esta entregará la información acerca de cuáles son las variables y su tipo antes de ser incluidas en el estamento.

- Nombtabla: es el nombre de la tabla en la base de datos
- ?,?: presentan a las variables a insertar en la base de datos y que como se mencionó utilizan la función de *set.* El número de signos determinará el número de variables a ser insertadas en la base de datos, en caso de no coincidir el número de signos con el número de columnas en la base de datos se presenta un error. Siendo esta la razón principal por la cual se utiliza el estamento de *update.*

2.8.2.2 Estamentos Utilizados en Lectura

Se ocuparon otras funciones de MySQL para la obtención de registros desde la base de datos. A continuación, se presentan los estamentos utilizados para la obtención de información desde la base de datos.

- SELECT: "SELECT "+columna+" FROM "+nomtabla”

Este estamento nos permite seleccionar específicamente desde que columna y desde que tabla se requiere los datos. El método de recuperación de información utiliza la función “*next()*” para acceder a cada uno de los elementos en la columna, permitiendo el uso del dato en la última fila, siendo este, el que se requiere. Se presentan las variables utilizadas en el estamento.

- Columna: es el nombre de la columna de donde se desea sacar la información.
- Nombtabla: Nombre de la tabla de donde se desea sacar la información.
- ALTER TABLE: "ALTER TABLE `"+nombretabla+"`ADD COLUMN "+ nombrecolum”

Este estamento se ocupa para la modificación del número de columnas existentes en una tabla, en este caso se añade una columna con un nombre específico, que es aquel que ingrese el usuario. A continuación, se presentan las variables utilizadas en el estamento.

- Nombretabla: nombre de la tabla en la base de datos.
- Nombrecolum: Nombre de la columna a añadir en la base de datos.

Con el estamento "*select"* se puede trabajar con las librerías de obtención de información, estas están orientadas a obtener datos de configuración de la tabla como el número de filas, el número de columnas o el nombre de estas. Para la obtención de estos datos se ocupa la subrutina de: "*getmetaData()*". Esta nos permite generar una variable especial donde se almacena la información de la tabla.

- getColumnCount(); Obtención del número de columnas existentes en la tabla.
- getColumnName(a); Obtención del nombre de la columna a en la tabla.
- getRow(); Obtención del número de filas existentes en la tabla.

Estas funciones se ocupan constantemente al momento de inicializar el driver, se usan sobre todo en el driver de escritura en la base de datos, ya que en el caso de existir algún error en la programación MySQL, o en el número de columnas o filas existentes se presentan errores de comunicación.

2.8.2.3 Método de Escritura y Lectura Básico

Se realizaron clases de prueba para la comunicación con la base de datos. A continuación, se muestra el pseudocódigo utilizado para el método de escritura y lectura básico.

Método 5: Conexión Básica Base de Datos

1. DEFINE PREPARED STATEMENT pps
   // se determina la variable para conexión. PPS es la variable original en la programación de JAVA.
2. TRY
3. pps = MySQLConnection
   // se realiza la conexión con la clase anterior
4. pps.setInt
   // solo se ocupa en caso de existir una variable a escribir como el caso de INSERT
5. get(Int/Float/Byte)

// Solo se ocupa para la obtención de datos de la base de datos se usa con SELECT

6. pps. Execute update
7. CATCH
8. Mostrar errores

2.9. FUNCIONES PRINCIPALES

Todos los métodos anteriormente ejemplificados por pseudocódigo deben ser agrupados. Hasta ahora se ha visto la implementación de métodos individuales con aplicaciones específicas, la mayoría de estos únicamente son aplicados de la forma en la que se ejemplificó en los pseudocódigos de los métodos anteriores, como clases para prueba del sistema en el desarrollo.

2.9.1 MÉTODO DE LECTURA BASE DE DATOS Y ESCRITURA EN PLC

Como su nombre lo indica este método se ocupa de enviar registros desde la base de datos hacia el PLC. Por lo tanto, debe ser capaz de escribir en el controlador y leer datos desde la base de datos. Debido a las características del método de envío de información en el protocolo Profinet, en la aplicación del driver existen 3 diferentes métodos de este tipo.

Este método ocupa la obtención de datos desde MySQL y el método de escritura en el PLC. Para el método de obtención de datos desde la base de datos, se requiere información extra, como por ejemplo el Key ID, que se obtiene con el método de recepción de información de una tabla. La aplicación completa de todas las funciones y métodos explicados se desarrollará en el funcionamiento de las pantallas de configuración y usuario.

En la figura 2.20, se muestran el diagrama de flujo del método de lectura de la base de datos y escritura en PLC. Se puede observar la existencia de 2 lazos de try catch, por lo tanto, existen excepciones que se presentan cuando ocurren errores. Existe a su vez una tercera excepción no mostrada dentro diagrama de flujo, esta es exclusiva del manejo de MySQL, por lo que se omite, y es paralela a la primera excepción mostrada en el diagrama.

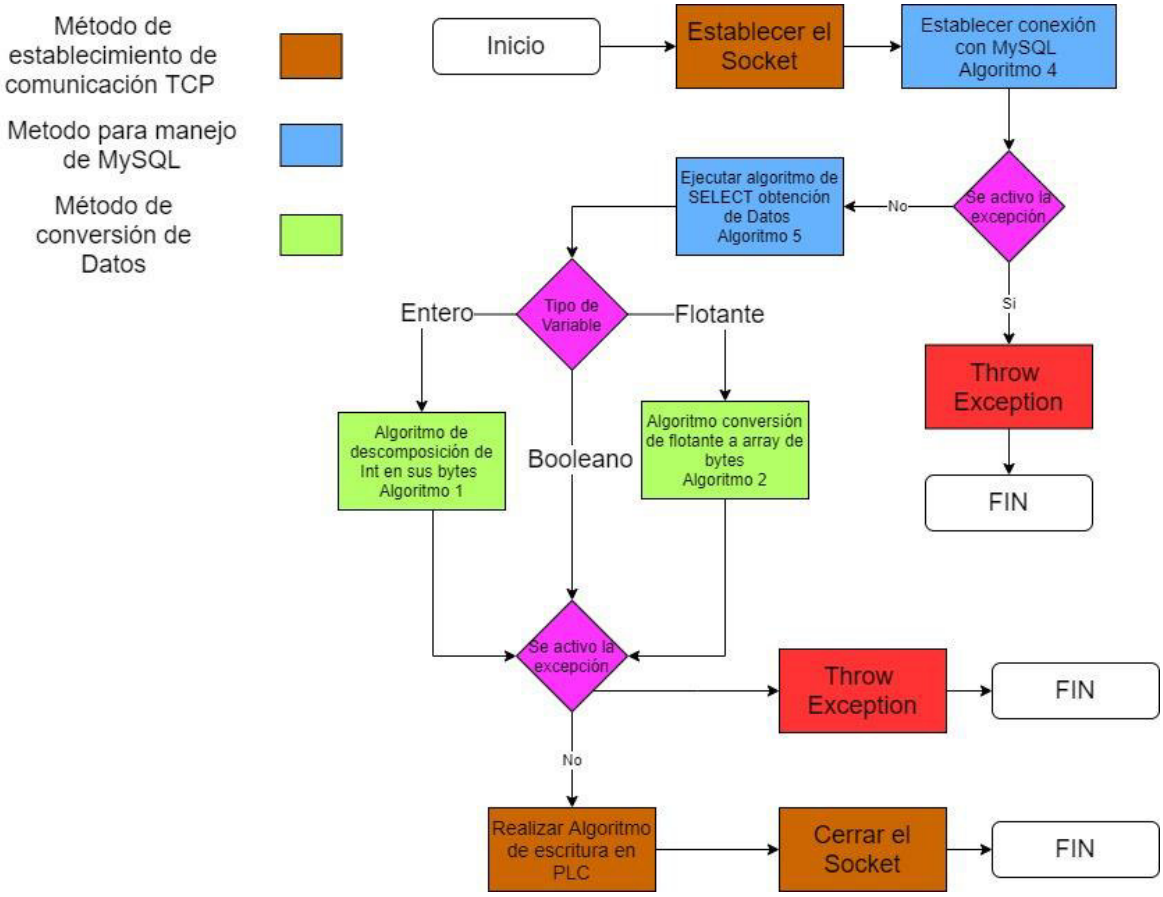


Figura 2.20. Diagrama de flujo del método de escritura en PLC lectura base de datos.

En la figura se tiene una leyenda para explicar los tipos de métodos utilizados durante el diagrama de flujo. Se ocupan los métodos para el manejo de datos en las bases de datos así como las librerías y los métodos utilizados en la subrutina de escritura de datos de forma directa en el PLC.

2.9.2 MÉTODO DE LECTURA DE PLC Y ESCRITURA EN BASE DE DATOS

El objetivo del método de lectura PLC y escritura en base de datos es obtener la información desde el PLC, traducirla dependiendo del tipo de variable que sea y guardarla en la base de datos. Este método cuando se encuentra aplicado en la programación final tiene la característica de que requiere mucha información obtenida de la base de datos, como es el caso del número de filas, número de columnas, o nombre de la columna.

En el diagrama mostrado en la figura 2.21 se maneja una leyenda de colores para diferenciar los métodos de comunicación TCP, MySQL y conversión de datos utilizados.

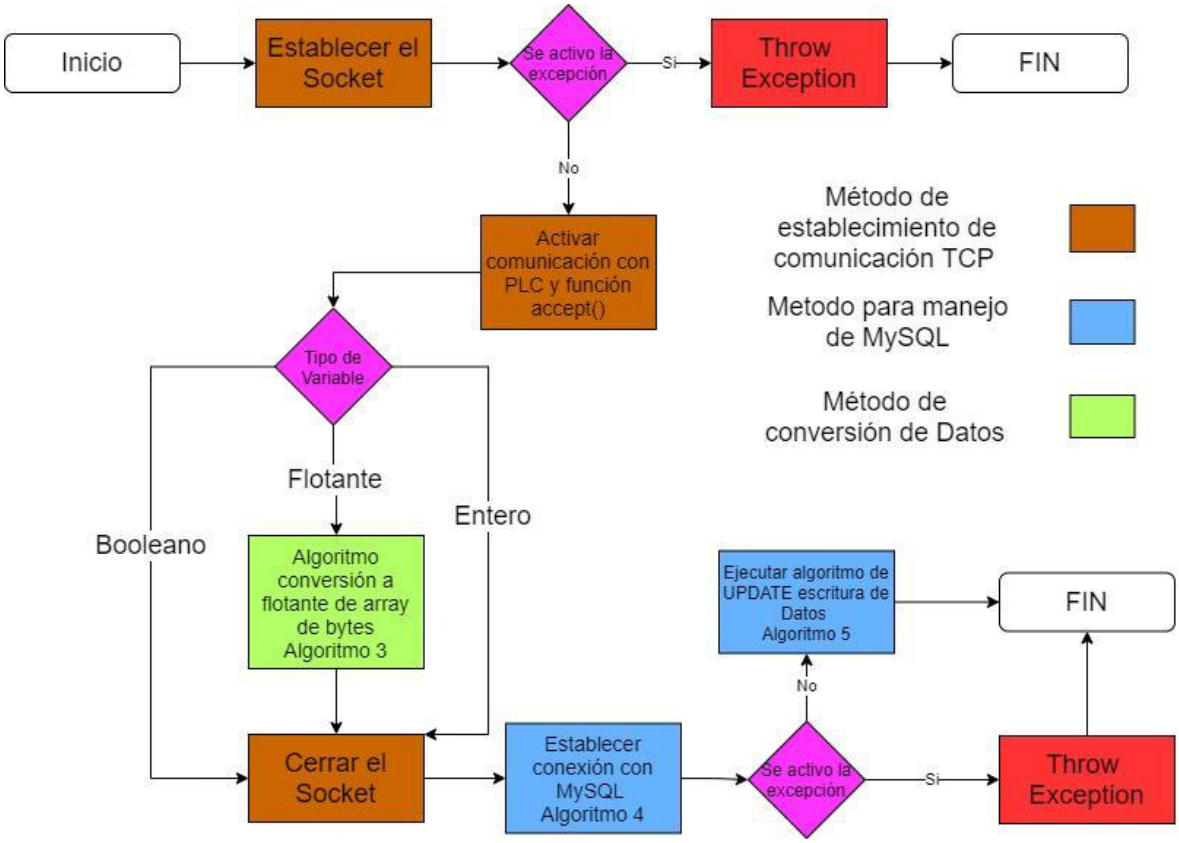


Figura 2.21. Diagrama de flujo del método de escritura en base de datos lectura PLC

2.10. INTERFAZ GRÁFICA

La interfaz completa está compuesta por un menú principal, dos pantallas de prueba, dos pantallas de configuración y dos pantallas de operador. Es importante aclarar que, existe una limitación de la cantidad de datos a enviar o recibir desde el driver, por cada una de las pantallas abiertas. Esta limitación se elimina con la apertura de más pantallas, ya que la programación permite la creación y uso de múltiples pantallas. Es necesario tener en cuenta que, al momento de ocupar varias pantallas del driver para enviar o recibir datos constantemente, la configuración tanto de la base de datos, así como del uso de puertos, en el PLC como en la computadora es más conflictiva, ya que los requerimientos para realizar la comunicación en la computadora así como en el PLC se incrementan.

En caso de que requerir utilizar el driver en una aplicación que requiera más de una pantalla se recomienda tener conocimiento previo del funcionamiento del driver mediante este documento y realizar pruebas independientes con cada una de las pantallas a utilizar. No se recomienda para este prototipo, pero es viable con una optimización de canales de comunicación propuesta para una mejora del driver en su largo plazo.

2.10.1 DISEÑO PANTALLA CONFIGURACIÓN DE LECTURA Y ESCRITURA

El diseño de las pantallas de configuración tanto de lectura y escritura deben ser amigables con el usuario, de tal forma que sea comprensible el funcionamiento de esta. Teniendo en cuenta que se desarrolla un manual de usuario para entender del funcionamiento general y requerimientos para configuración.

Como se mencionó antes, las pantallas de escritura y de lectura se encuentran limitadas por una cantidad de datos, esto se debe principalmente a la complejidad y recursos continuos que requiere el driver para funcionar. Es importante aclarar que, esto se realizó en favor de una interfaz más amigable con el usuario, ya que en caso de aumentar más datos a analizar, la densidad de elementos aumenta considerablemente. El objetivo de las pantallas es ocupar un sistema que permita monitorear el estado del driver mientras se realizan otras actividades, o se ocupan más pantallas del prototipo. La organización del espacio está orientada a la fácil ubicación de los diferentes campos de configuración sin sobrecargar al operador.

En todos los casos se propuso el uso de una interfaz tipo tabla en la que se determinan columnas las cuales entregan las características del dato y se determinan filas las que diferencian los valores en las columnas entre dato y dato.

2.10.1.1 Pantalla Escritura Configuración

La pantalla de configuración de escritura es la antesala a la pantalla de operador, que se considera como el resultado final del proyecto. En esta pantalla se configuran los requerimientos y la información necesaria para recibir la información desde la base de datos y posteriormente escribirla en el PLC. Se presenta la pantalla en la figura 2.22.

Figura 2.22. Ventana de configuración del driver interfaz de escritura en el PLC.

Como se mencionó anteriormente, la configuración de los elementos en la pantalla tiene una orientación de tipo tabla. La parte superior contiene el título de la pantalla, así como el estado del servidor y un botón de retorno. La parte inferior de la pantalla contiene los controles de encendido y apagado del driver, las configuraciones de la IP del PLC, el número de datos, el tiempo de bucle y el nombre de la tabla a ocupar en la base de datos. Esto se puede observar en la figura 2.22.

Todos estos datos son requeridos tanto, para la obtención de datos desde MySQL, así como para la creación de las tramas de Profinet. Las configuraciones añadidas son el tiempo de bucle, que determina la repetición del proceso de escritura en el PLC desde la base de datos, así como la cantidad de datos, la cual sirve para determinar la cantidad de celdas a habilitar en la tabla y los métodos internos a habilitar en la programación.

A continuación, se muestra una tabla detallando cada una de las partes de la ventana y el funcionamiento de estas.

Tabla 2.1. Elementos Dentro De La Pantalla De Configuración Escritura

Cuadros de Texto o Selección	
Dato (i)	Permite seleccionar si el dato se envía o no.
Puerto	Configuración del puerto a enviar la información

Tipo de Dato	Selecciona el tipo de dato a enviar
Nombre de Columna	Nombre de la columna de donde se saca la información de MySQL
Nombre de Tabla	Nombre de la tabla de donde se saca la información de MySQL
Cantidad de Datos	Selección de cantidad de datos a enviar. Habilita configuración
Tiempo de Loop	Tiempo de repetición de escritura de la información.
Botones	
Atrás	Cierra la venta y muestra de nuevo la pantalla de menú principal
Ping	Permite verificar si existe conexión con la IP del PLC
Pantalla de Usuario	Termina la configuración y abre la pantalla de operador
Encendido	Se enciende el driver, empieza la comunicación con la base de datos y el PLC.
Apagado	Después que se termine el bucle en proceso, se para la comunicación.
Indicadores	
Estado del Servidor	Estado del driver

2.10.1.2 Pantalla Lectura Configuración

La pantalla de configuración de lectura precede a la pantalla de operador, estableciendo aquí las configuraciones requeridas para la puesta en marcha del driver. Esta tiene la misma disposición de los elementos que la pantalla de configuración de escritura, ambas se encuentran organizadas en un formato tipo tabla para facilidad del entendimiento de la información. En la figura 2.23 se observa la configuración mencionada.

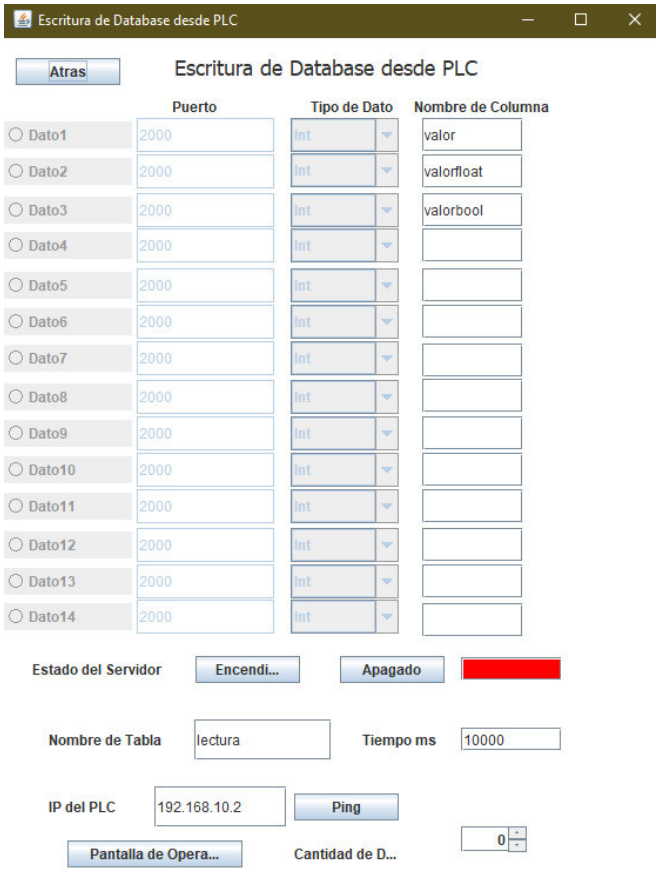


Figura 2.23. Ventana de configuración del driver interfaz de lectura en el PLC.

Se observan ciertas similitudes con la pantalla de escritura en el PLC, como el requerimiento de la IP del controlador, número de datos, nombre de la tabla, tiempo de bucle, etc. Como ya se ha dicho anteriormente, toda esta información es requerida para desarrollar de forma óptima las sentencias que conectan al driver con la base de datos y el PLC.

Se muestra el uso de cada una de las partes de la pantalla en la tabla 2.2 Elementos en la Pantalla de Configuración Lectura.

Tabla 2.2. Elementos en la Pantalla de Configuración Lectura

Cuadros de Texto o Selección	
Dato (i)	Permite seleccionar si el dato se envía o no.
Puerto	Configuración del puerto a enviar la información
Tipo de Dato	Selecciona el tipo de dato a enviar
Nombre de Columna	Nombre de la columna donde se escribe la información de MySQL. Nombre con que se creará la columna.

Nombre de Tabla	Nombre de la tabla donde se escribe la información de MySQL
Cantidad de Datos	Selección de cantidad de datos a recibir. Habilita configuración y determina cantidad de columnas en la tabla.
Tiempo de Loop	Tiempo de repetición de recepción de la información.
Botones	
Atrás	Cierra la venta y muestra de nuevo la pantalla de menú principal
Ping	Permite verificar si existe conexión con la IP del PLC
Pantalla de Usuario	Termina la configuración y abre la pantalla de operador
Encendido	Se enciende el driver, empieza la comunicación con la base de datos y el PLC.
Apagado	Después que se termine el bucle en proceso, se para la comunicación.
Indicadores	
Estado del Servidor	Estado del driver

2.10.2 PANTALLAS DE PRUEBA

Una herramienta propuesta en el driver son las pantallas de prueba. Estas sirven para verificar cada una de las conexiones desarrolladas en el driver de forma individual. Las pantallas de prueba no permiten el uso de las otras pantallas, ya que al momento de utilizar la comunicación se ocupan los mismos recursos y solo se puede realizar una conexión a la vez. Las pantallas de prueba mandan la información desde el driver o reciben información desde el PLC, debido a que utilizan la información ingresada en el driver; no tienen una conexión con la base de datos.

2.10.2.1 Pantalla de Prueba Lectura

La pantalla de prueba de lectura tiene como objetivo únicamente ingresar los puertos para los cuales se programó el envío de la información desde el PLC. De tal forma que se pueda leer la información directamente en el driver, comprobar que está se encuentra correctamente configurada y que no existen problemas en la conexión. En caso de no poderse realizar la comunicación con el PLC desde la pantalla de prueba, se debe revisar las conexiones en red, así como las características del adaptador de red de la computadora.

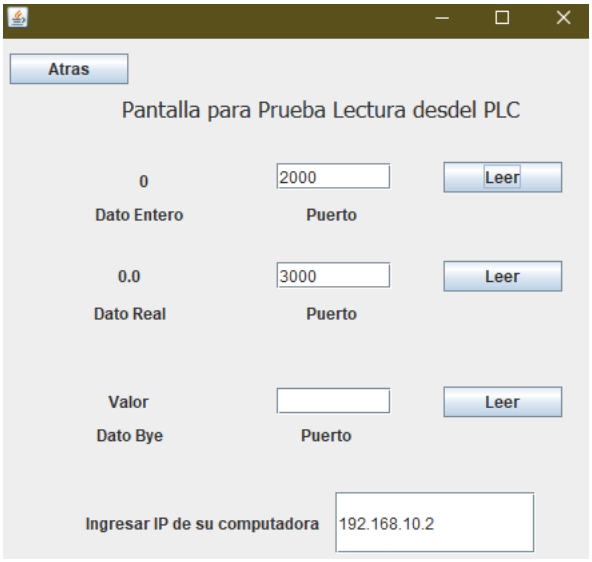


Figura 2.24. Ventana de prueba de conexión de lectura del PLC.

La figura 2.24 se presenta que la ventana de prueba de lectura tiene una distribución de los elementos bastante simple. Únicamente se encuentran 3 posibilidades de lectura las cuales se realizan una sola vez, sin la existencia de ningún tipo de bucle. La pantalla está desarrollada para hacer un análisis del estado de las conexiones con el PLC una por una.

2.10.2.2 Pantalla de Prueba Escritura

La pantalla de prueba de escritura tiene el mismo objetivo que la pantalla de prueba de lectura, el método se basa en la escritura de un dato en el PLC. El método en el driver genera los datos que requieren ser escritos en el PLC dependiendo del proceso de la planta. Para esta conexión el dato debería generarse en la interfaz del operador orientada a su uso específico en la planta, pero debido a que esto no es posible; se ingresa un valor en la ventana de prueba. La pantalla de prueba sirve para verificar la conexión del driver con el PLC, y la base de datos no tiene ninguna influencia en este análisis.

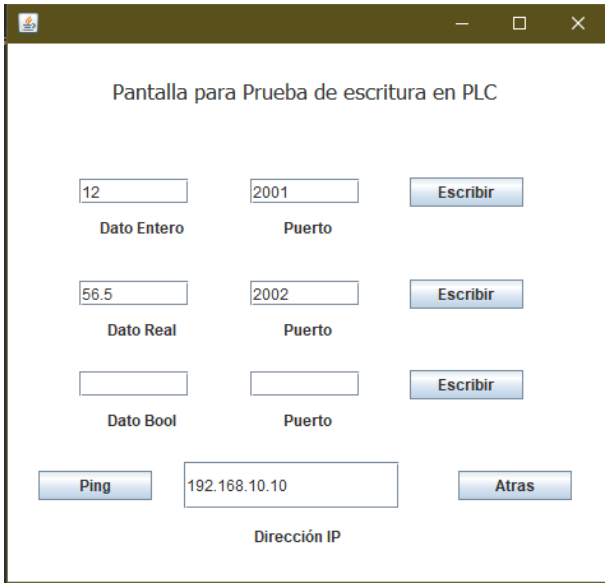


Figura 2.25. Ventana de prueba de conexión de escritura del PLC.

En la figura 2.25 se puede observar que tiene una distribución similar en muchos aspectos a la pantalla de prueba de lectura, la diferencia principal radica en la existencia de un cuadro de texto para el ingreso del dato a ser enviado, así como el requerimiento de la IP, que en este caso es del PLC. Existe el botón de ping que realiza un intento de conexión en la red con la IP ingresada, esto no significa que la conexión de los puertos y configuración sea correcta, solamente significa que la dirección IP del PLC es alcanzable en la red.

2.10.3 PANTALLAS DE OPERADOR Y MÉTODO DE FUNCIONAMIENTO

Estas pantallas son solamente accesibles después de haber realizado el establecimiento de información necesaria en las pantallas de configuración respectivas, su funcionamiento depende exclusivamente de la correcta puesta en marcha, que vaya acordé con la correcta programación realizada en los controladores. La programación aplicada a estas pantallas determina el flujo de datos y lógica de trabajo de todo el sistema, este proceso se explica de forma simplificada en esta parte del proyecto.

2.10.3.1 Pantalla De Escritura

La pantalla de operador de escritura tiene la misma organización que su pantalla de configuración, exceptuando que, el usuario únicamente puede encender y apagar el driver.

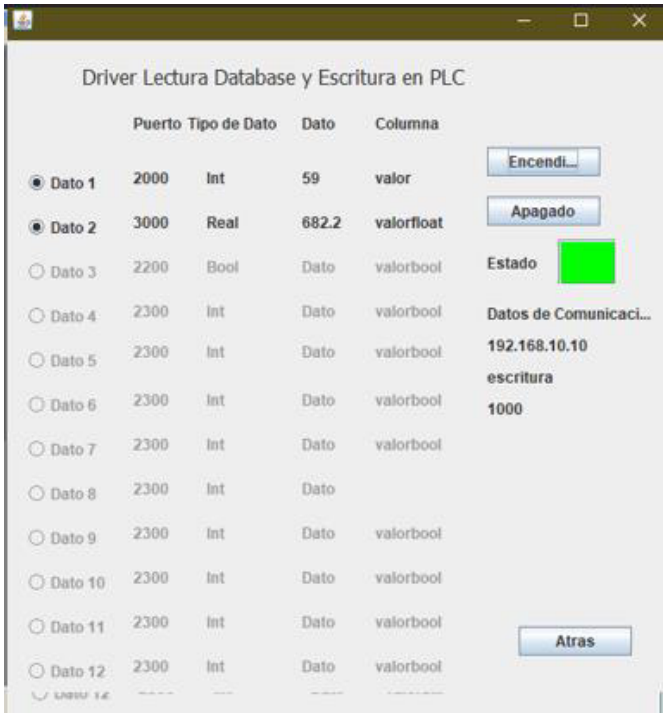


Figura 2.26. Ventana de Operador de conexión de escritura del PLC.

Para el momento de la toma de la captura, mostrada en la figura 2.26, el driver se encuentra en funcionamiento. Los valores que están siendo leídos desde la base de datos y enviados al PLC se encuentran en la columna dato. Los botones que existen en esta pantalla cumplen la misma función que su contraparte en la pantalla de configuración.

Como ya se ha explicado antes, esta pantalla se encarga de escribir la información de y desde la base de datos hacia el PLC. Esta trabaja con el método que se ha mencionado anteriormente, véase método de lectura base de datos y escritura en PLC; sección 2.6.2, que recibe la información desde una base de datos MySQL, transforma esta información para que pueda ser insertada en una trama Profinet y la envía por la red hacia el PLC. El método inicialmente busca la última fila en la base de datos. Esta información se obtiene a partir de la subrutina antes mencionada en la cual se consigue la información referente a las características de la tabla de la base de datos.

Después de que el método tiene esta información, determina cuáles son las configuraciones que se van a ocupar dependiendo del número de datos que se seleccionó. Se toma el primer dato, con el nombre de la columna y su respectiva tabla, se selecciona un tipo de subrutina dependiendo del tipo de dato ocupado, después se ocupa el método de obtención de datos de MySQL para obtener la información.

2.10.3.2 Pantalla de Lectura

La pantalla de lectura tiene una configuración similar a la pantalla de escritura, en este caso las diferencias únicamente radican en el método, que en este caso es considerablemente más complejo, como se explicará más adelante en esta sección.

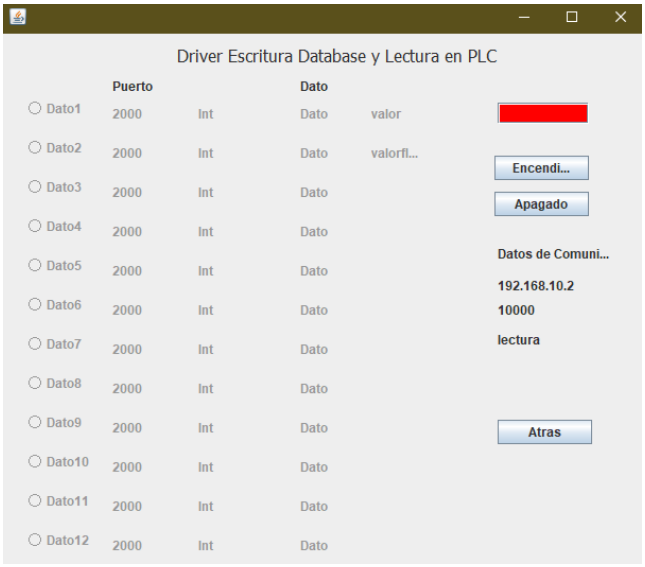


Figura 2.27. Ventana de Operador de conexión de lectura del PLC.

Como se puede observar en la figura 2.27 el sentido y la disposición de los elementos se conserva, referentes a la pantalla de operador de escritura. Los botones en la pantalla siguen su misma función que en su contraparte de escritura: encender y apagar el driver.

El método para esta pantalla es más complejo, debido a la cantidad de estamentos requeridos en lenguaje MySQL para la obtención de información del estado de la tabla desde el driver. El driver primero obtiene la información referente el número de columnas y número de filas; con esta y la cantidad de datos que el usuario selecciona, el programa determina la cantidad de columnas que deben añadirse en la tabla para cumplir con los requerimientos. El añadido de columnas se realiza con la subrutina de ALTER TABLE, en esta se añaden las columnas y se nombran con un número dependiendo del orden de estas, después existe otra función que se encarga de modificar únicamente los nombres de las columnas existentes, según los requerimientos del usuario. Es importante recalcar que no se modifica de ninguna forma el Key ID.

Después de la creación de las columnas necesarias, el método crea una fila de ceros debajo de la tabla. Esta fila de ceros será modificada mediante el estamento de *UPDATE.* Después de la creación de esta fila se verifica el tipo de dato por cada uno de los datos que se desea enviar y el tipo de método a utilizarse.

3. PRUEBAS Y RESULTADOS

Se presentan los resultados obtenidos en las pruebas realizadas para la verificación del funcionamiento y desempeño del driver.

Se desarrollan 4 secciones referentes a las pantallas que contempla el driver para su aplicación, dos secciones en relación con las pantallas de prueba de lectura y escritura; dos secciones referentes a las pantallas de usuario de prueba y escritura. Las pantallas de usuario se encuentran concatenadas con sus respectivas pantallas de configuración, ya que estas preceden a sus respectivas pantallas de usuario.

Se presenta los requerimientos para la realización de las pruebas, esto referente a las aplicaciones de comunicación entre la interfaz utilizada, la base de datos y el conector DSN. Se explica la instalación y configuración de este para un correcto funcionamiento.

3.1 PANTALLAS DE PRUEBA

Las pantallas de prueba tienen como objetivo el intercambio de información directa entre el driver y el PLC, siendo la principal utilidad de estas, la verificación de la comunicación entre las partes. No tienen ninguna relación con las bases de datos, su único propósito es el de funcionar una sola vez para verificar la comunicación o encontrar posibles errores existentes en esta. La interfaz en general presenta una configuración simple.

3.1.1 PRUEBA PANTALLA DE LECTURA

Esta pantalla se utiliza para la lectura de datos desde el PLC. En los objetivos del proyecto de titulación se encuentra estipulado el requerimiento de lectura de datos tipo entero, booleano y flotante. Debido a las características de la trama Profinet, el envío de datos booleanos se hará mediante bloques tipo byte.

La configuración necesaria es únicamente los puertos de cada uno de los datos y la IP de la computadora. Se presenta a continuación la configuración de la pantalla.

Puertos: Entero → 2000 Flotante → 3000 Booleano → 2003

Los valores establecidos en el PLC son los siguientes:

Entero → 23 Flotante → 26.5 Booleano → 1

La IP configurada es: 192.168.10.2

En la figura 3.1 se muestra la configuración realizada en el driver. Se incluye los puertos de uso y los valores obtenidos desde el PLC. Se presenta el estado de las variables en tiempo real desde el PLC, en su herramienta de programación TIA Portal.

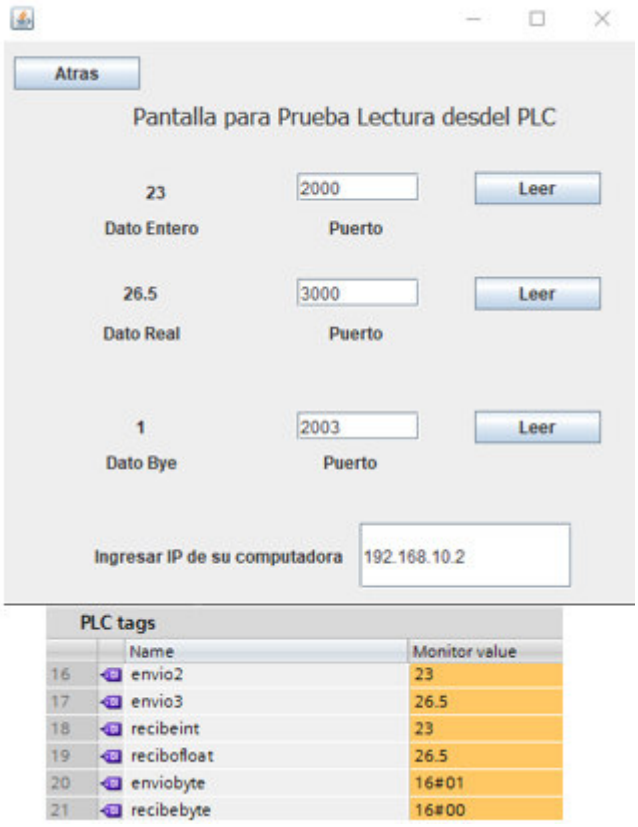


Figura 3.1. Resultados de las pruebas realizadas en la pantalla de prueba.

3.1.2 PRUEBA PANTALLA DE ESCRITURA

En la pantalla de prueba de escritura, se permite al usuario configurar la IP del PLC y los puertos para el manejo de datos. Esta pantalla no tiene acceso a la base de datos por lo tanto, el usuario debe generar información para ser enviada al PLC.

Una de las herramientas específicas de esta pantalla de prueba es la posibilidad de realizar un Ping al PLC, de tal forma que se pueda confirmar la conexión del controlador y del driver a la red. Se recomienda el uso de esta herramienta antes de la verificación de la escritura. Su uso se ve ejemplificado en la figura 3.2.

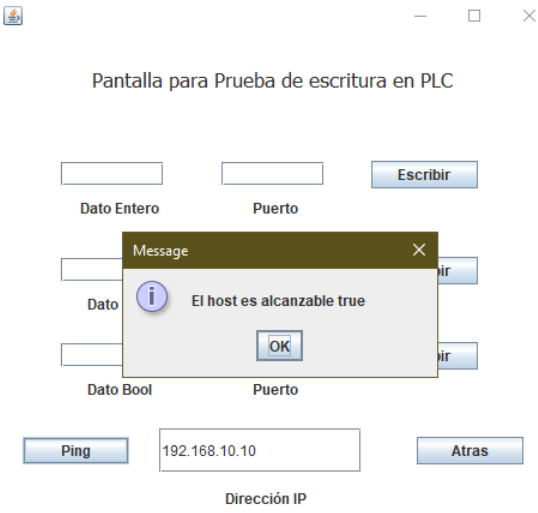


Figura 3.2. Verificación de conexión en red de controlador y driver.

Pasada la prueba de verificación de conexión, se puede realizar la escritura de datos. Es importante recalcar que, la escritura solo se realiza una vez; pero los datos se quedan guardados en las variables del PLC. En la figura 3.3 se muestra la aplicación y el estado de las variables modificadas por la pantalla de prueba.

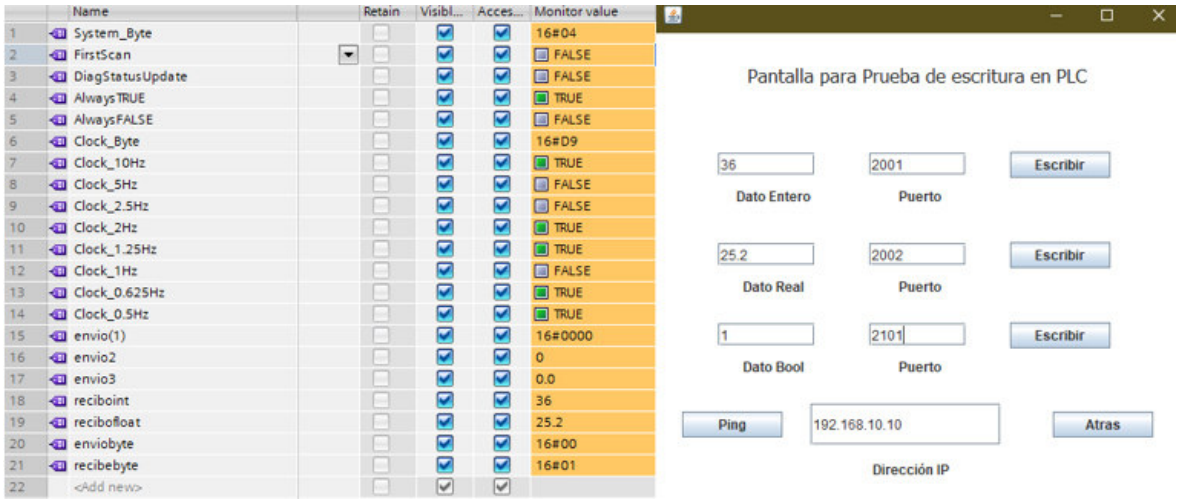


Figura 3.3. Escritura de datos en PLC y comprobación de estos.

Como se puede observar en ambas pantallas la comunicación y el objetivo de trabajo se cumple. Ya sea, en escritura, o en lectura el sistema demuestra ser funcional solamente considerando el driver y el PLC.

3.2 DESARROLLO DEL ENTORNO DE PRUEBAS

Para el proceso de prueba de las pantallas de configuración se realizó una interfaz de usuario, en el software comercial seleccionado. Este debe ser capaz de comunicarse con la base de datos, leer la información y escribirla. Uno de los requerimientos para que esto sea posible es la configuración y uso de un conector ODBC. El conector es obtenido directamente de la comunidad de MySQL y está orientado a la herramienta de MySQL Workbench.

3.2.1 CONECTOR MySQL WORKBENCH

El conector utilizado es de 32 bits. Se utiliza este conector debido a las características de Intouch, ya que al momento de utilizar el conector de 64 bits existe un error en el tamaño de los bloques de datos. La conexión del HMI con MySQL se da desde una máquina virtual, por lo tanto se prueba la comunicación de la base de datos en red con una interfaz de operador, demostrando el uso de la base de datos como base de datos relacional; aplicable en un sistema SCADA.

El conector se configura como un DSN, de tal forma que Intouch pueda acceder a la dirección de la base de datos y recuperar o escribir la información. El nombre del servidor DSN es requerido para la programación interna de la interfaz. Es importante recalcar que, la programación en la herramienta de desarrollo de la interfaz utiliza para el uso de bases de datos, estamentos similares a los utilizados en MySQL.

Debido a las características del conector ODBC, el ingreso a su pantalla de configuración no se hace mediante el panel de control, con la pantalla de configuración de DSN de 64 bits; sino que existe un comando específico que se ocupa en la herramienta de símbolo de sistema de Windows, que permite acceder a los conectores ODBC existentes en la computadora y la modificación de aquellos existentes en 32 bits.

3.2.1.1 Configuración Conector MySQL Workbench

Para empezar la configuración del conector, este debe estar previamente instalado en la computadora. Posteriormente en el símbolo del sistema se ingresa el siguiente comando:

- *c:\windows\sysWOW64\odbcad32.exe*

Partiendo de este comando se accede al administrador de origen de datos ODBC. Se permite la creación, eliminación y configuración de diferentes conectores. Al momento de seleccionar la creación de un nuevo origen de datos, se debe buscar el driver *MySQL*

ODBC Unicode Driver. Con la selección de este driver se despliega la pantalla de configuración del Data Source, la cual se puede observar en la figura 3.4.

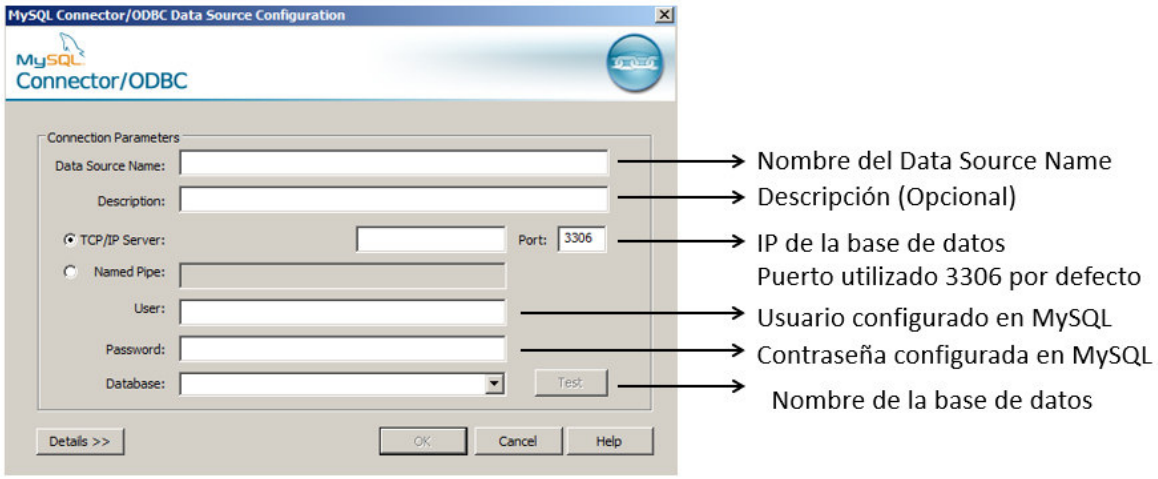


Figura 3.4. Pantalla de configuración de ODBC Data Source con sus partes.

Los campos por configurar en el Data Source son los siguientes:

- Nombre: se otorga el nombre que se ocupara la creación del Data Source y que se requiere para la programación en Intouch.
- TCP/ IP Server: La dirección IP donde se encuentra la base de datos en red, en este caso la configuración depende de la IP de la máquina física. La IP configurada es 192.168.10.2. Es importante revisar la existencia de la conexión entre la máquina virtual y la máquina física, en caso de no existir, la configuración del driver no se puede realizar.
- Puerto: El puerto es por defecto 3306, aunque el mismo puede ser cambiado en caso de ser necesario. Esta configuración se realiza en el programa de MySQL Workbench.
- Usuario: Se escribe el usuario creado en el programa MySQL Workbench, se recomienda la creación de un nuevo usuario para el uso del conector ya que la conexión con la base de datos puede no ser exitosa utilizando únicamente un usuario. Se recomienda leer el Anexo B.
- Contraseña: Se escribe la contraseña correspondiente al usuario.
- Database: Se escribe el Nombre de la base de datos configurada en la aplicación MySQL, en este caso el nombre es “tesis”.

Pasada la configuración de todos los parámetros antes mencionados, se procede a la comprobación del funcionamiento del conector mediante el botón *test,* siendo este una

herramienta que comprueba la conexión con la base de datos. El sistema configurado y su respectiva confirmación de conexión se presenta en la figura 3.5.

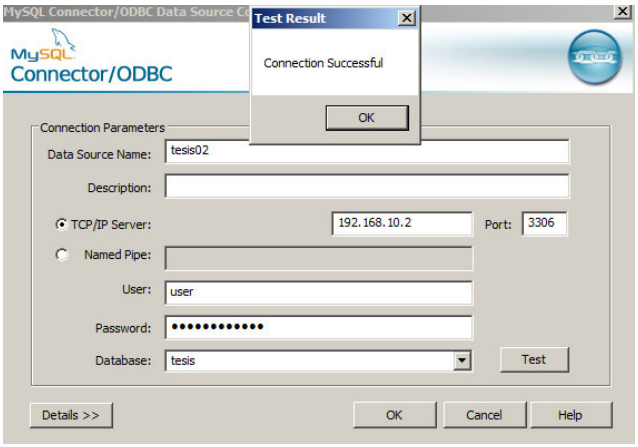


Figura 3.5. Pantalla configurada del conector ODBC, se incluye la verificación de la conexión con la base de datos.

Cuando se confirma la conexión es posible la creación del DSN. Este va a ser utilizado en la programación de Intouch. La que debe ser configurada para comunicarse con la base de datos.

3.2.2 INTERFAZ DE USUARIO CON INTOUCH

Para la creación de la interfaz de usuario que utiliza Intouch, se consideró el uso de cada tipo de variable, de tal forma que se pueda demostrar el cumplimiento del objetivo del manejo de los diferentes tipos de variables en Intouch. Estas variables son tipo booleano, entero y flotante.

El uso del driver desarrollado es bidireccional, por lo que se configuran tres variables de uso por cada sentido de comunicación. Es importante aclarar que la capacidad del driver es considerablemente mayor a lo verificado en las pruebas, siendo el objetivo de estas la demostración del funcionamiento básico en un entorno establecido para facilitar el trabajo del sistema.

La interfaz considerada es el uso de 2 tanques de almacenamiento con sus respectivos niveles, cada uno de estos tiene un nivel recibido del PLC y uno enviado. Un tanque utiliza las variables tipo entero mientras que otro utiliza variables tipo flotante. Existe una variable tipo booleana que permite el envío y la recepción de un dato; esta puede ser considerada como el encendido de un motor y la recepción del estado de este. Cada una de las partes mencionadas anteriormente se pueden observar en la figura 3.6.

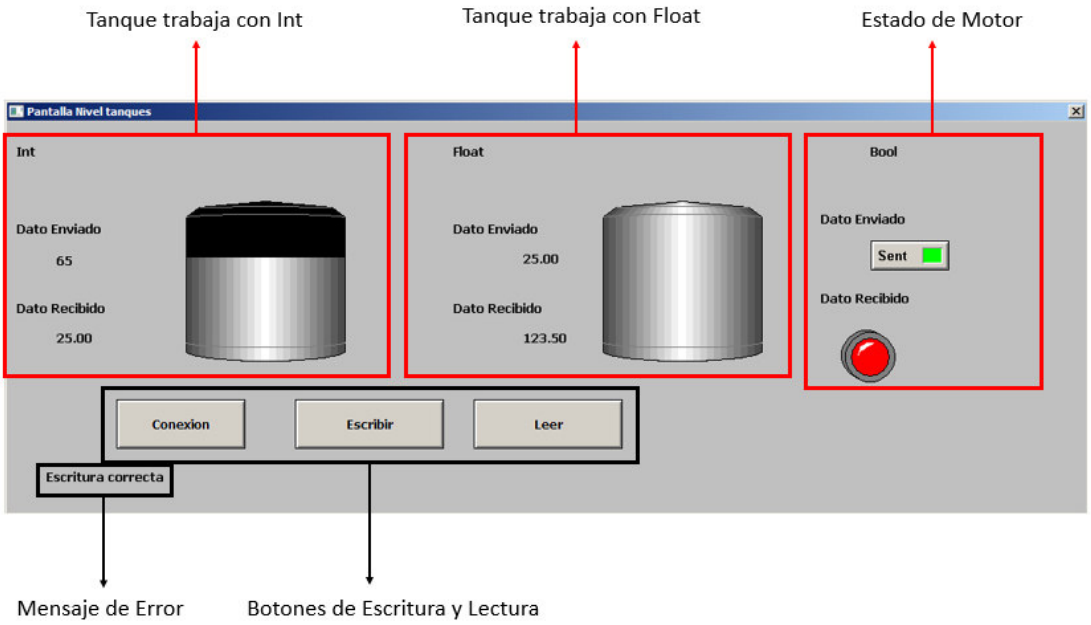


Figura 3.6. Pantalla de Interfaz de usuario de envío y recepción de 3 datos diferentes.

En la pantalla se puede apreciar el manejo de los 3 tipos de datos. Se ocupa un sistema de comunicación simple con la base de datos que permite la conexión, la escritura y la lectura de esta. Como el sistema solamente requiere la prueba general del sistema estos comandos simples permiten la comunicación necesaria con la base de datos, de tal forma que se pueda probar el uso del driver.

3.2.2.1 Programación Interna Intouch

Se debe realizar un proceso de programación interna que permita la comunicación con la base de datos. Esta debe ser realizada individualmente en cada uno de los botones utilizados. La programación está basada en la comunicación mediante "*bind list*" con MySQL. Se presenta a continuación, cada uno de los botones en la interfaz de usuario, véase figura 3.6; y su respectiva programación:

Conexión:

Este botón se centra en iniciar la comunicación y probar que la misma sea posible con la base de datos.

- *ResultCode=*
- *SQLConnect(connectionId, "Provider=MSDASQL;DSN=tesis02");*
- *Msg = SQLErrorMsg(ResultCode);*

Como se puede observar aquí se ocupa el DSN antes configurado. Es importante definir las variables requeridas para la comunicación.

Escribir:

- *SQLInsert(connectionId, "escritura", "lecturadatabasedesdeplc");*
- *IF ResultCode <> 0 THEN*
- *Msg = SQLErrorMsg(ResultCode);*
- *ELSE*
- *Msg = "Escritura correcta";*
- *ENDIF;*

Se observa el uso del estamento de *Insert* ya existente en el lenguaje MySQL, solamente que se ocupa con ciertas modificaciones. Se ocupa el código de resultado para mostrar los errores, en la programación, o en la configuración de la comunicación; en caso de existir.

Leer:

- *ResultCode = SQLSelect(connectionId, "lectura", "escrituradatabasedesdeplc", "","");*
- *ResultCode=SQLLast(connectionId);*

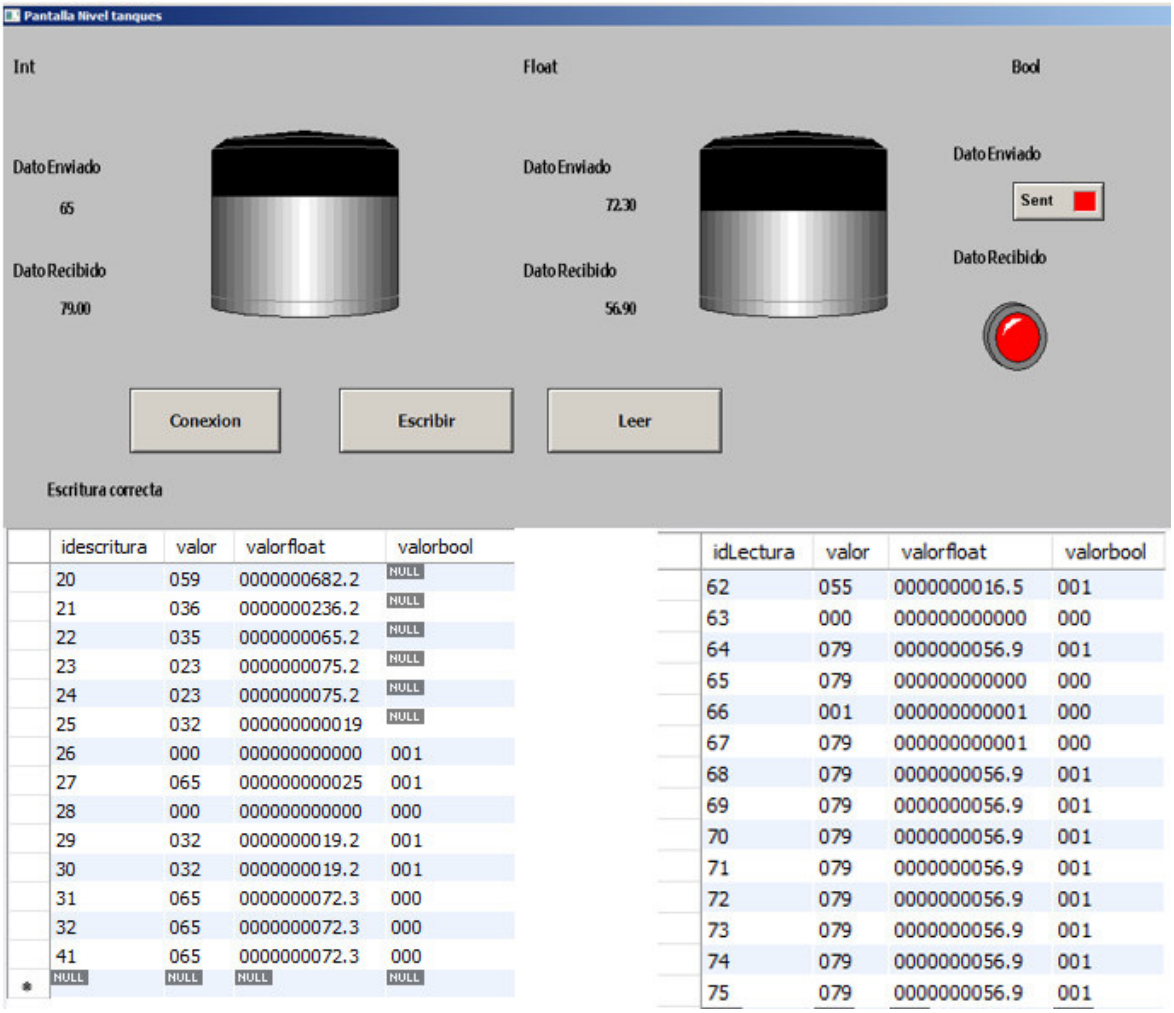
- *IF ResultCode <> 0 THEN*
- *Msg = SQLErrorMsg(ResultCode);*
- *ELSE*
- *Msg = "Lectura correcta";*
- *ENDIF;*

Se puede observar el uso de los estamentos "*SQLast*" y "*SQLSelect*". Siendo el estamento "*SQLSelect*" el utilizado para la recuperación de la información desde la base de datos. Como se estipuló anteriormente, el estamento tiene similitudes con su código hermano existente en el lenguaje MySQL, pero existen diferencias entre estos. El estamento *"SQLLast*" se ocupa para obtener los datos referentes a la última fila de la base de datos.

3.2.3. PRUEBA DE COMUNICACIÓN DE HMI CON MySQL

Para verificar que el interfaz desarrollado es capaz de comunicarse y modificar la base de datos, se comprueba el uso de los botones y las modificaciones o lecturas que realizan en la base de datos.

Primero se prueba la comunicación con la base de datos con el botón "*Conexión*". Se utiliza el texto inferior al botón que indica si la conexión fue correcta o existió un error en esta. A su vez, con este texto inferior se puede verificar la correcta escritura o lectura, que sirve para comprobar la correcta programación y configuración. Se presenta en la figura 3.7. la interfaz de operador y la base de datos, con un intercambio exitoso de datos.



idescritura	valor	valorfloat	valorbool
20	059	0000000682.2	NULL
21	036	0000000236.2	NULL
22	035	0000000065.2	NULL
23	023	0000000075.2	NULL
24	023	0000000075.2	NULL
25	032	000000000019	NULL
26	000	000000000000	001
27	065	000000000025	001
28	000	000000000000	000
29	032	0000000019.2	001
30	032	0000000019.2	001
31	065	0000000072.3	000
32	065	0000000072.3	000
41	065	0000000072.3	000
NULL	NULL	NULL	NULL

idLectura	valor	valorfloat	valorbool
62	055	0000000016.5	001
63	000	000000000000	000
64	079	0000000056.9	001
65	079	000000000000	000
66	001	000000000001	000
67	079	000000000001	000
68	079	0000000056.9	001
69	079	0000000056.9	001
70	079	0000000056.9	001
71	079	0000000056.9	001
72	079	0000000056.9	001
73	079	0000000056.9	001
74	079	0000000056.9	001
75	079	0000000056.9	001

Figura 3.7. Comunicación, escritura y lectura exitosa de la interfaz con la base de datos.

Como se puede observar en la figura 3.7, la comunicación se desarrolla de forma satisfactoria. A partir de la comprobación de esta se puede pasar a las pruebas referentes al driver.

3.3 PRUEBAS PANTALLA ESCRITURA PLC LECTURA BASE DE DATOS

En esta pantalla se leerá la información desde la base de datos, para ser escrita en el PLC. Se considera que toda la configuración anteriormente mencionada, véase literal 3.2 Desarrollo Entorno de Pruebas, ya se encuentra realizada. Las pruebas de comunicación del driver al PLC y de la Interfaz a la base de datos ya se encuentran realizadas.

Se presenta un resumen de la configuración necesaria.

PLC:

- IP del PLC → 192.168.10.10
- 1er dato → Dato tipo Entero → Puerto 2001
- 2do dato → Dato tipo Flotante → Puerto 2002
- 3er dato → Dato tipo Booleano → Puerto 2101

PC Física:

- IP de la PC → 192.168.10.2
- Estado de la base de datos → Encendida y en funcionamiento

Base de Datos:

Tabla 1: Lectura

- Columnas:
    - IdLectura
    - Valor (Valor tipo Entero)
    - Valorfloat (valor tipo flotante)
    - Valorbool (valor tipo booleano)

Tabla 2: Escritura

- Columnas:
    - Idescritura
    - Valor (valor tipo entero)
    - Valorfloat (valor tipo flotante)
    - Valorbool (valor tipo booleano)

Driver:

- Dato 1 → Puerto 2001 → Dato tipo Entero → Columna: valor
- Dato 2 → Puerto 2002 → Dato tipo Flotante → Columna: datofloat
- Dato 3 → Puerto 2101 → Dato tipo Booleano → Columna dato bool
- Nombre de tabla → escritura
- Cantidad de Datos → 3
- IP del PLC → 192.168.10.10
- Tiempo de loop → 1000 ms

PC Máquina Virtual HMI:

- IP de la PC → 192.168.10.32
- Estado del DSN → Activo y testeado

Partiendo de la configuración realizada es posible utilizar el driver. Para la comprobación del funcionamiento de todo el sistema se tiene en cuenta la pantalla de operador, ya que esta permite la visualización del dato enviado. Se tendrá en cuenta también la pantalla de configuración, pero únicamente será mostrada como un sistema paralelo a la pantalla de operador. Se presenta en la figura 3.8 la configuración de la pantalla de operador y la pantalla de configuración en funcionamiento.

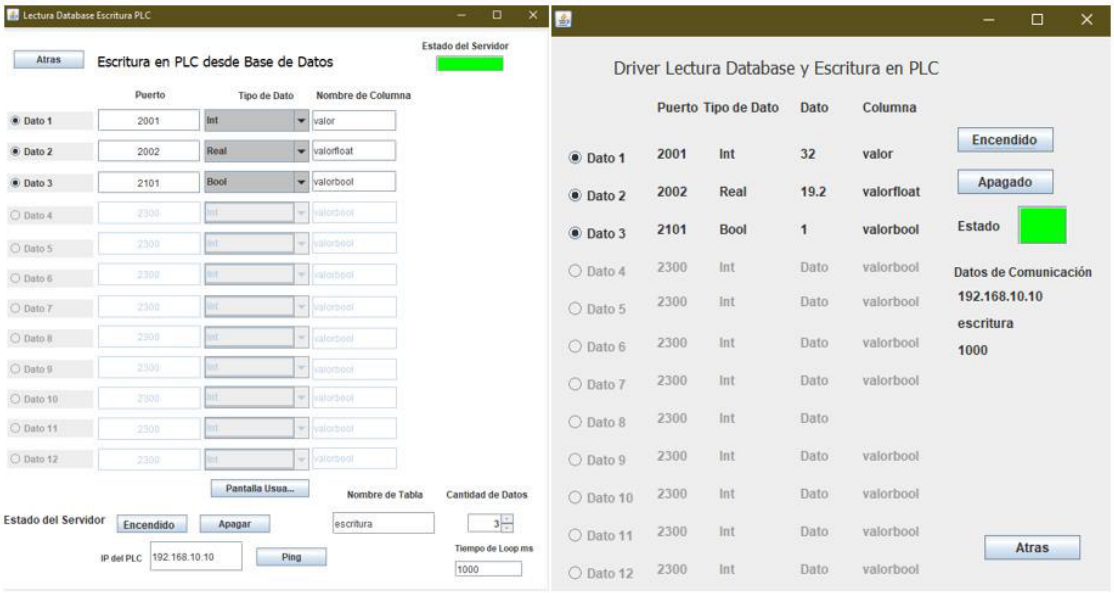
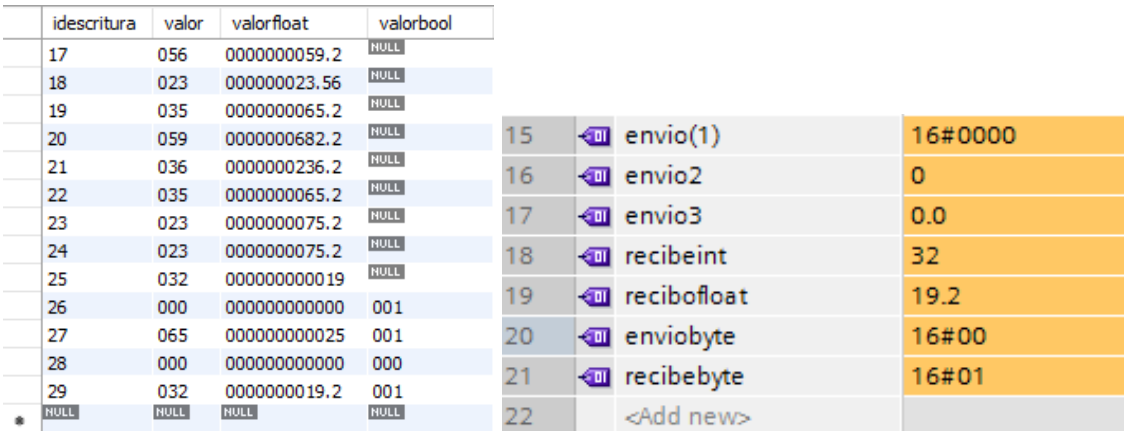


Figura 3.8. Pantallas de configuración y de operador, configuradas y en funcionamiento.

Con las pantallas en funcionamiento, es importante verificar, cómo estas modifican los estados de las marcas en el PLC y si los datos obtenidos de la base de datos son los

correctos. Se presenta en la figura 3.9. los estados de la base de datos en la tabla de escritura y en la aplicación de programación de siemens, en tiempo real.



idescritura	valor	valorfloat	valorbool
17	056	0000000059.2	NULL
18	023	000000023.56	NULL
19	035	0000000065.2	NULL
20	059	0000000682.2	NULL
21	036	0000000236.2	NULL
22	035	0000000065.2	NULL
23	023	0000000075.2	NULL
24	023	0000000075.2	NULL
25	032	000000000019	NULL
26	000	000000000000	001
27	065	000000000025	001
28	000	000000000000	000
29	032	0000000019.2	001
NULL	NULL	NULL	NULL

15		envio(1)	16#0000
16		envio2	0
17		envio3	0.0
18		recibeint	32
19		recibofloat	19.2
20		enviobyte	16#00
21		recibebyte	16#01
22		<Add new>	

Figura 3.9. Estado de las variables en la base de datos y el PLC.

Como se puede observar, el funcionamiento del driver es adecuado y cumple con los objetivos propuestos. Se propone su funcionamiento a la par de un uso con una interfaz de usuario, que ya fue desarrollada anteriormente. Se presenta en la figura 3.10 la interfaz de usuario con el estado de las variables deseadas, la base de datos con la recepción de estas, el driver con su pantalla de operador funcional y el estado de las variables en el PLC; modificado según lo establecido en la interfaz.

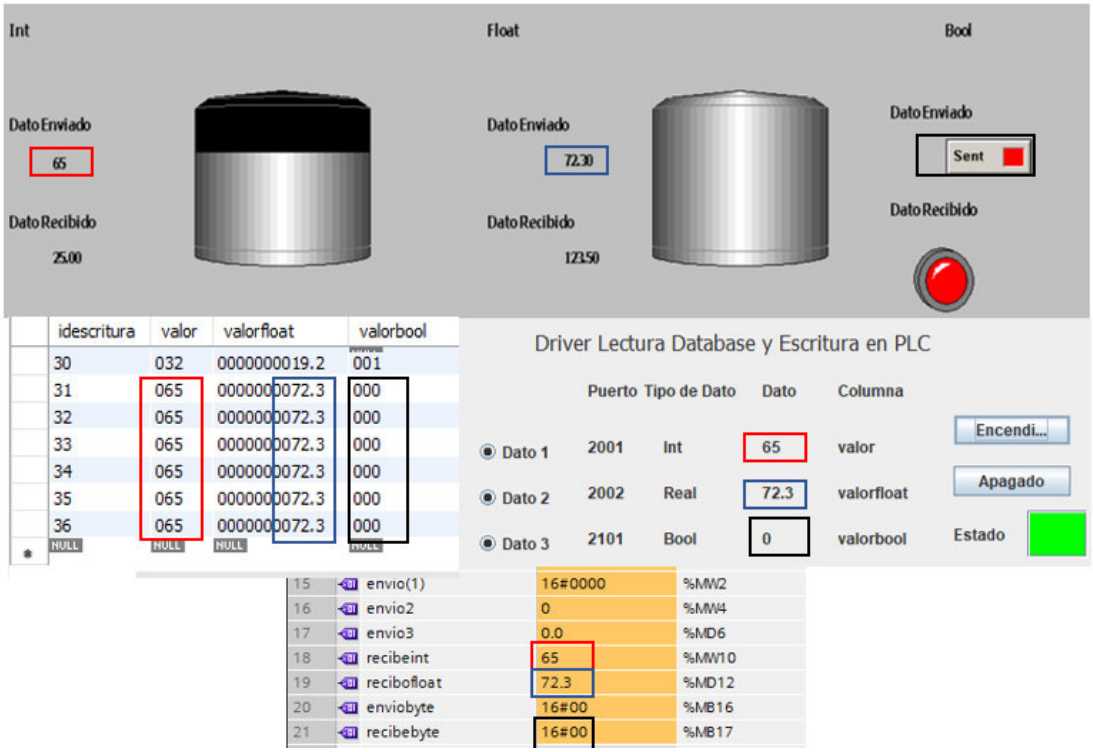


Figura 3.10. Estado de las variables enviadas en HMI, base de datos, driver y PLC.

En la figura 3.10. el dato entero esta señalado en los recuadros rojos, se observa su movimiento entre las diferentes aplicaciones. Su origen en la interfaz de usuario, a partir de aquí su envío a la base de datos, su recepción por parte del driver y su escritura en el PLC. El seguimiento mostrado se puede realizar, a su vez para el dato booleano y flotante; los cuales están representados por los colores azul y negro respectivamente.

3.4 PRUEBAS PANTALLA LECTURA PLC LECTURA BASE DE DATOS

Para el desarrollo de estas pruebas debe estar realizada la configuración previamente explicada, véase literal 3.2. La configuración del bloque de comunicación del PLC es de mucha importancia, ya que en caso de este encontrarse mal configurado, el sistema no puede funcionar, véase Programación de los bloques internos literal 2.5.

Se presenta un resumen de la configuración necesaria para realizar el sistema:

PLC:

- IP del PLC → 192.168.10.10
- 1er dato → Dato tipo Entero → Puerto 2000
- 2do dato → Dato tipo Flotante → Puerto 3000
- 3er dato → Dato tipo Booleano → Puerto 2003

PC Física:

- IP de la PC → 192.168.10.2
- Estado de la base de datos → Encendida y en funcionamiento

Base de Datos:

- Tabla 1: Lectura
    - Columnas:
        - IdLectura
        - Valor (Valor tipo Entero)
        - Valorfloat (valor tipo flotante)
        - Valorbool (valor tipo booleano)
- Tabla 2: Escritura

- Columnas:
  - Idescritura
  - Valor (valor tipo entero)
  - Valorfloat (valor tipo flotante)
  - Valorbool (valor tipo booleano)

Driver:

- Dato 1 → Puerto 2000 → Dato tipo Entero → Columna: valor
- Dato 2 → Puerto 3000 → Dato tipo Flotante → Columna: datofloat
- Dato 3 → Puerto 2003 → Dato tipo Booleano → Columna datobool
- Nombre de tabla → lectura
- Cantidad de datos → 3
- IP del PLC → 192.168.10.10
- Tiempo de loop → 10000 ms

PC Máquina Virtual HMI:

- IP de la PC → 192.168.10.32
- Estado del DSN → Activo y probado.

Partiendo de la configuración antes mostrada, se pone el driver en funcionamiento. En la figura 3.11. se observa las pantallas de configuración y de operador en funcionamiento.

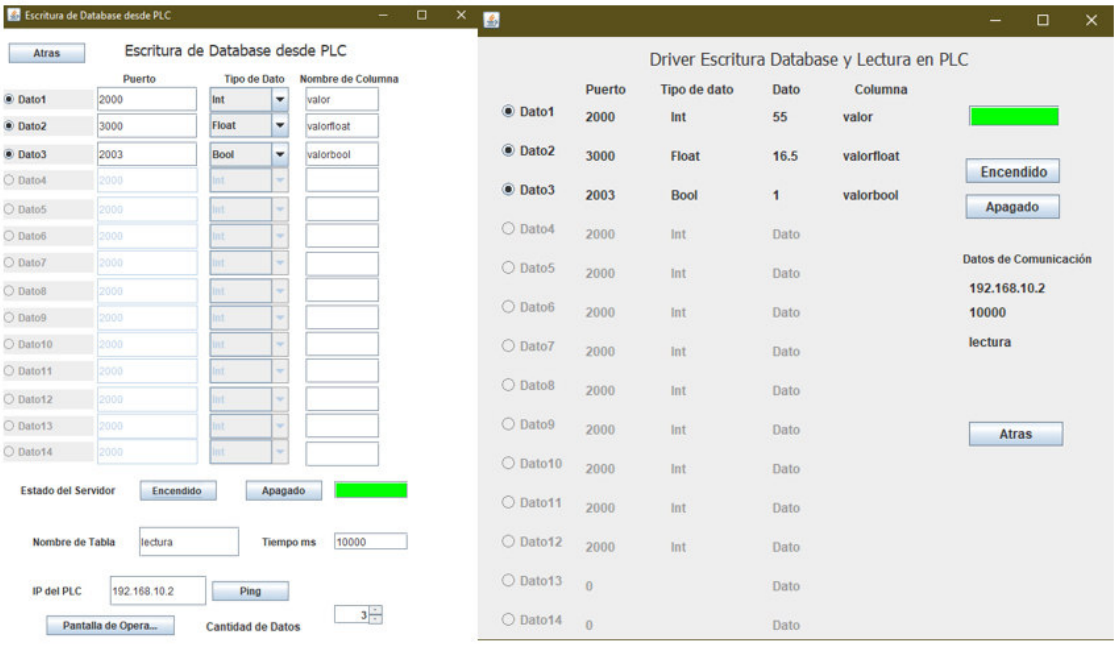
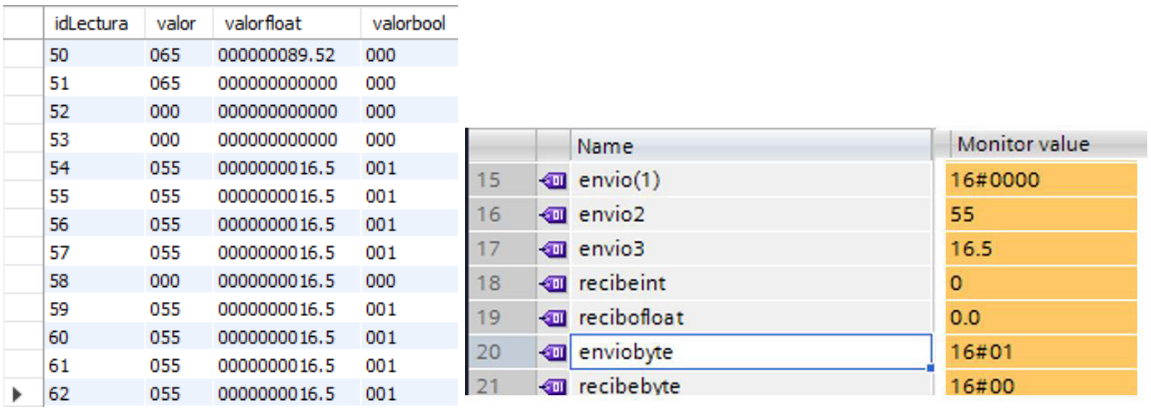


Figura 3.11. Pantallas de configuración y de operador, configuradas y en funcionamiento.

Como se observa, el funcionamiento es adecuado y no presenta errores. Partiendo de este funcionamiento se acopla la interfaz de usuario antes utilizada en el sistema de escritura en el PLC. En la figura 3.12. se muestra el estado de las variables en la interfaz de usuario así como en el PLC, en tiempo real.

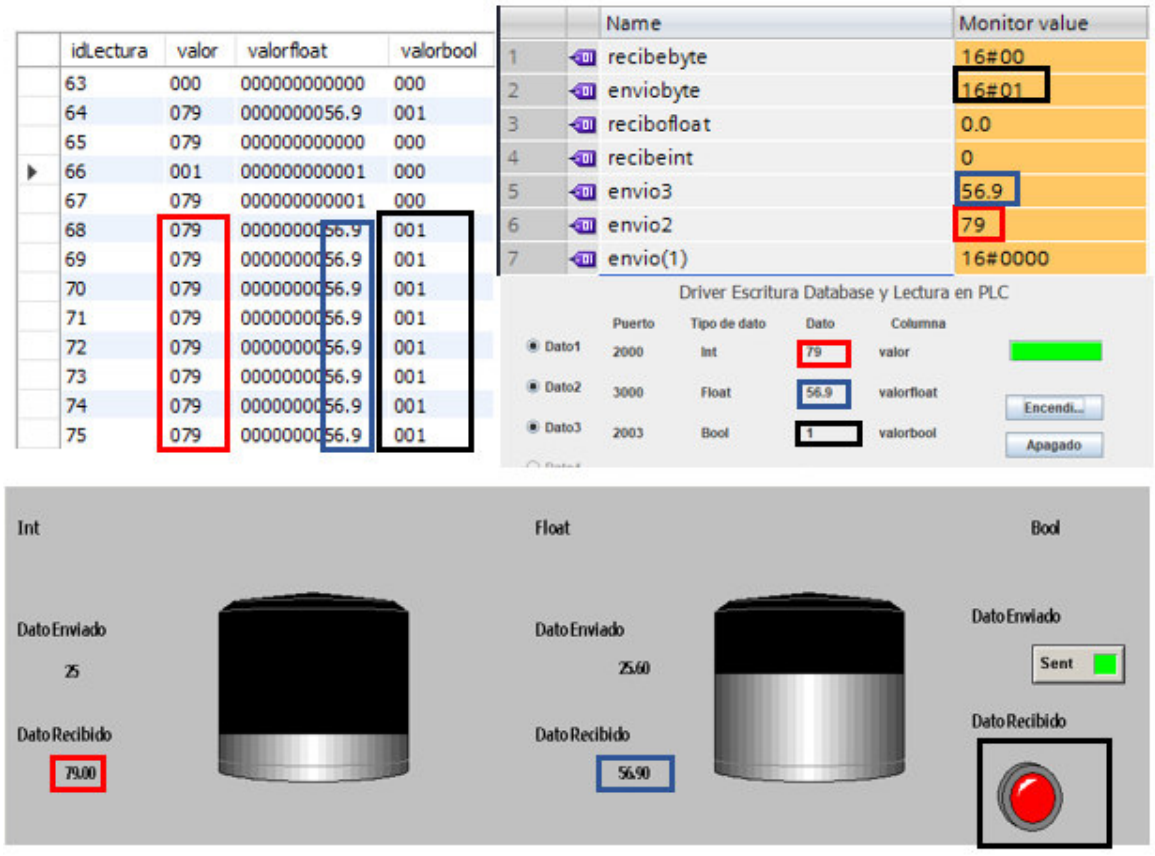


	idLectura	valor	valorfloat	valorbool
	50	065	000000089.52	000
	51	065	000000000000	000
	52	000	000000000000	000
	53	000	000000000000	000
	54	055	0000000016.5	001
	55	055	0000000016.5	001
	56	055	0000000016.5	001
	57	055	0000000016.5	001
	58	000	0000000016.5	000
	59	055	0000000016.5	001
	60	055	0000000016.5	001
	61	055	0000000016.5	001
▶	62	055	0000000016.5	001

	Name	Monitor value
15	envio(1)	16#0000
16	envio2	55
17	envio3	16.5
18	recibeint	0
19	recibofloat	0.0
20	enviobyte	16#01
21	recibebyte	16#00

Figura 3.11. Pantallas de configuración y de operador, configuradas y en funcionamiento.

El funcionamiento del driver es el esperado. Con el sistema funcionando se incluye el uso de la interfaz de usuario, que se muestra en la figura 3.12.



	idLectura	valor	valorfloat	valorbool
	63	000	000000000000	000
	64	079	0000000056.9	001
	65	079	000000000000	000
▶	66	001	000000000001	000
	67	079	000000000001	000
	68	079	0000000056.9	001
	69	079	0000000056.9	001
	70	079	0000000056.9	001
	71	079	0000000056.9	001
	72	079	0000000056.9	001
	73	079	0000000056.9	001
	74	079	0000000056.9	001
	75	079	0000000056.9	001

	Name	Monitor value
1	recibebyte	16#00
2	enviobyte	16#01
3	recibofloat	0.0
4	recibeint	0
5	envio3	56.9
6	envio2	79
7	envio(1)	16#0000

Figura 3.12. Estado de las variables enviadas en HMI, base de datos, driver y PLC.

En la figura 3.12. se puede observar la generación de información desde el PLC. El driver se encarga de recibir esta información mediante protocolo Profinet y procesarla para enviarla a la base de datos, la que es modificada previamente para estar adecuada para la recepción de la información según las especificaciones del usuario. La interfaz lee la información desde la base de datos y la muestra de forma sencilla al operador.

Los datos enteros se muestran en recuadros rojos y se puede observar, el movimiento de la información entre el PLC, el driver, la base de datos y el interfaz. Los datos flotantes y booleanos se muestran en los recuadros azul y negro, respectivamente.

3.5 COMPARACIÓN DEL SERVIDOR REALIZADO CON SOFTWARE WONDERWARE DAServer.

Se realiza un proceso comparativo entre las funciones del servidor DAServer de Wonderware, DassiDirect, y el prototipo realizado. El resumen comparativo se basa en las capacidades de cada uno de los servidores y características de funcionamiento en el software, se observa en la tabla 3.1.

Tabla 3.1. Tabla Comparativa entre el servidor desarrollado y Wonderware DassiDirect

Descripción	Profinet DAServer	Servidor Desarrollado
Sistema Operativo	Windows	Cualquiera
Conectividad	Aplicaciones de Windows que actúen como cliente DDE, SuiteLink u OPC	Aplicaciones que manejen lenguaje SQL.
Licencia	Licencia pagada	Software Libre
Tiempo de bucle	Configurable < 15s hasta 1000 ms	Configurable > 1s
Dirección IP	Ingreso de dirección IP	Ingreso de dirección IP y opción de comprobación de conexión
Puertos	1 puerto por PLC	1 puerto por dato

Configuración	Se requiere configurar device groups y device item.	Configuración de información de red por PLC.
Visualización	Visualización en Consola de Wonderware	Visualización en Interfaz desarrollada en Java
Variables Manejables	Todas las consideradas en la familia Siemens	Datos entero, flotante y booleano.

El software desarrollado trabaja con protocolo Profinet, y Wonderware trabaja con DDE, SuiteLink u OPC; siendo Wonderware un sistema con protocolos especializados en interfaces de usuario. La tabla comparativa, tabla 3.1, presenta una perspectiva de aplicación donde no se toma en cuenta el objetivo de uso de diferentes protocolos.

A partir de las pruebas de validación de funcionamiento del servidor, se desarrollan los datos en la tabla 3.2. donde se realiza un proceso sumario del costo del desarrollo del prototipo.

Tabla 3.2. Tabla de Costos de Desarrollo

Descripción	Software	Costo	Observaciones
Software de programación	JAVA IDE Eclipse	$0	Gratuito
Base de datos	MySQL Workbench	$0	Gratuito
Interfaz de operador	Depende de Aplicación de Usuario	-	Depende del Usuario Final
Desarrollo de Software	-	$2400	Horas Invertidas *5$
Instalación	-	$25	Valor Variable Consideración

Partiendo de los resultados obtenidos del costo de desarrollo del driver, se realiza una comparativa con los precios de los programas existentes en el mercado revisados previamente. Se valida la accesibilidad de precio que el driver puede tener en relación con sus contrapartes existentes comercialmente, logrando esto mediante el uso de herramientas de software libre que no tienen costo alguno. La aplicación de Java y MySQL permite al prototipo reducir de forma significativa sus costos de desarrollo, debido a que una cantidad considerable del presupuesto ocupado para la implementación de un sistema industrial, con una interfaz de usuario, se ocupa en software de comunicación.

Se presenta en la tabla 3.3 una comparativa de los precios en el mercado de drivers para comunicación industrial existentes, con características similares al prototipo; y un costo aproximado del driver. Esto teniendo en cuenta una expectativa de retorno de inversión en 100 ventas del programa, siendo esta expectativa de ROI (*return over investment*) menor a la de sus contrapartes comerciales.

Tabla 3.3. Tabla Comparativa Costos Comerciales

Nombre del Programa (Desarrollador)	**Costo**
DASSIDirect (Wonderware)	3329.43$
Matrikon OPC (Matrikon)	2300$
NI OPC (National Instruments)	3925$
Kepserverex OPC (Kepware)	475$
Prototipo	49$ (Instalación y programa)

El análisis realizado considera un costo de 49$ por programa, esto sin tener en consideración otros factores externos que determinan el valor de un sistema en el mercado. En este proyecto de titulación no se realiza un análisis financiero para establecer un valor comercial del prototipo. El objetivo de la comparativa de precios con programas existentes comercialmente es establecer que el driver desarrollado, con un entorno comercial hostil y una expectativa de venta casi nula; es capaz de presentar una solución asequible para la pequeña y mediana empresa.

4. CONCLUSIONES Y RECOMENDACIONES

De acuerdo con el desarrollo y las pruebas observadas, con el funcionamiento del servidor de datos industrial desarrollado, se puede llegar a las siguientes conclusiones y recomendaciones.

4.1 CONCLUSIONES

El driver ha demostrado claramente que está calificado para obtener información desde una red de PLCs que usan el protocolo Profinet y decodificar esta información de tal forma que pueda ser utilizada por cualquier aplicación que tenga la habilidad de obtener información desde una base de datos.

Se concluye que, el uso de software libre permite cumplir el objetivo de desarrollar un driver que cumpla con los requerimientos de interoperabilidad para trabajar con diferentes tipos de aplicaciones que puedan acceder a bases de datos relacionales. Con la correcta configuración e implementación del driver, este puede ser utilizado en un entorno industrial como una herramienta de comunicación para sistemas de control supervisorio.

El driver puede ser desarrollado como una aplicación mucho más compleja. La propuesta a largo plazo como la siguiente mejora aplicable, es de un driver basado en comunicación con la nube, con aplicaciones de históricos. Esta mejora requiere un tipo diferente de comunicación entre la base de datos y el driver, así como mejores funciones de seguridad relacionadas a la comunicación.

Es posible la implementación de otras herramientas que permitan sistemas de diagnóstico de la comunicación, así como eliminación del uso de base de datos y sistemas de redundancia para seguridad de la conexión. Esto en orden de un manejo más simple del prototipo y cumplir las expectativas de uso de CC-B según lo establecido por Siemens[11].

En la aplicación a futuro del driver en Profinet I/O, se considera un sistema de reconocimiento de puertos para elementos de campo. Permitiendo al usuario configurar los puertos de comunicación sin necesidad de buscar las características del entramado TCP establecidas en el componente de campo, con el que se busca establecer la comunicación.

Conforme a los resultados obtenidos en las pruebas realizadas en el driver, se puede concluir que la aplicación del prototipo esta limitada por la capacidad de trabajo de la computadora en la cual se encuentra instalado, por el trafico de datos en MySQL y el control que se tenga en el intercambio de datos en red.

El driver presenta una opción asequible económicamente para la pequeña y mediana empresa, el uso de herramientas de software libre permite reducir de forma considerable el costo de desarrollo. Con un precio tan reducido el prototipo puede desenvolverse como una solución viable de bajo costo, aún en un entorno comercial hostil y una expectativa de alcance conservadora.

4.2. RECOMENDACIONES

Se recomienda mantener los tiempos de comunicación en valores altos, mayores a 5 segundos, que permitan una revisión eficiente de los datos. Caso contrario, la modificación continua de la base de datos puede generar un error en la comunicación con el prototipo reduciendo la efectividad del sistema.

Se recomienda verificar el protocolo de comunicación en diferentes PLC, se enfatiza los PLCs de la familia S7 fuera del modelo S7-1200. Esto para comprobar el funcionamiento en PLCs con diferentes capacidades de Profinet, o modificar la programación para corregir la adaptabilidad del driver.

Se recomienda utilizar la herramienta de prueba en el prototipo, que permite verificar el funcionamiento independiente entre el PLC y el driver. El uso de esta reduce de forma considerable posibles errores de comunicación en el sistema, entre el controlador y el programa.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] D. F. A. ESPIN, “Desarrollo De Una Herramienta Computacional Que Contenga Comunicación Modbus Rtu Y Modbus Tcp Para La Implementacion De Sistemas De Control Supervisorio Y Adquisición De Datos a Bajo Costo”, *Esc. Politécnica Nac. Fac.*, 2018.

[2] Atul, S. Ghosh, y S. Balamurugan, “Cement plant simulation and dynamic data communication using NI labview and matrikon OPC”, *J. Theor. Appl. Inf. Technol.*, 2012.

[3] P. Wenzel, “Profinet - Lösungsplattform für die Prozessautomatisierung”, 2015.

[4] Softing Industrial Automation y A. Knoll, “OPC UA als Schlüssel zur Industrie 4.0”, *Markt&Technik*, núm. 24, pp. 1–3, 2013.

[5] I. Systems, “Wonderware ® SIDirect DAServer User ’ s Guide”, pp. 1–156.

[6] V. Maldonado, “Desarrollo de un servidor de Datos industrial con protocolo Modbus

TCP para los 8 códigos de función básicos.”, *EPN*, p. 109, 2021.

[7] T. E. Way y N. A. Edition, “PROFINET Technology the easy way to PROFINET PROFINET features”, 2010.

[8] X. Wu, L. Xie, y F. Lim, “Network delay analysis of EtherCAT and PROFINET IRT protocols”, en *IECON Proceedings (Industrial Electronics Conference)*, 2014.

[9] P. Contact, “Tendencias en Buses de Campo y Ethernet industrial”.

[10] D. Pliatsios, P. Sarigiannidis, T. Lagkas, y A. G. Sarigiannidis, “A Survey on SCADA Systems: Secure Protocols, Incidents, Threats and Tactics”, *IEEE Commun. Surv. Tutorials*, vol. 22, núm. 3, pp. 1942–1976, 2020.

[11] D. Comer, “Internetworking With TCP / IP”, *Internetworking Res. Exp.*, 2005.

[12] S. Industry, “S7- 1200 Programmable controller S7-1200 Programmable controller”, 2012.

[13] B. ÖZKILBAÇ, “Implementation and Design of 32 Bit Floating-Point ALU on a Hybrid FPGA-ARM Platform”, *Brill. Eng.*, 2019.

[14] J. Farrell, *JAVA™ PROGRAMMING*. .

[15] H. Schildt y H. Schildt, *Java : the complete reference*. 2007.

[16] E. Foundation, “The Eclipse Foundation”, vol. 0, núm. October. 2021.

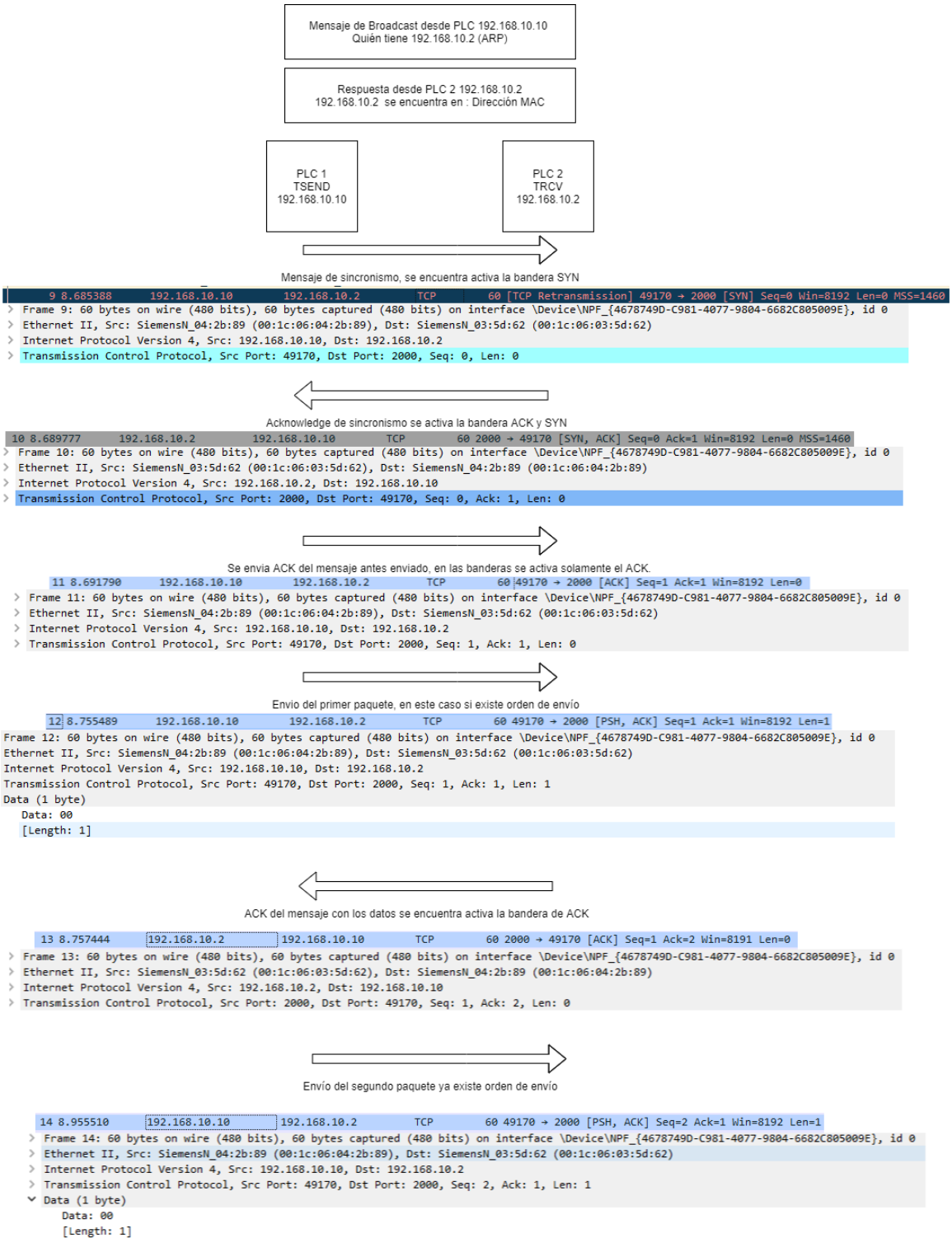
[17] D. Vohra, *Advanced Java EE Development with WildFly*. 2015.

[18] M. Matthews, J. Cole, y J. D. Gradecki, *MySQL and Java Developer’s Guide*. 2003.

[19] Handy Wicaksono, Resmana Lim, y William Sutanto, “Perancangan SCADA Software dengan Wonderware InTouch Recipe Manager dan SQL Access Manager pada Simulator Proses Pencampuran Bahan”, *J. Tek. Elektro*, 2008.

ANEXO A

Diagrama guía para orientación del funcionamiento de la comunicación Profinet.



ANEXO B

ÍNDICE DE CONTENIDO

B.1 INTRODUCCIÓN

El servidor de datos industrial desarrollado puede ser ejecutado en cualquier sistema operativo, que tenga instalado Java. Este permite realizar un intercambio de información con dispositivos industriales que manejen el protocolo Profinet TCP. Se requiere a su vez un programa de base de datos basado en MySQL que pueda ser modificado por el usuario para la preparación del entorno de trabajo.

El servidor cuenta con 3 modos de funcionamiento, subdivididos en lectura y escritura en el PLC. Se considera 1 modo de prueba que permite al usuario revisar la conexión con el controlador y la comunicación con los diferentes tipos de datos. El modo de configuración permite una pantalla de configuración para análisis del pase de información entre la base de datos, el driver y el PLC. Finalmente el sistema de usuario solamente permite el encendido y apagado del driver; se accede pasada la configuración en las pantallas previas y la prueba del intercambio de datos en la arquitectura del sistema.

Posteriormente, se presentan los requisitos previos, las pantallas a ocupar y configuración.

B.2 REQUISITOS PREVIOS

Para el uso del modo de prueba no se requiere de ningún requisito previo, solamente tener la conexión realizada con el PLC, en una red LAN; y el driver instalado.

Para las otras pantallas de funcionamiento del prototipo se requiere, que se encuentren instalados estos programas y complementos; además del driver.

- MySQL Workbench
- MySQL Conector ODBC

Estos deben de encontrarse correctamente configurados y deben ser capaces de comunicarse en la red con el PLC y el computador donde se piensa programar la interfaz de usuario de planta. Estas configuraciones se presentan a continuación.

B.3 CONFIGURACIONES PREVIAS

Se presentará las configuraciones referentes al conector ODBC requerido para la comunicación con la base de datos por parte de Intouch y MySQL Workbench.

B.3.1 Base de Datos MySQL

Es requerido la creación de una base de datos en MySQL, que almacenará los datos obtenidos desde el PLC y desde la interfaz de operador. En la pantalla inicial de MySQL Workbench se debe seleccionar la creación de una nueva base de datos (*Create New Schema*), el nombre utilizado en la programación para esta base de datos es tesis. Este nombre puede ser cambiado a conveniencia, pero se requiere su modificación en la programación interna del método de conexión MySQL; se recomienda mantener el nombre propuesto. En la figura B.1 se presenta la pantalla de creación de una nueva base de datos en MySQL Workbench, señalando en cuadros las partes importantes de esta.

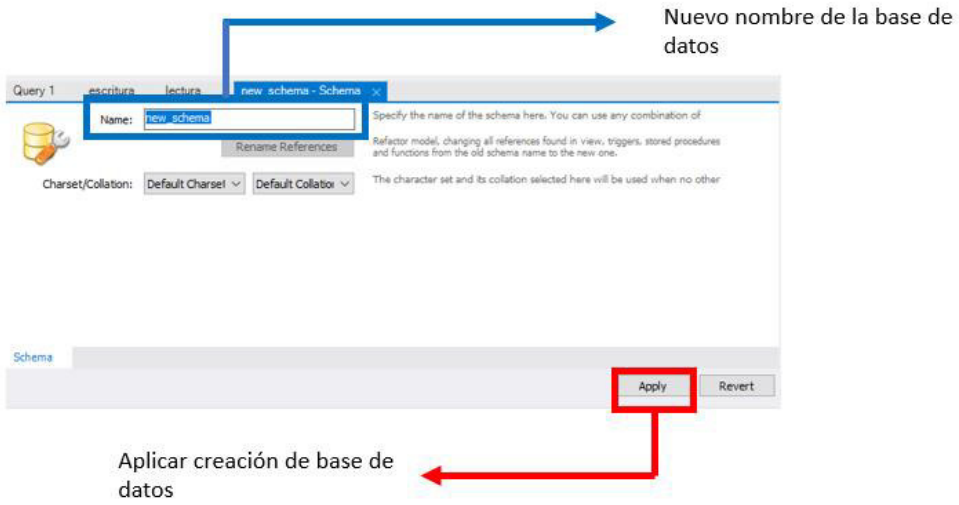


Figura B.1 Creación de base de datos en MySQL Workbench.

Partiendo de la creación de la base de datos se requiere la creación de las tablas en función de la cantidad de pantallas que se ocuparán para almacenamiento de la información en MySQL. Estas se crean haciendo clic izquierdo en la subdivisión *Tables* de la base de datos creada, desplegando un submenú que nos permite crear y nombrar nuevas tablas en la base de datos. Los nombres de las tablas pueden ser modificados según los requerimientos del usuario y no es requerido acceder a la programación para esto. En la figura B.2 se presenta este proceso.

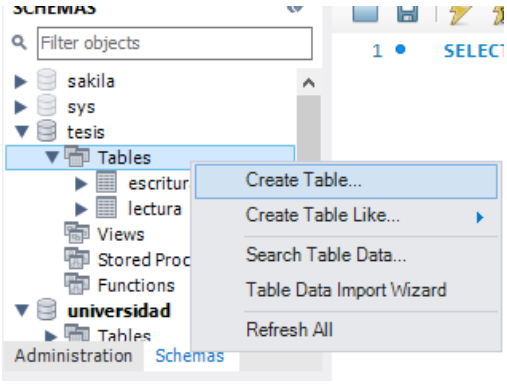


Figura B.2 Creación de tablas en MySQL Workbench.

Finalmente se deben de crear las KeyID y columnas requeridas en función de la cantidad de datos a compartir en cada base de datos. Estas se generan por el usuario en una pantalla de configuración de tabla posterior a la creación de esta. La configuración de las columnas con las variables a trabajar no es un requerimiento obligatorio pero ayuda método utilizado en la base de dato dentro del prototipo. En la figura B.3 se presenta esta pantalla y las configuraciones utilizadas previamente para prueba del driver.

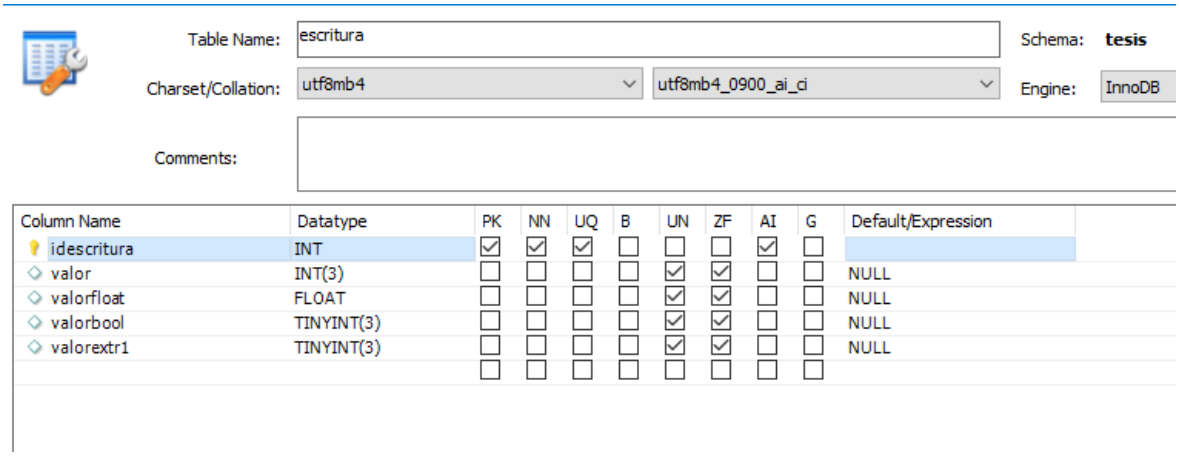


Figura B.3 Creación columnas y de KeyID en MySQL Workbench.

B.3.2 Creación de Usuario con Permisos en MySQL

La conexión con la base de datos desde el conector ODBC requiere de la creación de un usuario con todos los permisos de MySQL. El usuario generado automáticamente al momento de la creación de la base de datos no es una alternativa viable para esta comunicación, por lo que se detalla a continuación el proceso de creación de uno con permisos dentro de Workbench.

1. Abrir MySQL y conectarse en el usuario predeterminado

2. Ir a *Administration,* dentro de la base de datos, *y* seleccionar *Users and Privileges* como se presenta en la figura B.4

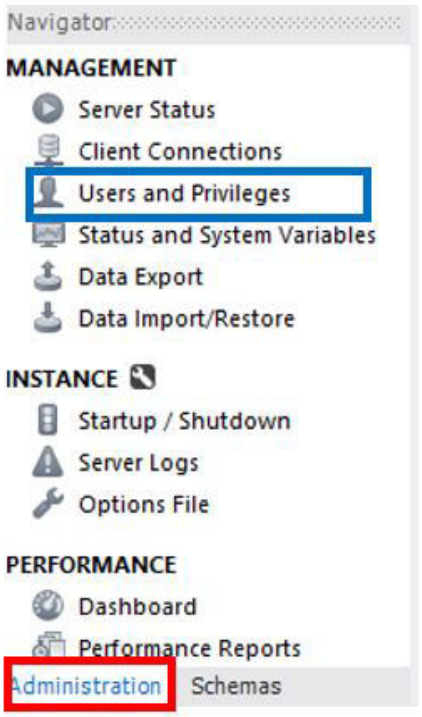


Figura B.3 Pestaña de herramientas en MySQL Workbench.

3. Dentro de esta pantalla se desplegará los usuarios ya existentes. Se deberá seleccionar la opción de *Add Account* para la creación de una nueva cuenta. Dentro de la pantalla que se habilitará se configura las características de la cuenta según lo presentado en la figura B.4 (El nombre de cuenta y la contraseña serán a elección del operador)

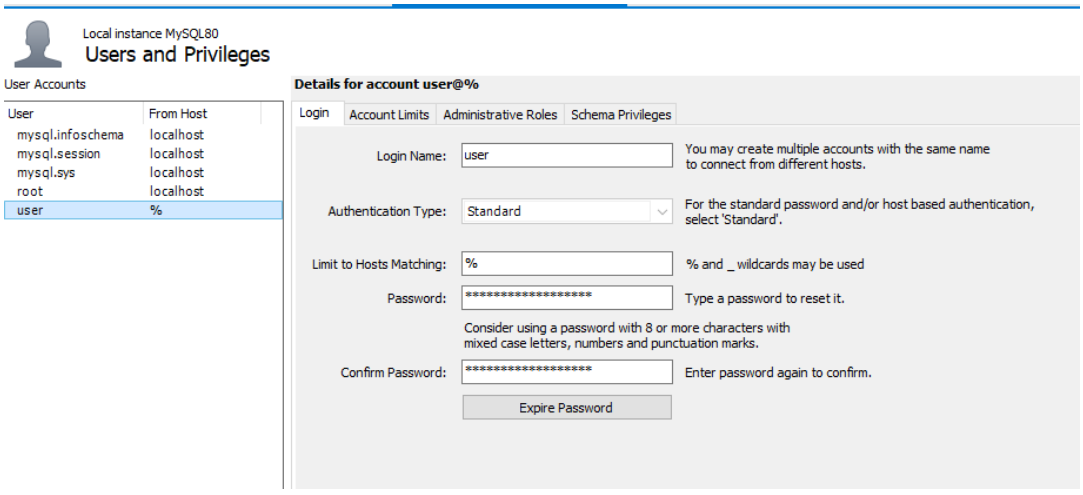


Figura B.4 Pantalla de configuración de nuevo usuario en MySQL Workbench.

4. Aplicar la creación del usuario

5. En la pestaña *Schema Privileges,* dar clic a *Add Entry,* dar clic en OK. Seleccionar todos con el botón *Select All*, para entregar todos los privilegios de acceso a este usuario. Se presenta la pantalla del proceso en la figura B.5.

6. Aplicar los cambios y guardar.

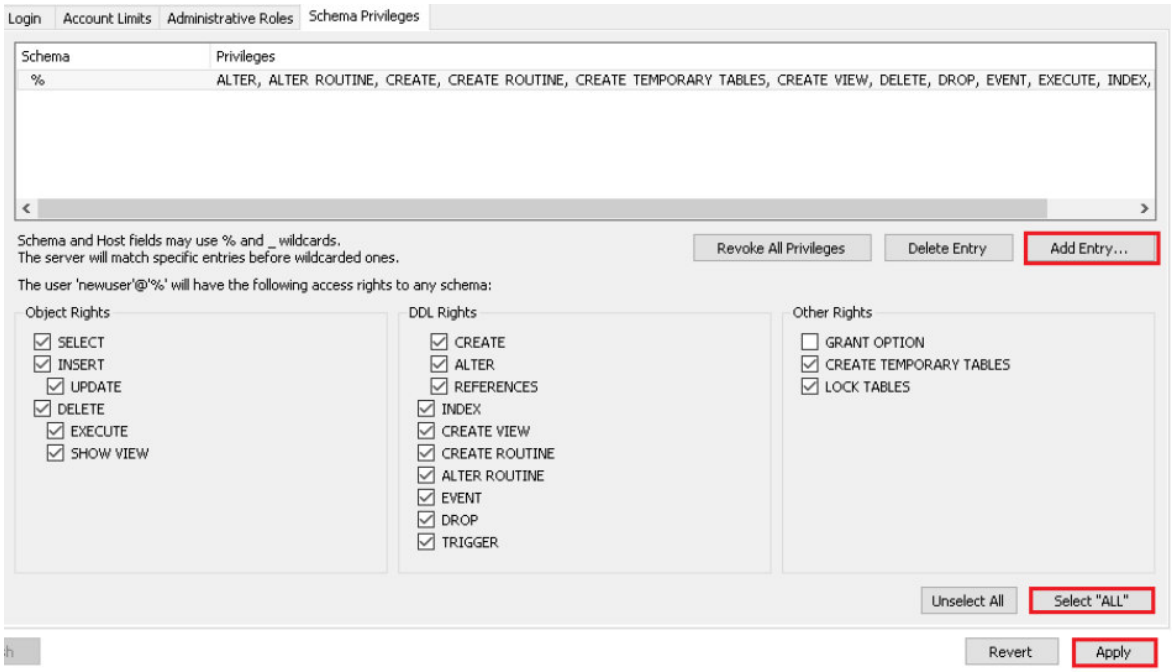


Figura B.5 Pantalla de configuración de nuevo usuario en MySQL Workbench.

B.3.3 Configuración conector ODBC

Se presentan los pasos requeridos para la configuración del conector en la máquina donde se encuentra el interfaz de usuario en Intouch.

1. Instalar el conector ODBC de MySQL Workbench en 32 bits

2. Ingresar a ODBC (32 bits). En caso de no encontrar el conector en el administrador de datos se puede ingresar la siguiente dirección para la configuración de este: *c:\windows\sysWOW64\odbcad32.exe*

3. En la figura B.6 se presenta la ventana del administrador de origen de datos ODBC. Se añade un DSN con el botón *Add* y se selecciona el MySQL ODBC 8.0 Unicode Driver.

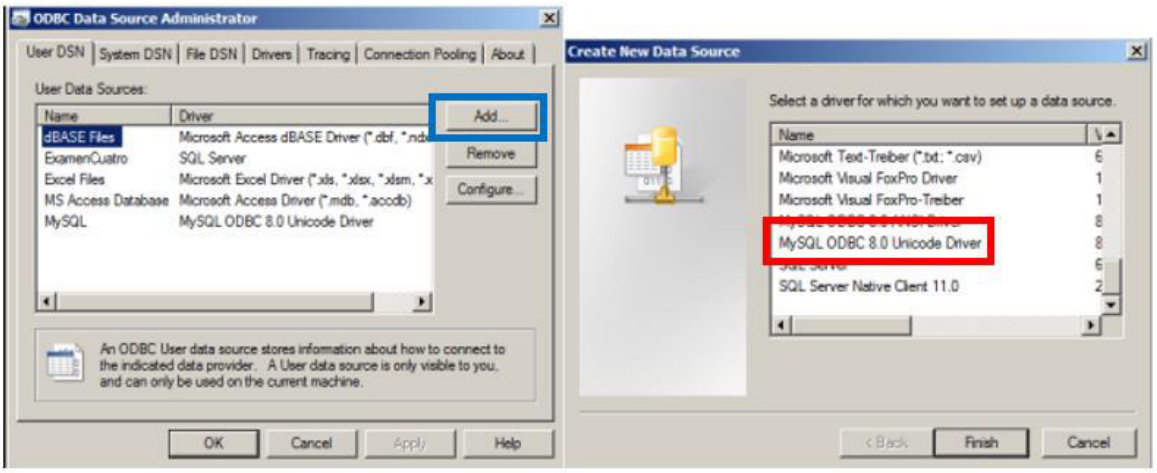


Figura B.6 Pantalla de configuración del administrador ODBC.

4. Terminar el proceso con el botón *finish*.
5. Llenar los parámetros de configuración del conector ODBC, en función de lo que se muestra en la figura B.7.

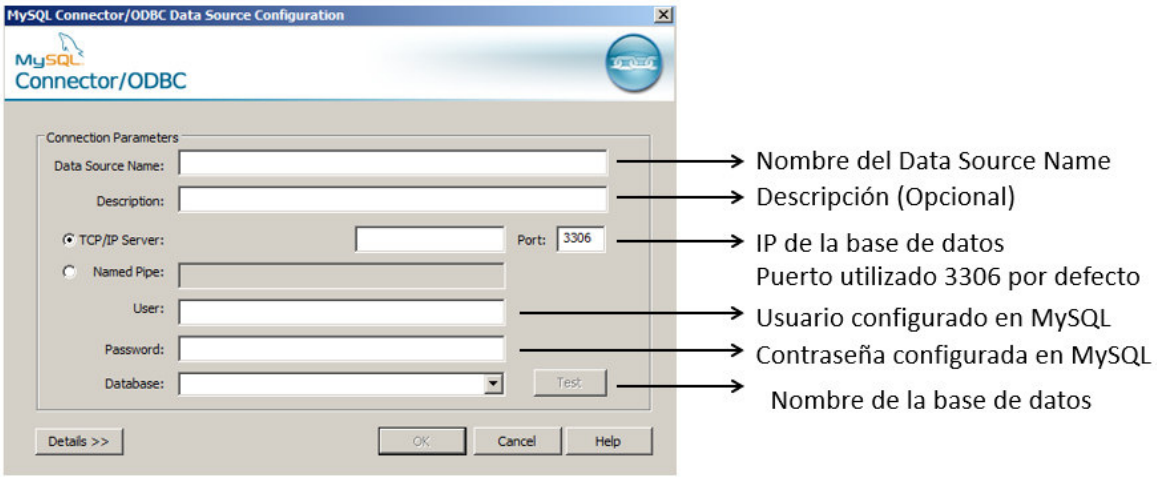


Figura B.7 Pantalla de configuración de ODBC Data Source con sus partes.

B.4 VENTANAS DEL SERVIDOR DE DATOS DESARROLLADO

Las ventanas posibles en total son 7, esto incluyendo a la ventana principal, las pantallas de prueba, de configuración y de usuario.

B.4.1 Ventana Principal

En la figura B.8 se presenta la ventana principal, se observa los botones para acceder a las pantallas de prueba y configuración.

Figura B.8 Ventana principal del prototipo.

Partiendo de esta pantalla se pueden acceder a las ventanas que permiten configurar el driver para un funcionamiento en planta.

B.4.2 Ventanas de Prueba

Las ventanas de prueba se dividen en lectura y escritura, el funcionamiento es similar. Se presenta a continuación ambas pantallas de prueba.

B.4.2.1 Pantalla de Prueba Lectura

La pantalla de prueba de lectura tiene como objetivo únicamente ingresar los puertos para los cuales se programó el envío de la información desde el PLC. De tal forma que se pueda leer la información directamente en el driver, comprobar que está se encuentra correctamente configurada y que no existen problemas en la conexión. Se presenta esta pantalla en la figura 2.24.

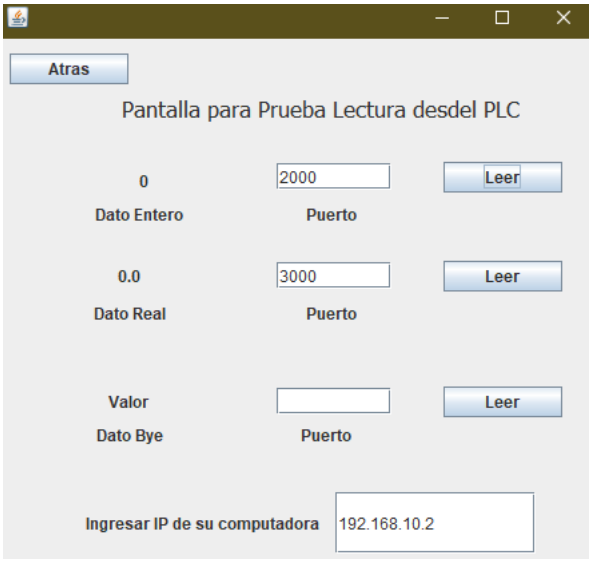


Figura B.9 Ventana de prueba de conexión de lectura del PLC.

Se presenta que la ventana de prueba de lectura tiene una distribución de los elementos bastante simple. Únicamente se encuentran 3 posibilidades de lectura las cuales se realizan una sola vez, sin la existencia de ningún tipo de bucle.

B.4.2.2 Pantalla de Prueba Escritura

La pantalla de prueba de escritura tiene el mismo objetivo que la pantalla de prueba de lectura, el método se basa en la escritura de un dato en el PLC. El método en el driver genera los datos que requieren ser escritos en el PLC dependiendo del proceso de la planta. Para esta conexión el dato debería generarse en la interfaz del operador orientada a su uso específico en la planta, pero debido a que esto no es posible; se ingresa un valor en la ventana de prueba. La pantalla de prueba sirve para verificar la conexión del driver con el PLC, y la base de datos no tiene ninguna influencia en este análisis. Se presenta en la figura B.10 la pantalla de escritura.

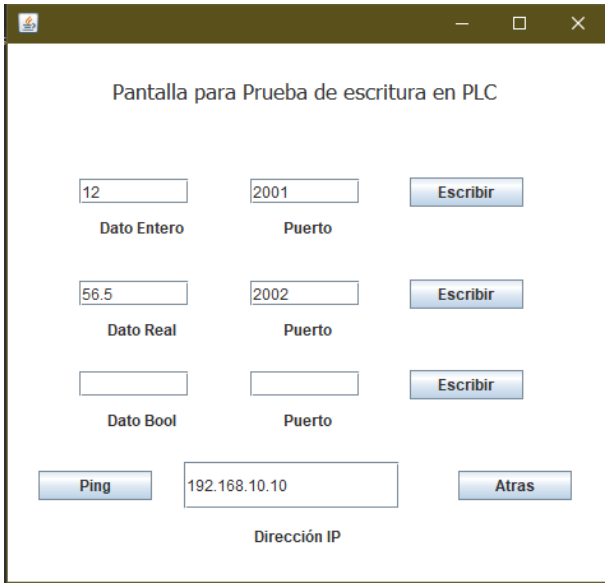


Figura B.10 Ventana de prueba de conexión del PLC.

B.4.3 Ventana de Configuración Escritura

En esta pantalla se configuran los requerimientos y la información necesaria para recibir la información desde la base de datos y posteriormente escribirla en el PLC. Se presenta la pantalla en la figura B.11.

Figura B.11 Ventana de configuración del driver interfaz de escritura en el PLC.

A continuación, se muestra una tabla detallando cada una de las partes de la ventana y el funcionamiento de estas.

Tabla B.1. Elementos Dentro De La Pantalla De Configuración Escritura

Cuadros de Texto o Selección	
Dato (i)	Permite seleccionar si el dato se envía o no.
Puerto	Configuración del puerto a enviar la información
Tipo de Dato	Selecciona el tipo de dato a enviar
Nombre de Columna	Nombre de la columna de donde se saca la información de MySQL
Nombre de Tabla	Nombre de la tabla de donde se saca la información de MySQL
Cantidad de Datos	Selección de cantidad de datos a enviar. Habilita configuración
Tiempo de Loop	Tiempo de repetición de escritura de la información.
Botones	
Atrás	Cierra la venta y muestra de nuevo la pantalla de menú principal
Ping	Permite verificar si existe conexión con la IP del PLC
Pantalla de Usuario	Termina la configuración y abre la pantalla de operador
Encendido	Se enciende el driver, empieza la comunicación con la base de datos y el PLC.
Apagado	Después que se termine el bucle en proceso, se para la comunicación.
Indicadores	
Estado del Servidor	Estado del driver

B.4.4 Ventana de Configuración Lectura

Se establecen aquí las configuraciones requeridas para la puesta en marcha del driver. Esta tiene la misma disposición de los elementos que la pantalla de configuración de escritura, ambas se encuentran organizadas en un formato tipo tabla para facilidad del entendimiento de la información. Se presenta esta en la figura B.12.

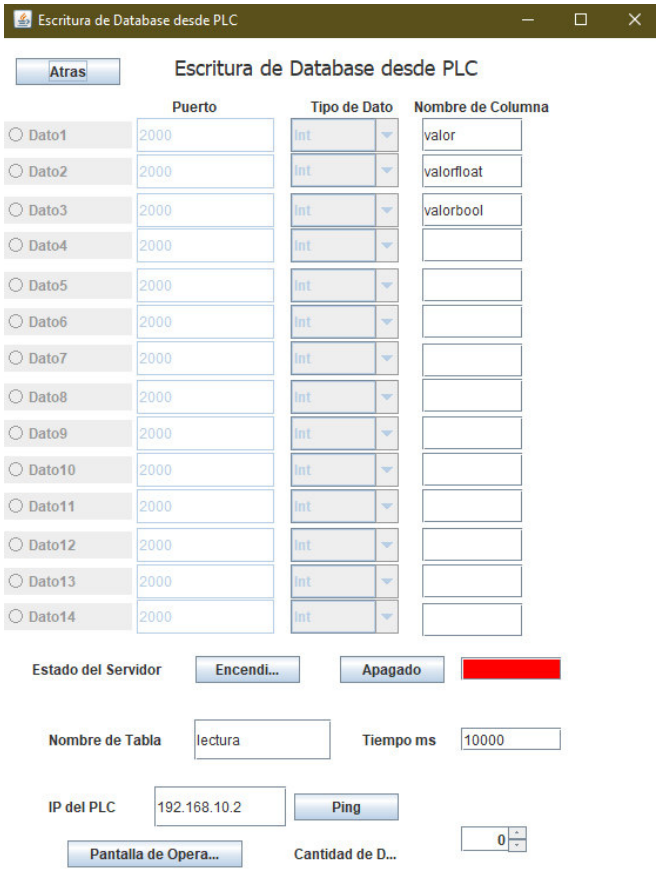


Figura B.12 Ventana de configuración del driver interfaz de lectura en el PLC.

Se presentan los elementos de la pantalla con su respectivo uso en la siguiente tabla.

Tabla B.2. Elementos en la Pantalla de Configuración Lectura

Cuadros de Texto o Selección	
Dato (i)	Permite seleccionar si el dato se envía o no.
Puerto	Configuración del puerto a enviar la información
Tipo de Dato	Selecciona el tipo de dato a enviar

Nombre de Columna	Nombre de la columna donde se escribe la información de MySQL. Nombre con que se creará la columna.
Nombre de Tabla	Nombre de la tabla donde se escribe la información de MySQL
Cantidad de Datos	Selección de cantidad de datos a recibir. Habilita configuración y determina cantidad de columnas en la tabla.
Tiempo de Loop	Tiempo de repetición de recepción de la información.
Botones	
Atrás	Cierra la venta y muestra de nuevo la pantalla de menú principal
Ping	Permite verificar si existe conexión con la IP del PLC
Pantalla de Usuario	Termina la configuración y abre la pantalla de operador
Encendido	Se enciende el driver, empieza la comunicación con la base de datos y el PLC.
Apagado	Después que se termine el bucle en proceso, se para la comunicación.
Indicadores	
Estado del Servidor	Estado del driver

B.4.5 Ventanas de Usuario

Estas pantallas son solamente accesibles después de haber realizado el establecimiento de información necesaria en las pantallas de configuración respectivas, su funcionamiento depende exclusivamente de la correcta puesta en marcha, que vaya acordé con la correcta programación realizada en los controladores. Estas son similares a sus respectivas ventanas de configuración y se observan en la figura B.13.

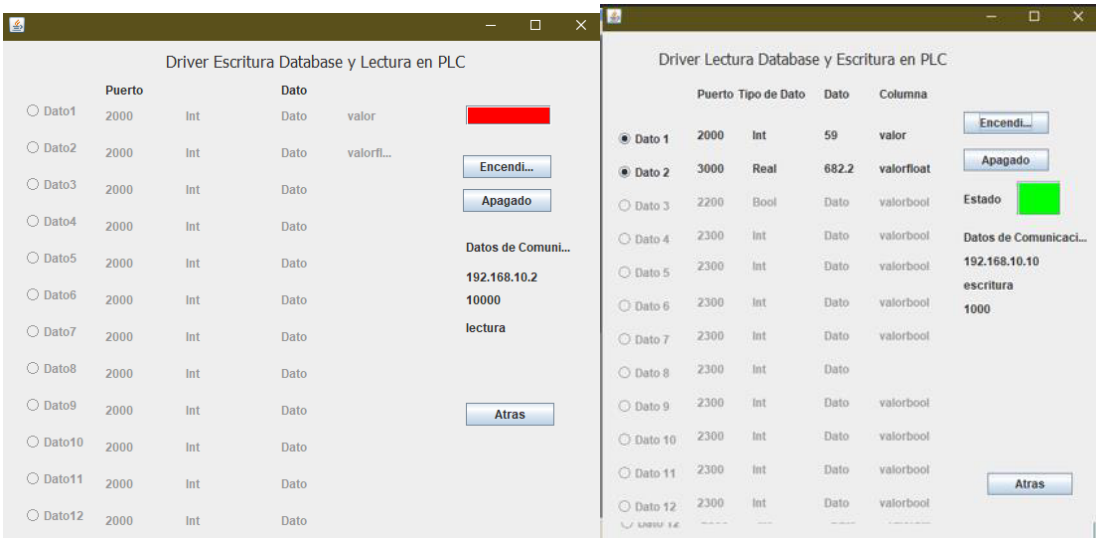


Figura B.13 Ventanas de usuario del driver.

B.5 COMUNICACIÓN DE INTOUCH CON BASE DE DATOS

Para la conexión se considera un botón de inicio que permite escribir y leer datos desde MySQL. Para este botón se considera el siguiente comando:

- SQLConnect (ConnectionId, “Provider=MSDASQL;DSN=MySQL”)

El DSN es el configurado anteriormente en el conector ODBC.

B.5.1 Creación de Una Bind List

El sistema esta orientado a utilizar la herramienta de Intouch de Bind List para comunicación con base de datos. Se crea una Bind List por cada una de las tablas en la base de datos. Se configura un TagName por cada una de las columnas a leer dentro de MySQL, esto se realiza en la siguiente pantalla.

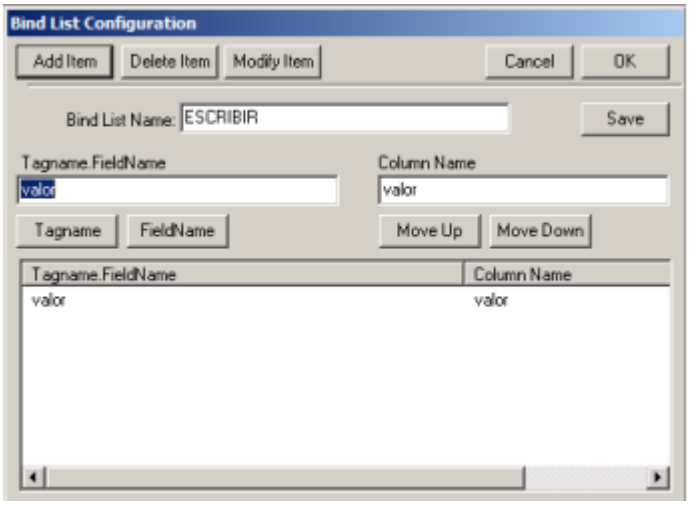


Figura B.14 Ventanas de creación de Bind List.

Se realiza la programación de los botones de escritura y lectura.

Programación botón de escritura:

- SQLInsert(connectionId, "escritura", "lecturadatabasedesdeplc");
- IF ResultCode <> 0 THEN
- Msg = SQLErrorMsg(ResultCode);
- ELSE
- Msg = "Escritura correcta";
- ENDIF;

Programación botón de lectura:

- ResultCode = SQLSelect(connectionId, "lectura", "escrituradatabasedesdeplc", "","");
- ResultCode=SQLLast(connectionId);
- IF ResultCode <> 0 THEN
- Msg = SQLErrorMsg(ResultCode);
- ELSE
- Msg = "Lectura correcta";
- ENDIF;

La programación mostrada esta orientada a trabajar siempre con los datos finales de la tabla de MySQL. Esto en favor de la programación desarrollada en Java.