ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN DRIVER ETHERNET/IP PARA COMUNICACIÓN DE PLCs A INTERFACES DE OPERADOR

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

ALISSON VALERIA MOYA PEÑARRIETA

DIRECTOR: DRA. ING. SILVANA DEL PILAR GAMBOA BENÍTEZ

CODIRECTOR: MBA. ANA VERÓNICA RODAS BENALCÁZAR

Quito, agosto 2022

AVAL

Certificamos que el presente trabajo fue desarrollado por Alisson Valeria Moya Peñarrieta, bajo nuestra supervisión.

Firmado electrónicamente por:
SILVANA DEL PILAR GAMBOA BENITEZ

NOMBRE DIRECTOR

Dra.-Ing. Silvana del Pilar Gamboa Benítez

Firmado electrónicamente por:
ANA VERONICA RODAS BENALCAZAR

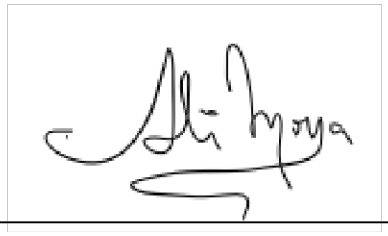
NOMBRE CODIRECTOR

MBA. Ana Verónica Rodas Benalcázar

DECLARACIÓN DE AUTORÍA

Yo, Alisson Valeria Moya Peñarrieta, 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.



ALISSON VALERIA MOYA PEÑARRIETA

DEDICATORIA

Dedico este trabajo con mucho cariño.

A mis padres, Maryury Peñarrieta y Renzo Moya, por su apoyo y enseñanzas.

A mis hermanos, Angie y Renzito, por su amor y confianza en mí.

AGRADECIMIENTO

En primer lugar, quiero agradecer a Dios y a mis padres por su apoyo incondicional en el transcurso de la carrera como también cada una de las enseñanzas de unión familiar y perseverancia que me han impartido a lo largo de la vida para llegar a cumplir las metas que he me propuesto. Gracias a sus consejos y apoyo soy la persona que soy.

Quiero agradecer también a mis hermanos menores que han sido siempre la fuente de energía para continuar día a día, con algún chiste o con alguna anécdota, en los momentos más difíciles me han sacado una sonrisa y me han hecho comprender que cada esfuerzo vale la pena y rinde sus frutos.

Agradezco a mis amigos de la carrera y al grupo de amigos Power Rangers, con los que he aprendido a caer y levantarme, a sonreír y disfrutar, a trabajar en equipo y disfrutar los frutos de una desvelada, les agradezco por empujarme a continuar en los momentos más difíciles, siempre los llevo en mi corazón. Además, quiero agradecer a mis amigas de toda la vida Karlis, Emi, Den, Cris, Karen y Mafer con las que he crecido en lo personal y académico, estoy muy agradecida por esa amistad sincera e incondicional.

Agradezco especialmente a mi mejor amigo en estos años de universidad, Andrés, que me ha brindado su apoyo incondicional y amor en cada momento de mi vida tanto personal como académico. Le agradezco por siempre confiar en mí e impulsarme a ser mejor cada día.

Con mucho cariño agradezco y admiro a mi directora de tesis Dra. Silvana Gamboa, por los consejos, guía, tiempo y paciencia que ha tenido durante la elaboración del presente trabajo, además, agradezco a mi codirectora MBA. Ana Rodas, por su tiempo, experiencia y correcciones realizadas en este proyecto.

Finalmente, agradezco a la Escuela Politécnica Nacional donde he pasado estos últimos años continuamente aprendiendo y forjando mi carácter, como también le agradezco haberme brindado la oportunidad de conocer personas maravillosas que han aportado en mi formación académica y personal.

ÍNDICE DE CONTENIDO

RESUMEN

Para el funcionamiento de sistemas SCADA a nivel industrial se requiere de drivers y servidores que permitan obtener información de los dispositivos de campo que trabajan con distintos protocolos de comunicación industrial como Ethernet/IP. Los softwares comerciales usados en la industria suelen requerir una elevada inversión para pequeñas y medianas empresas, por lo cual, éstas no pueden acceder a los beneficios que brindan los sistemas de supervisión.

En el presente trabajo técnico se diseña, desarrolla e implementa, mediante el lenguaje de programación JAVA y el gestor de base de datos MySQL, un servidor de datos industrial utilizando el protocolo Ethernet/IP.

En el primer capítulo se presenta una introducción, objetivos, alcance y una recopilación de teoría relacionada con el tema de desarrollo del trabajo.

En el segundo capítulo se tiene el análisis, diseño del driver, base de datos e interfaz de usuario, como también se presenta la lógica de programación mediante diagramas de actividades y diagramas de clase.

En el tercer capítulo se presentan los resultados obtenidos en las pruebas realizadas al software desarrollado, la implementación del prototipo con un ejemplo de proceso industrial y la comparación con el software comercial DASABCIP.

Finalmente, en el cuarto capítulo se presentan las conclusiones y recomendaciones del trabajo técnico.

PALABRAS CLAVE: Driver, Ethernet/IP, Comunicación, Protocolo, Java, MySQL, PLC.

ABSTRACT

The operation of SCADA systems at industrial levels, requires drivers to obtain information from field devices that work with different industrial communication protocols such as Ethernet/IP. The commercial software used in the industry usually requires a high investment for small and medium companies, so they cannot access the benefits provided by monitoring systems.

This technical paper designs, develops and implements, using the JAVA programming language and the MySQL database manager, an industrial data server using the Ethernet/IP protocol.

The first chapter presents an introduction, objectives, project scope and a collection of theory related to the topic of development of the work.

In the second chapter, the analysis, driver design, database and user interface are presented, as well as the programming logic through activity diagrams and class diagrams.

The third chapter presents the results obtained in the tests performed on the developed software, the implementation of the prototype with an example of an industrial process and the comparison with the commercial software DASABCIP.

Finally, the fourth chapter presents the conclusions and recommendations of the technical work.

KEYWORDS: Driver, Ethernet/IP, Communication, Protocol, Java, MySQL, PLC.

1. INTRODUCCIÓN

Los sistemas SCADA son una herramienta de automatización y control industrial enfocada a la supervisión para alcanzar la máxima eficiencia en el desempeño de procesos. Estos sistemas, con la ayuda de software de monitoreo y control permiten un gran desarrollo en la industria; sin embargo, los programas enfocados a la ejecución de estas tareas tienen un costo que puede no ser accesible para pequeñas y medianas empresas, debido a que en algunos casos no cuentan con los recursos suficientes para realizar esta inversión.

Por otro lado, en la actualidad el software de código abierto debido a su gran crecimiento brinda una serie de ventajas y posibilidades para el desarrollo de aplicativos sin requerir una gran inversión. Cabe destacar que el software libre garantiza el uso de código informático, copia y modificación con cualquier fin, por lo cual se podrá tener acceso al código para ser estudiado y adaptado según requerimientos, además, se tiene también la libertad de redistribuir copias del software y realizar nuevas versiones con mejoras.

En la actualidad el desarrollo de la Industria 4.0 [1] ha sido más notorio, convirtiéndose en el objetivo de varias industrias. Para lo cual la automatización de procesos requiere de protocolos y estándares que permitan una eficiente comunicación, uno de estos protocolos es Ethernet/IP, desarrollado por Rockwell Automation con el propósito de estandarizar la comunicación en la industria. La automatización tiene efectos positivos, puesto que refleja un incremento en la producción y a su vez reduce los costos de fabricación, dando como resultado productos a precio competitivo.

En el presente proyecto se desarrolla un driver de comunicación que adquiera información de un PLC utilizando el protocolo industrial Ethernet/IP mediante el lenguaje de programación JAVA que brinda la posibilidad de crear pantallas para el manejo del aplicativo, además se almacena la información en la base de datos MySQL para que sea consumida por un software industrial que permita el desarrollo de HMI de proceso como Intouch.

1.1 OBJETIVOS

El objetivo general de este Proyecto Técnico es:

- Desarrollar un driver Ethernet/IP para comunicación de PLCs a interfaces de operador.

Los objetivos específicos del Proyecto Técnico son:

- Realizar una revisión bibliográfica acerca del funcionamiento y las características representativas del protocolo Ethernet/IP.
- Analizar la comunicación de PLCs con protocolo Ethernet/IP con un software de análisis de redes de comunicación para determinar las características y los requerimientos de comunicación de este protocolo.
- Seleccionar el lenguaje de programación en el cual se realizará la implementación del driver de comunicación propuesto.
- Implementar en software de código abierto el driver de comunicación Ethernet/IP y una interfaz de usuario para su configuración.
- Realizar pruebas de comunicación que validen el correcto funcionamiento del driver desarrollado con PLCs que cuenten con protocolo Ethernet/IP e interfaces de operador industrial.

1.2 ALCANCE

En el presente proyecto se estudiará el funcionamiento del protocolo de comunicación Ethernet/IP y se determinará la estructura de trama para realizar funciones de lectura y escritura con datos tipo: booleanos, enteros y flotantes. Para definir la trama Ethernet/IP se utilizará un software especializado en análisis de redes y posteriormente se contrastarán los resultados obtenidos para identificar su estructura. A continuación, se establecerán los parámetros necesarios para realizar la configuración de comunicación Ethernet/IP.

Se estudiarán mecanismos de transferencia de información entre aplicaciones de Windows con el objeto de seleccionar uno de ellos para establecer intercambio de datos entre el driver a desarrollar e interfaces de operador industriales basados en Windows.

Se elaborará un diagrama UML con la lógica del programa y se diseñará la interfaz de configuración del driver de comunicación desarrollado. Se seleccionará el software de código abierto para desarrollar el driver y el mecanismo de comunicación con aplicaciones

de Windows, en el cual también se implementará una interfaz de usuario para configurar la aplicación y lograr la conexión con el PLC.

Se efectuarán pruebas de conexión con el Driver desarrollado y un PLC que cuente con el protocolo de comunicación Ethernet/IP, corroborando que la información enviada o recibida por el Driver con la visualizada en los registros del programa en tiempo real del PLC sea la misma. Posteriormente se realizará la programación del PLC y una interfaz de usuario de un proceso para emular un sistema de control industrial, éste será integrado con el software desarrollado para analizar el funcionamiento en conjunto: software realizado, el PLC con Ethernet/IP y el HMI de monitoreo.

1.3 MARCO TEÓRICO

1.3.1. SISTEMAS SCADA

Los sistemas SCADA (Supervisory Control and Data Acquisition) tienen el objetivo de supervisar procesos para cumplir exigencias de rendimiento y calidad en la industria, además de facilitar las tareas de supervisión al operario. Inicialmente se revisa el concepto de supervisión la cuál según [2] se define como un conjunto de acciones que se realizan por parte del operador con el objetivo de garantizar el funcionamiento correcto de un proceso. La finalidad de la supervisión es la automatización de las tareas a monitorear a través de la recolección de datos y conocimiento del proceso. Los sistemas de supervisión incluyen 3 etapas: detección de fallos, diagnóstico y reconfiguración. En la primera etapa se definen las condiciones que se encuentran fuera de lo normal en el proceso para lo cual se debe conocer el funcionamiento y los parámetros dentro de norma de este. En la segunda etapa el objetivo es encontrar las causas que producen el fallo detectado a través de un análisis deductivo del proceso. Finalmente, en la última etapa se toman las acciones requeridas para que el proceso regrese a condiciones normales [2]. Para el cumplimiento de estas etapas es fundamental la adquisición de datos.

Los sistemas SCADA han tenido gran éxito como sistemas de supervisión, éstos se pueden definir como un conjunto de programas que tienen acceso a los dispositivos que forman parte de la planta como sensores y actuadores, entre sus principales características se encuentran: adquisición de datos, representación visual de variables de proceso, fácil conectividad con otras aplicaciones, control por medio de actuadores y lecturas de sensores. Un aspecto primordial para el funcionamiento de estos sistemas es la adquisición

de información de proceso para lo cual se utiliza software que se comunique, a través de protocolos definidos para la industria, con los elementos de campo (PLCs, sensores, medidores de energía, módulos de control, etc.).

En la Figura 1.1 se muestra la estructura de un sistema SCADA, la misma que está compuesta por:

- Driver de comunicación: se encarga de gestionar la comunicación con los dispositivos de campo y el servidor de datos utilizando protocolos industriales.

- Servidor de datos: se encarga del almacenamiento de datos para su uso y análisis posterior, además tiene acceso directo a las bases de datos.

- Servidor Web: se encarga de gestionar el acceso de datos y su disponibilidad en internet.

- HMI o interface hombre máquina: permite la visualización del estado del proceso mediante el uso de gráficos, objetos animados, textos, listados, tablas, etc.

- OLE/ODBC: interface de Microsoft encargada de la comunicación con bases de datos y otras aplicaciones.

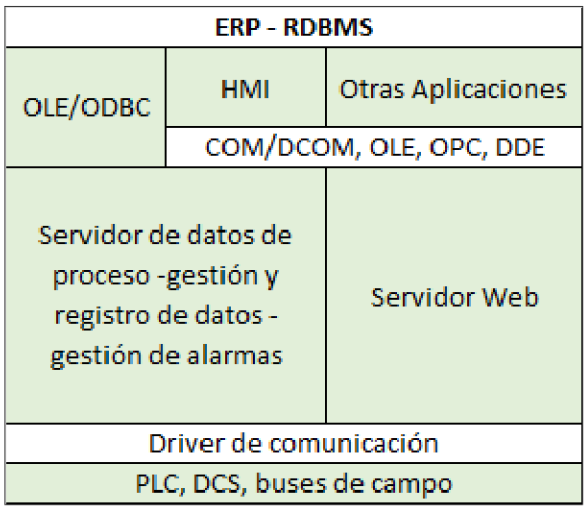


Figura 1.1. Estructura básica de sistemas SCADA [2].

1.3.2. CONCEPTOS DE COMUNICACIÓN

Interface Ethernet

Ethernet es un estándar de red de acceso múltiple con detección de portadora y colisiones que determina el cableado en la capa física del modelo OSI y el formato de tramas en la

capa de enlace de datos. Con respecto a la capa física se diferencian 4 variaciones principales: Ethernet Standard, Fast Ethernet, Gigabit Ethernet y 10 Gigabit Ethernet. En cuanto al formato de trama se tiene Ethernet II, IEEE 802.3, IEEE 802.2 LLC y IEEE 802.2 SNAP [3].

A nivel industrial se utiliza la variación Fast Ethernet y el formato de trama de Ethernet II. Fast Ethernet es de conexión multipunto, comunicación Half-Duplex o Full -Duplex con una velocidad de 10/100MBPS a una distancia de 200 m con cable par trenzado y 2000m con fibra óptica monomodo [4].

La trama Ethernet se compone de preámbulo, dirección MAC de origen y destino, protocolo de red, datos y check sequence, en la Figura 1.2 se observa una trama Ethernet básica dentro de la cual se utiliza TCP/IP o UDP/IP para la transmisión de datos.

Ethernet Frame							
Preamble	Destination MAC Address	Source MAC Address	Type	IP	TCP/UDP	Data	Check Sequence

Figura 1.2. Trama básica del estándar Ethernet [5].

Protocolo de red

Un protocolo de red es un conjunto de reglas que determinan como se llevará a cabo la comunicación entre dispositivos de una misma red. Existen diversos protocolos como: IP, ARP, NDP, ICMP, SNA, NBF, IPX, DDP y OSPF. El protocolo se clasifica según el número de dispositivos, transmisión de datos, jerarquía de los dispositivos, sincronización en la comunicación y tipo de conexión. Uno de los más conocidos es el protocolo de internet IP que se encarga del direccionamiento y enrutamiento de los paquetes de datos [5].

La trama de IP que se observa en Figura 1.3 cuenta con un campo de versión, campo de longitud de cabecera, tipo de servicio, longitud total, ID, flags, fragmentación Offset, tiempo de vida, protocolo, header checksum, dirección IP de origen y destino.

Internet Protocol											
Internet Protocol Version	Header length	Services field	Total length	ID	Flags	Fragment offset	Time to live	Protocol	Header checksum	Source Address	Destination Address

Figura 1.3. Trama de Internet Protocol.

Protocolo de transporte TCP

TCP es un protocolo encargado de controlar, manejar y asegurar la transmisión de datos, cuenta con campos de cabecera para identificar los puertos de origen y destino [6]. Para una comunicación orientada a la conexión en TCP se realiza el hand-shaking para la apertura, siguiendo un protocolo de tres pasos:

1. Se envía un paquete de datos desde el dispositivo A al puerto del dispositivo B con la bandera SYN activada para iniciar una conexión.
2. Si el puerto en B se encuentra abierto, se envía un paquete al puerto en A con las banderas SYN y ACK activadas. ACK (Acknowledgement) se utiliza para confirmar la recepción de un paquete SYN.
3. Al recibir el paquete enviado por el dispositivo B, A envía un último paquete con la bandera ACK activada para confirmar la conexión.

Cuando la conexión se ha establecido se inicia un intercambio de datos hasta que se requiera el cierre de la conexión, para esto cada uno de los dispositivos envía un mensaje para finalizar la conexión. En la Figura 1.4 se observa el intercambio de mensajes para la apertura y cierre de conexión.

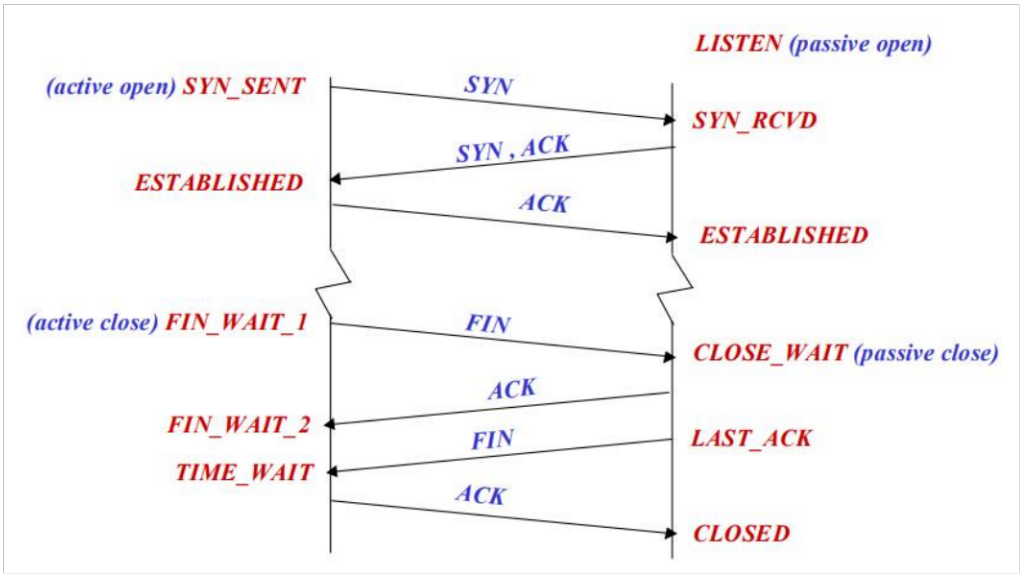


Figura 1.4. Intercambio de mensajes para apertura y cierre de conexión.

La trama de TCP que se visualiza en la Figura 1.5, se compone de puerto de origen y destino, número de secuencia, número de ACK, longitud de la cabecera TCP, banderas, ventana, checksum, urgent pointer y opciones [7].

Transmission Control Protocol									
Source Port	Destination Port	Sequence number	Acknowledgment number	Header length	Flags	Window	Checksum	Urgent Pointer	Options

Figura 1.5. Trama de Transmision Control Protocol.

1.3.3. PROTOCOLO INDUSTRIAL COMÚN CIP

CIP (Common Industrial Protocol) es un protocolo industrial orientado a las capas superiores del modelo OSI el cual utiliza el modelo productor-consumidor para integrar control e información. En el modelo productor-consumidor el productor coloca un mensaje en la red para que pueda ser utilizado por uno o varios consumidores basándose en el ID de conexión del paquete. Para cumplir su objetivo, CIP abarca un amplio conjunto de mensajes y servicios aplicables en la industria como control, seguridad, energía, información y administración de redes. Existen cuatro redes adaptadas para CIP que pueden ser implementadas en las capas inferiores: DeviceNet, ControlNet, CompoNet y EtherNet/IP [8].

CIP utiliza el modelo basado en objetos para describir mediante clases, instancias y atributos el comportamiento de un nodo CIP. Estos objetos y componentes se direccionan bajo un esquema que consta de:

- Dirección de nodo: es la dirección que identifica a cada uno de los nodos de la red CIP, en el caso de Ethernet/IP es la dirección IP.

- Identificador de clase: es el identificador asignado a cada clase dentro de la red.

- Identificador de instancia: es el identificador asignado a cada instancia entre todas las instancias de la misma clase.

- Identificador de atributo: es el identificador asignado a cada atributo dentro de una clase.

- Código de servicio: es un valor que indica una solicitud de acción que puede dirigirse a un objeto o atributo mediante mensajes explícitos.

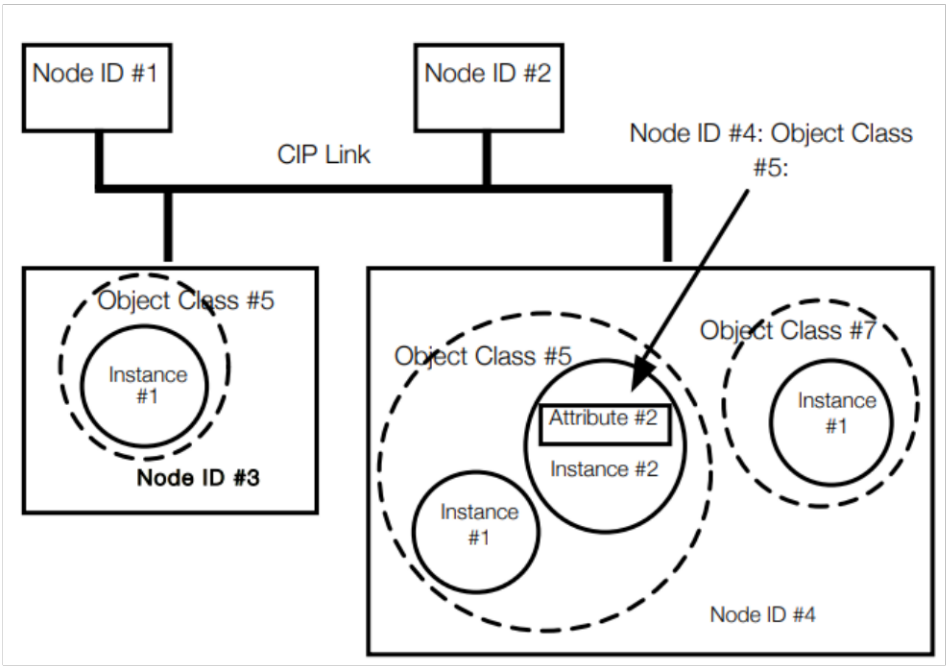


Figura 1.6. Direccionamiento de objetos [9].

Para el intercambio de datos en CIP primero se establece la conexión entre los nodos involucrados y posteriormente se transmiten los datos a través del protocolo de mensajería. La conexión se abre a través de la función *Unconnected Message Manager* UCMM Forward_Open, la cual tiene la siguiente información:

- Time-out
- Network CID origen-destino
- Network CID destino-origen
- Vendor ID y número de serie
- Tamaño máximo de datos en los mensajes
- Unicast o multicast
- Mecanismo de activación
- Ruta de conexión
- Llave electrónica
- Segmento de datos con información de nodos
- Información de enrutamiento

Para la configuración de dispositivos se utilizan hojas de datos electrónicas que contienen información del objeto, en la sección de Connection Manager se tiene básicamente el contenido de un mensaje Forward_Open utilizado para apertura de conexión

Las conexiones CIP se pueden clasificar en dos: mensajería implícita y mensajería explícita [10].

- Mensajería Implícita o I/O: funciona bajo el modelo productor/consumidor para enviar información de entrada y salida en tiempo real, esta mensajería utiliza el protocolo de transporte UDP/IP para una transmisión de datos más rápida.

- Mensajería Explícita: funciona bajo el modelo consulta/respuesta en el cual cada solicitud enviada contiene información específica como parámetros de configuración, utiliza el protocolo de transporte TCP/IP para la transmisión de información que no es crítica en el tiempo.

1.3.3.1 Objetos CIP

La familia CIP tiene una serie de objetos definidos divididos en tres tipos [9]:

- Uso general
- Específico de aplicación
- Específico de red

En el grupo de uso general se tienen las siguientes clases fundamentales en cada dispositivo con sus respectivos atributos, los cuales forman parte de la trama.

Identity Object (Class ID 0X01)

Esta clase describe las características del objeto utilizado para la conexión, todos los dispositivos que formen parte de la red deben tener un Identity Object. Los atributos principales que se requieren en la mayoría de los casos, se detallan a continuación:

- Vendor ID: es un atributo tipo entero que indica el fabricante del dispositivo.
- Device Type: es un atributo tipo entero que indica el tipo de dispositivo a utilizar.
- Product Code: es un atributo tipo entero definido por el fabricante para identificar dispositivos con el mismo Vendor ID y Device Type.

- Status: provee información del estado del dispositivo que podría estar en modo controlado, configurado o posee alguna falla.
- Serial Number: se utiliza para identificar dispositivos junto con el Vendor ID.
- Product name: este atributo permite asignar un código ASCII al dispositivo.

Parameter Object (Class ID 0X0F)

Esta clase permite obtener información del parámetro al cual se quiere acceder en el dispositivo, a continuación, se detallan atributos característicos del objeto parámetro:

- Parameter value: este atributo contiene el valor actual del dato.
- Link Path size: este atributo describe el origen del parameter value.
- Descriptor: se encarga de describir propiedades del parámetro.
- Data type: este atributo describe el tipo de dato.
- Data size: indica el tamaño en bytes del dato.

El manejo de datos se realiza por medio del direccionamiento en base a segmentos, divididos en lógicos y tipos de dato. Los segmentos lógicos permiten direccionar objetos y atributos que se estructuran por ID de clase, instancia y atributo [9].

CIP define un estándar para la entrega de datos bajo la estructura UCMM. El formato del mensaje de solicitud se observa en la Figura 1.7 y el formato de mensaje de respuesta en la Figura 1.8.

Campo	Tipo de dato	Descripción
Service	USINT	Código de servicio de la solicitud
Request Path Size	USINT	Tamaño del request path
Request Path	PATH	Contiene Path segment e instance segment
Request data	ARRAY	Contiene información de los datos enviados como tipo de dato, tamaño del dato, etc

Figura 1.7. Formato mensaje de solicitud.

Campo	Tipo de dato	Descripción
Reply Service	USINT	Código de servicio de respuesta
Reserved	USINT	Debe ser 0
General status	PATH	Código de estado general
Size of additional	ARRAY	Tamaño de additional status
Additional status	ARRAY	Estado adicional
Response data	ARRAY	Datos de respuesta solicitados

Figura 1.8. Formato de mensaje de respuesta.

1.3.4 ETHERNET/IP

Ethernet/IP es un protocolo de comunicación definido a principios de 1998 con el propósito de ser utilizado con el protocolo de capas superiores CIP, se basa en la normativa IEEE 802.3 y trabaja con el estándar Ethernet y TCP/IP para el enlace y transmisión de datos. La Figura 1.9 muestra la arquitectura de red con referencia a las capas del modelo OSI [11].

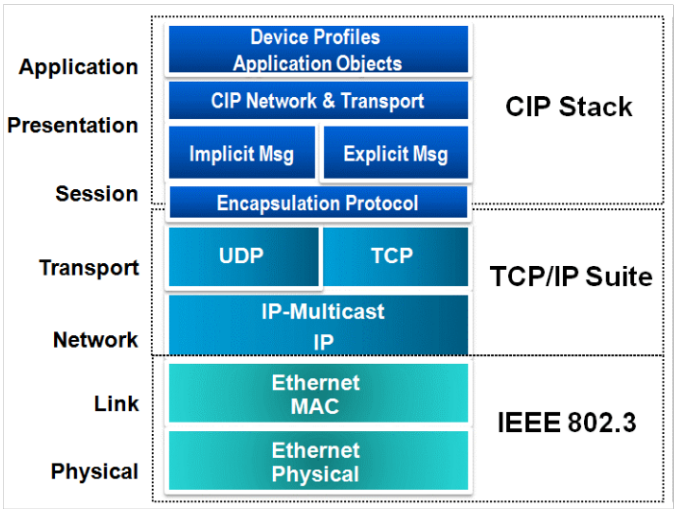


Figura 1.9. Ethernet/IP y CIP en modelo [12].

El paquete de encapsulamiento de Ethernet/IP se compone de una cabecera y el campo de datos como se observa en la Figura 1.10.

En la cabecera se tienen los siguientes campos:

- Command: contiene el comando de encapsulación.
- Length: contiene la longitud en bytes de la cantidad de datos del mensaje.

- Session handle: contiene el identificador de sesión que es generado por el dispositivo de destino en respuesta de un Register Session request.
- Status: contiene el código de estado que indica si el receptor pudo ejecutar el comando de encapsulación solicitado.
- Sender context: contiene información de identificación para el request y el reply.
- Options: contiene el flag de opciones.

En la siguiente sección Command Specific Data se tienen los datos encapsulados con el mensaje requerido. La estructura de este campo depende del command code y se basa en el Common Packet Format.

Encapsulation header						Command specific data
Command	Length	Session handle	Status	Sender Context	Options	Encapsulated data

Figura 1.10. Encapsulamiento de trama Ethernet/IP.

Uno de los códigos para el campo comando es SendRRData que transfiere un paquete encapsulado del tipo solicitud/respuesta entre dispositivo de origen y destino, este comando se utiliza para el envío de mensajes UCMM. En las Figura 1.11 y Figura 1.12 se observa la estructura del paquete utilizando este código de comando. El Common Packet Format contiene en su estructura el item count, address item y data item con su respectivo Type ID, length y data. Cabe recalcar que en el caso de mensajes UCMM estos campos serán cero [11].

Estructura	Campo		Tipo de dato	Descripción
Encapsulation header	Command		UINT	SendRRData 0x6f
	Length		UINT	Tamaño de Command Specific Data
	Session handle		UDINT	Identificador retornado por RegisterSession
	Status		UDINT	0
	Sender context		ARRAY	Valor de 8 bytes
	Options		UDINT	0
Command specific data	Interface handle		UDINT	Toma el valor de 0 para CIP
	Timeout		UINT	Tiempo de espera de la operación
	Encapsulated packet	Item count	UINT	Toma el valor de 2
		Address Type ID	UINT	Toma el valor de 0 para UCMM
		Address Length	UINT	Toma el valor de 0 para UCMM
		Data Type ID	UINT	Toma el valor de 0 para UCMM
		Data Length	UINT	Toma el valor de 0 para UCMM

Figura 1.11. Estructura de mensaje de solicitud UCMM.

Estructura	Campo		Tipo de dato	Descripción
Encapsulation header	Command		UINT	SendRRData 0x6f
	Length		UINT	Tamaño de Command Specific Data
	Session handle		UDINT	Identificador retornado por RegisterSession
	Status		UDINT	0
	Sender context		ARRAY	Identificador del request de 8 bytes
	Options		UDINT	0
Command specific data	Interface handle		UDINT	Toma el valor de 0 para CIP
	Timeout		UINT	No utilizado en trama de respuesta
	Encapsulated packet	Item count	UINT	Toma el valor de 2.
		Address Type ID	UINT	Toma el valor de 0 para UCMM
		Address Length	UINT	Toma el valor de 0 para UCMM
		Data Type ID	UINT	Toma el valor de 0 para UCMM
		Data Length	UINT	Toma el valor de 0 para UCMM

Figura 1.12. Estructura de mensaje de respuesta UCMM.

1.3.5 MANEJO DE SESIÓN

Existen 3 fases en el manejo de sesiones TCP:

Establecimiento de sesión

El dispositivo de origen abre una conexión TCP/IP al dispositivo de destino, posteriormente, el dispositivo de origen envía un comando RegisterSession para que el dispositivo de destino verifique que la versión de protocolo es la misma. En la Figura 1.13 se observa el

formato de un mensaje de solicitud RegisterSession y en la Figura 1.14 se observa el formato de un mensaje de respuesta RegisterSession.

Estructura	Campo	Tipo de dato	Descripción
Encapsulation header	Command	UINT	Register Session 0x65
	Length	UINT	4 bytes
	Session handle	UDINT	0
	Status	UDINT	0
	Sender context	ARRAY	Valor de 8 bytes
	Options	UDINT	0
Command specific data	Protocol version	UINT	Toma el valor de 1
	Option flags	UINT	Toma el valor de 0

Figura 1.13. Solicitud de Register Session.

Estructura	Campo	Tipo de dato	Descripción
Encapsulation header	Command	UINT	Register Session 0x65
	Length	UINT	4 bytes
	Session handle	UDINT	Valor retornado por el dispositivo de destino
	Status	UDINT	0
	Sender context	ARRAY	Valor de 8 bytes
	Options	UDINT	0
Command specific data	Protocol version	UINT	Toma el valor de 1
	Timeout	UINT	Toma el valor de 0

Figura 1.14. Respuesta de Register Session.

El campo Session handle contiene el identificador generado por el dispositivo de destino, el cuál será registrado por el dispositivo de origen para posteriormente insertarlo en la cabecera de los siguientes mensajes CIP. Si el registro de sesión ha sido exitoso el campo Status será 0, caso contrario tomará el valor 0x69.

Desarrollo de sesión

Una vez establecida la sesión, continuará hasta que el dispositivo de origen o destino cierre la conexión TCP, existan problemas con el comando UnRegisterSession o la conexión TCP sea interrumpida.

Cierre de sesión

El dispositivo de origen o destino termina la sesión de dos formas: cierre de la sesión TCP por ambos lados de la conexión o se envía un comando de UnRegisterSession y se espera hasta que se detecte el cierre de conexión TCP.

1.3.6. HERRAMIENTAS Y LIBRERIAS DE DESARROLLO

1.3.6.1. LENGUAJE DE PROGRAMACIÓN JAVA

Java es un lenguaje de programación orientado a objetos que se utiliza para el desarrollo de software, se compone de reglas sintácticas y semánticas que dan significado a sus elementos o expresiones. Este lenguaje fue desarrollado en 1995 por Sun Microsystems [13].

La programación orientada a objetos se define como un conjunto de elementos que interactúan entre sí. Se compone, de clases, objetos, instancias, atributos y métodos. Una clase se define como una plantilla de un objeto con características comunes y se compone de atributos y métodos. Los atributos son las características del objeto y los métodos son funciones que determinan el comportamiento de los objetos.

Una gran ventaja de este lenguaje es que puede ser ejecutado en cualquier sistema operativo utilizando un entorno denominado Java Runtime Environment JRE que le brinda portabilidad al software desarrollado. Existen varios entornos de desarrollo para Java conocidos como IDE (Integrated Development Environment) que brindan un entorno de trabajo, diseño, documentación y verificación de programas, algunos ejemplos son: NetBeans, Eclipse, JBuilder y JCreatorPro [14].

Eclipse es un ambiente de desarrollo integrado para la creación de aplicaciones utilizando el lenguaje de programación JAVA, C/C++, Python, PERL, Ruby, etc. Esta plataforma se compone de plug-ins. Las principales partes de una ventana en Eclipse son: vistas, editores, barra de menú y barra de herramientas [15].

1.3.6.2. LIBRERÍAS

- Java Swing

Java Swing es una librería que forma parte de Java Foundation Classes que es utilizada para la creación de aplicaciones basadas en ventanas ya que brinda una serie de herramientas para la construcción de interfaces gráficas de usuario. Una gran ventaja de esta librería es que puede ser utilizada en cualquier plataforma. En la **Figura 1.15** se observa la jerarquía de la API swing de Java que muestra los elementos principales que se utilizan para la construcción interfaces [16].

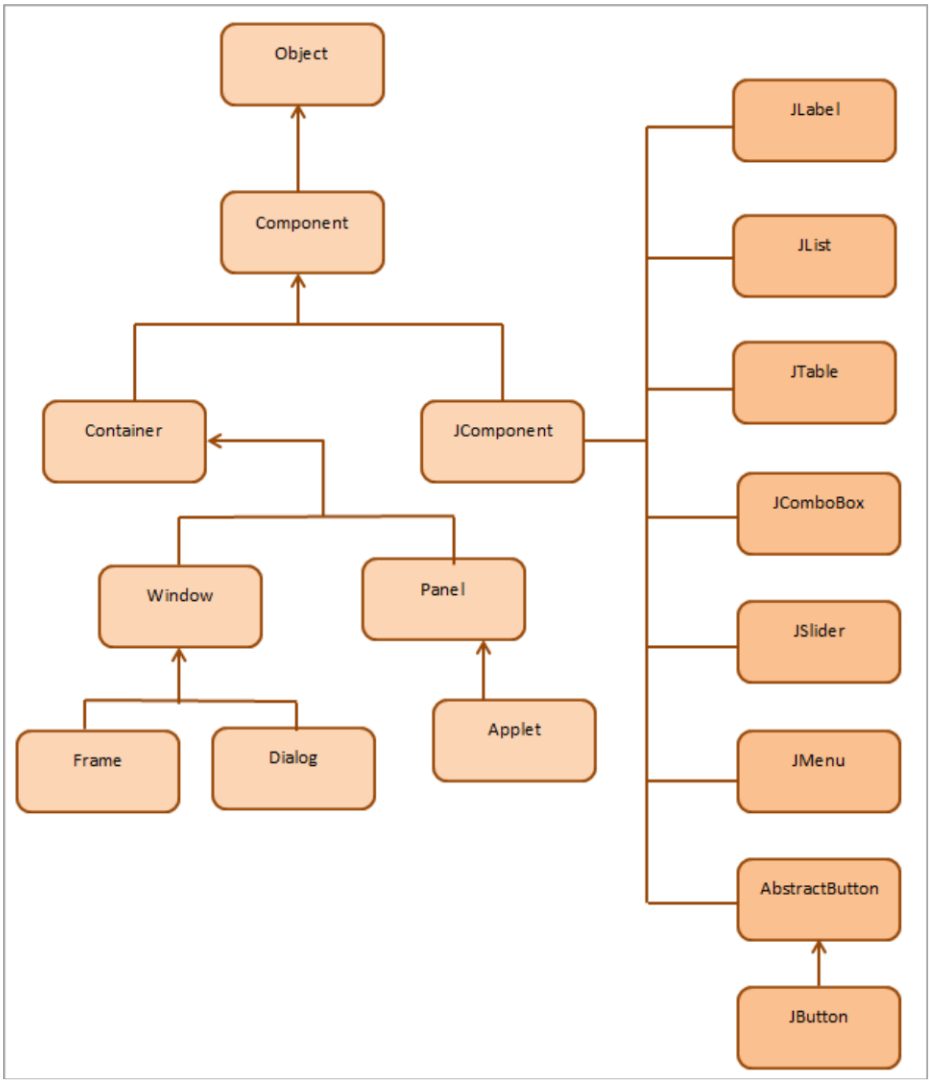


Figura 1.15. Jerarquía de API de swing de java.

- Swing Worker

Al desarrollar interfaces gráficas en Java se tiene un hilo especial llamado EDT (Event Dispatch Thread) que se encarga de manejar eventos de teclado, mouse y ventana. En este hilo se llevan a cabo todas las tareas relacionadas con la interfaz, por tanto, cualquier tarea que requiera un amplio tiempo en realizarse no puede ser ejecutada en este hilo. Si se desarrollan largas tareas la interface gráfica de usuario se queda congelada esperando el fin de la operación [17]. Por esta razón, éstas tareas deben ser ejecutadas en un hilo diferente utilizando SwingWorker, el uso de esta herramienta cumple los siguientes pasos:

1. Creación de variable tipo SwingWorker: *SwingWorker<Void, ArrayList<String>> workerLectura = new SwingWorker<Void, ArrayList<String>>()*
2. Orden de ejecución de hilo: *workerLectura().execute()*

3. Ejecución de hilo:

```
protected Void doInBackground() throws Exception {
        System.out.println("Hilo workerLectura en ejecución");
        return null;
}
```

4. Cancelación de ejecución: *workerLectura.cancel(true)*

- Libplctag

Libplctag [18] es una librería de código abierto, que permite establecer una conexión entre dispositivos mediante el protocolo de comunicación Ethernet/IP con el objetivo de realizar funciones de lectura y escritura en dispositivos lógicos programables. La librería fue diseñada en lenguaje de programación C, pero puede ser utilizada en JAVA, C++, Python, Go y Pascal mediante “wrappers”.

Un “wrapper” es un programa o código que “envuelve” otros componentes de programa mediante el uso de patrones estructurales que funcionan independientemente del lenguaje de programación, logrando interoperabilidad entre sistemas [19].

Para realizar la comunicación con el PLC inicialmente se crea el tag path para lo cual se debe tener conocimiento del protocolo que se requiere utilizar y los argumentos que necesita. Esta función crea un identificador de etiqueta en un tag ya existente en el PLC, los atributos principales que se requieren para la comunicación se detallan a continuación:

- Protocolo: indica el tipo de protocolo que se utilizará.
- Gateway: indica la dirección IP del PLC.
- PLC: indica el tipo de PLC de la familia Allen Bradley que se empleará.
- Tamaño del tag: indica el número de bytes por tag.
- Nombre del tag: indica el tag de destino, por ejemplo, B3:0/0, N7:0 o F8:0.

Los tags creados utilizan recursos internos en la memoria los cuales se deben liberar una vez se ha obtenido la información requerida, para esta tarea se utilizan métodos de cierre propios de la librería.

- Java JDBC

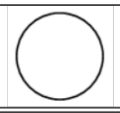
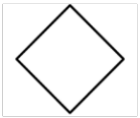
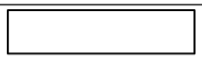
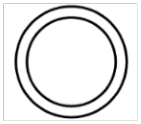
La librería JDBC gestiona la conexión, permite el acceso y procesamiento de información en la base de datos, por lo cual requiere la siguiente información:

- URL: este campo detalla el puerto por el cual se puede acceder a la conexión creada en MYSQL y el nombre de la base de datos.
- Nombre de usuario: indica el nombre de usuario creado en la conexión.
- Contraseña: indica la contraseña de la conexión creada.

1.3.6.3. Diagrama de actividades

Un diagrama de actividades es similar a un diagrama de flujo, éste muestra la secuencia de acciones que se realizan durante una actividad, operación o proceso. Se compone de los nodos que se visualizan en la Tabla 1.1 [20].

Tabla 1.1. Nodos de diagrama de actividades.

Símbolo	Nodo
	Inicial
	Acción
	Decisión
	Unión / Bifurcación
	Final

1.3.6.4. Diagrama de clases

Un diagrama UML es una representación gráfica estática utilizada para visualizar los componentes que forman parte de un sistema y su relación entre sí. Utiliza una figura rectangular conocida como clasificador que denota cada una de las clases que forman parte del sistema. El clasificador se compone de tres secciones, la primera contiene el

nombre de la clase, la segunda los atributos y la tercera los métodos que pertenecen a la clase [21].

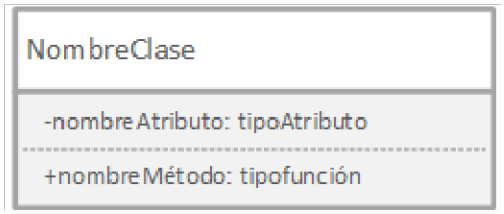


Figura 1.16. Clasificador de diagrama de clases.

La viñeta que acompaña a los componentes representa la visibilidad la cual indica el tipo de acceso del atributo o método. En la Tabla 1.2 se observa el símbolo y el acceso que representa.

Tabla 1.2. Visibilidad de métodos y atributos.

Símbolo	Acceso
+	Público
-	Privado
#	Protegido
~	Paquete

Cada una de las clases tiene una relación de dependencia con otra. En la Tabla 1.3 se observan las relaciones existentes entre clases.

Tabla 1.3. Relaciones de dependencia entre clases.

Símbolo	Relación
	Herencia
	Asociación
	Agregación
	Composición

1.3.7. BASE DE DATOS

Una base de datos es un conjunto organizado de información relacionada entre sí, la cual es almacenada y consultada sistemáticamente. Los gestores de base de datos (DBMS) permiten a los usuarios manejar y manipular esta información, así como también gestionar el acceso a la base de datos a varios usuarios [22]. Algunos ejemplos de gestores de bases de datos son MySQL, PostgreSQL, Microsoft Access, SQL server y Oracle que se utilizan según el modelo de base de datos y los requerimientos de la aplicación [23].

Los modelos de base de datos pueden ser:

- Relacional: se organiza la información mediante el uso de tablas, donde las columnas representan los campos y las filas los registros. Cada tabla cuenta con un identificador o llave primaria que facilita la realización de consultas SQL.
- Distribuida: se almacena la información en diferentes computadoras que se encuentran en red.
- Jerárquica: los datos se organizan en forma de árbol y permite la repetición de información y relación padre e hijo.
- Gráfica: utiliza estructuras de gráficos con nodos y aristas para almacenar y manejar la información.
- NoSQL: tiene estructuras que permiten almacenar información evitando problemas de escalabilidad y rendimiento.

1.3.7.1. MySQL

Es un gestor de base de datos relacionales libre y de código abierto que utiliza el lenguaje estándar de consulta estructurado SQL para el manejo de base de datos relacionales, éste permite realizar funciones como: añadir, buscar, actualizar y eliminar información [24]. El esquema de funcionamiento se muestra en la Figura 1.17.

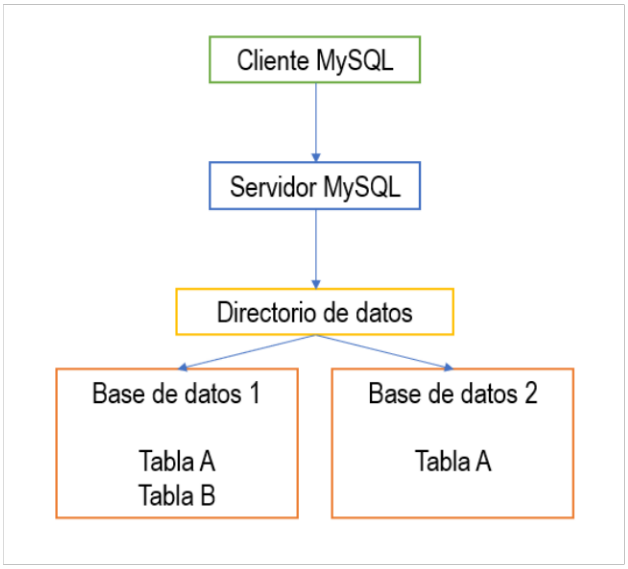


Figura 1.17. Esquema de funcionamiento MySQL.

SQL funciona con sentencias que indican la acción a realizar, a continuación, se enumeran algunas funciones básicas para el manejo de tablas en MySQL [25]. Cabe mencionar que *columna1, columna2, columna3* representan los nombres de cada columna o campo e *idRegistro* representa el identificador del registro a modificar.

- Lectura de tabla completa.

"SELECT *columna1, columna2* FROM *nombreTabla*"

- Lectura de un registro de la tabla.

"SELECT *columna2, columna3* FROM *nombreTabla* WHERE *columna1=idRegistro*"

- Insertar registro a una tabla.

"INSERT INTO *nombreTabla* (*columna1, columna2*) VALUES(*valor1, valor2*)"

- Modificar elementos de una tabla.

"UPDATE *nombreTabla* SET *columna2='25'* WHERE *columna1=idRegistro* "

- Eliminar registro de una tabla.

"DELETE FROM *nombreTabla* WHERE *columna1=idRegistro*"

- Crear tabla en la base de datos.

"CREATE TABLE IF NOT EXISTS *nombreTabla* (id INT NOT NULL AUTO_INCREMENT, *columna* DATETIME, PRIMARY KEY (id))"

Entre paréntesis se colocan los campos de la tabla con sus características y tipo de dato.

- Añadir columnas en determinada tabla.

"ALTER TABLE *nombreTabla* ADD COLUMN *columna1* INT"

La sentencia agrega una columna con el nombre *columna1* con el tipo de dato que se escriba a continuación, en este caso es tipo entero.

- Crear evento que se ejecute cada cierto tiempo.

"CREATE EVENT IF NOT EXISTS *nombreEvento* ON SCHEDULE EVERY #*num* MINUTE ON COMPLETION PRESERVE ENABLE DO *acción*"

El evento se ejecuta cada #num de minutos y ejecuta las sentencias que se coloquen a continuación de ENABLE DO.

- Eliminar evento.

"DROP EVENT IF EXISTS *nombreEvento*"

- Eliminar registros en función del tiempo.

"DELETE FROM *nombreTabla* WHERE *columnaTiempo* < DATE_SUB(NOW(), INTERVAL)"

La sentencia elimina registros con fechas menores a la diferencia de la fecha actual y un intervalo determinado, si el intervalo se define en dos horas se eliminarán los registros con más de dos horas de antigüedad.

1.3.7.2. Diagrama entidad relación

Un diagrama de entidad relación es una herramienta que sirve para representar visualmente la relación de datos entre sí [26]. Sus componentes se visualizan en la Tabla 1.4.

Tabla 1.4. Elementos de un diagrama entidad relación.

Símbolo	Nombre	Descripción
	Entidad	Hace referencia a un objeto. Ejemplo: alumno, auto, empresa.
	Relación	Hace referencia a la relación entre las entidades.

	Unión	Representa la unión entre entidades.
	Atributo	Hace referencia a una característica de la entidad. Ejemplo: nombre, edad, peso.
	Clave primaria	Hace referencia a una clave única para cada entidad.
	Clave foránea	Hace referencia a una entidad externa.

1.3.7.3. Normalización

La normalización de base de datos es un método utilizado para la simplificación de datos identificando relaciones y dependencias entre atributos. Su propósito es eliminar repetición de datos, mantener información ordenada, utilizar un menor espacio de almacenamiento y evitar errores lógicos. Existen niveles de normalización: primera forma normal, segunda forma normal, tercera forma normal, forma normal boyce Codd, cuarta forma normal y quinta forma normal. Una normalización hasta el tercer nivel presenta una base de datos ordenada sin datos repetidos y espacio optimizado, a continuación, se detallan y ejemplifican los tres primeros niveles de normalización [24].

- Primera forma normal: se identifica el grupo de repetición para un mismo registro y se divide en dos tablas relacionadas entre sí.

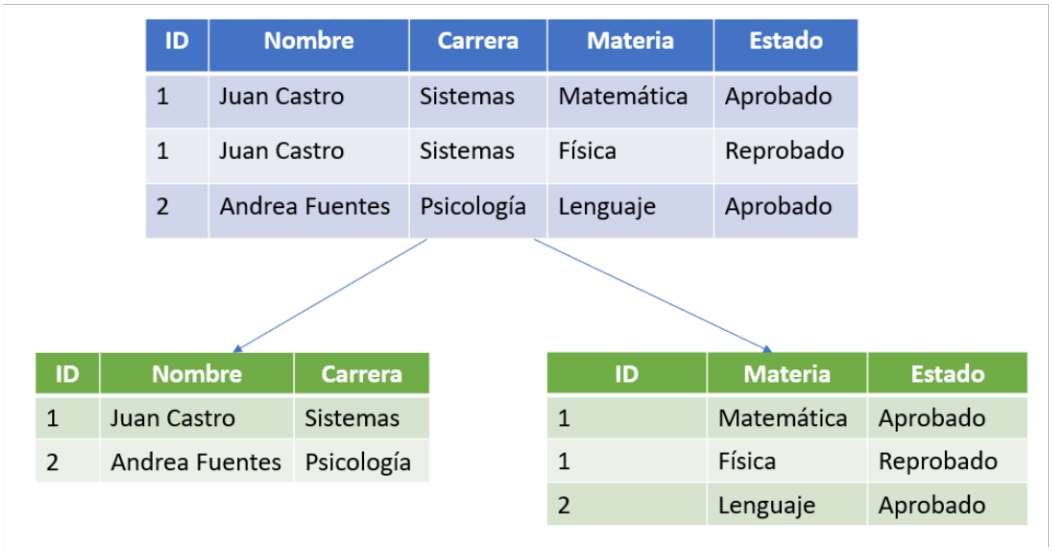
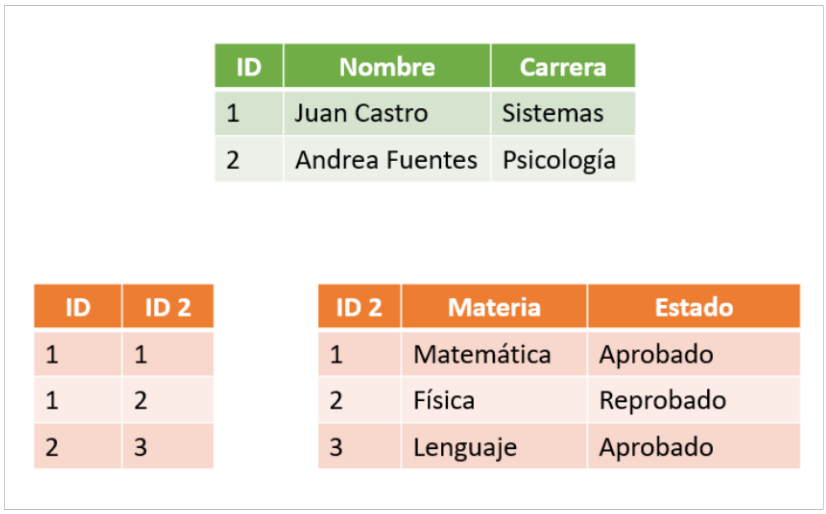


Figura 1.18. Ejemplo normalización 1FN.

- Segunda forma normal: después de tener la tabla en la primera forma normal se identifican las dependencias funcionales y transitivas. En la dependencia funcional se tiene una dependencia directa y en la transitiva es indirecta.



ID	Nombre	Carrera
1	Juan Castro	Sistemas
2	Andrea Fuentes	Psicología

ID	ID 2
1	1
1	2
2	3

ID 2	Materia	Estado
1	Matemática	Aprobado
2	Física	Reprobado
3	Lenguaje	Aprobado

Figura 1.19. Ejemplo normalización 2FN.

- Tercera forma normal: su objetivo es eliminar relaciones funcionales entre campos y verificar una relación funcional de cada campo con la clave primaria.

ID	Nombre	Carrera
1	Juan Castro	Sistemas
2	Andrea Fuentes	Psicología

ID	ID 2
1	1
1	2
2	3

ID 2	Materia	Estado
1	Matemática	1
2	Física	2
3	Lenguaje	1

ID 3	Estado
1	Aprobado
2	Reprobado

Figura 1.20. Ejemplo normalización 3FN.

1.3.8. Servidor de Allen Bradley CIP

El servidor de Allen Bradley CIP de Wonderware es un servidor de datos industrial comercial para Microsoft Windows que permite acceder a la información de PLCs de la familia Allen Bradley. El servidor se conecta a los controladores a través de la red Ethernet/IP, la cual es una red industrial abierta que utiliza el protocolo CIP [27].

El servidor se ejecuta en el System Management Console en la sección de servidores de datos DAServer Manager y se identifica con el nombre Archestra.DASABCIP. En la **Figura 1.21** se observa la creación del puerto CIP en el cual se crearán los objetos que describen a los PLCs que se conectan con el driver.

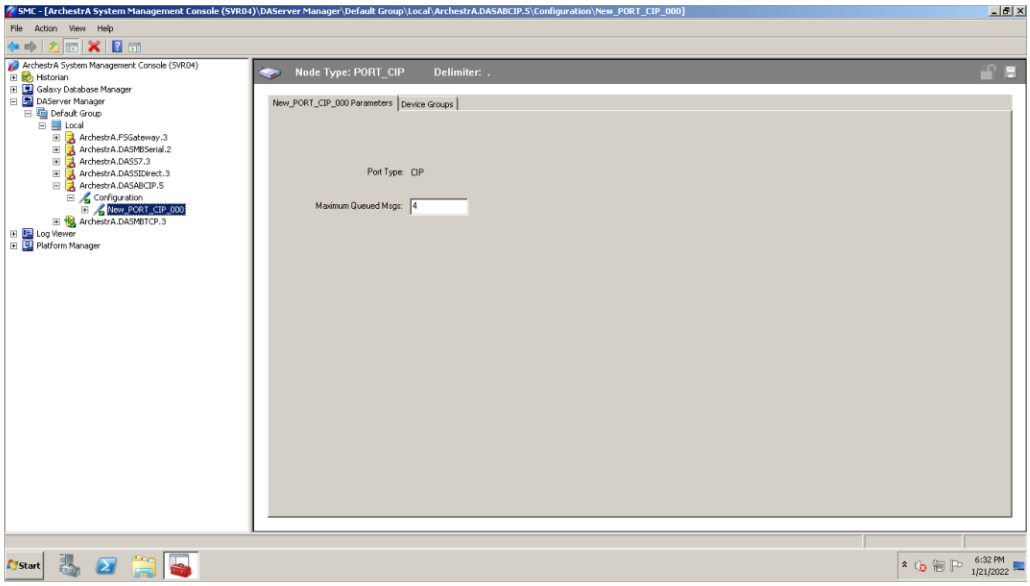


Figura 1.21. Pantalla de puerto CIP.

Al crear el objeto se habilita la configuración que solicita la dirección IP del controlador, tiempo de respuesta y tiempo de espera de la conexión.

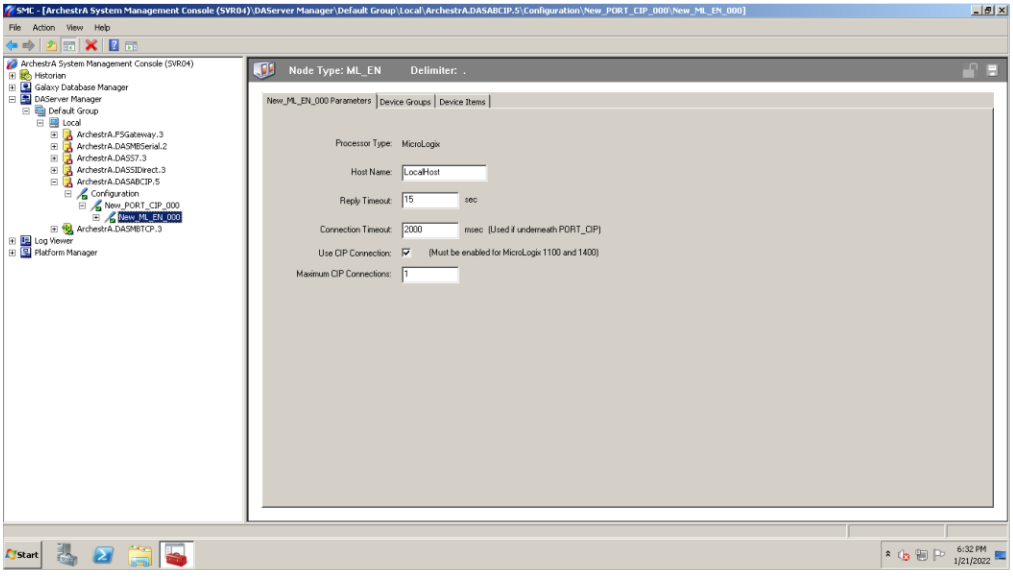


Figura 1.22. Configuración de conexión para Micrologix.

2. METODOLOGÍA

En este capítulo se analiza la trama de Ethernet/IP para determinar los parámetros requeridos para realizar tareas de lectura y escritura en el PLC, posteriormente se diseña el driver de comunicación en base a los requerimientos planteados, una vez se ha obtenido la información se construye el algoritmo de conexión a la base de datos en MySQL para el almacenamiento y consulta sistemática de datos. Se realiza también el diseño y normalización de la base de datos y se diseña una interfaz de manejo y visualización para el software desarrollado en base a los requerimientos planteados. El funcionamiento general del programa se detalla en la lógica de programación haciendo uso de diagramas de actividades UML y se visualiza la estructura estática del sistema en los diagramas de clase UML.

2.1. ANÁLISIS DE TRÁFICO DE DATOS

Inicialmente se realizan pruebas de lectura y escritura entre dos PLCs que cuenten con el protocolo de comunicación Ethernet/IP para identificar las tramas de comunicación mediante un software especializado en redes de comunicación. Los resultados obtenidos indican que primero se establece la conexión entre ambos PLCs realizando el handshaking como se detalla en la sección 1.3.2 del Capítulo 1, posteriormente desde uno de los PLCs se envía la instrucción RegisterSession con el objetivo de iniciar la sesión y obtener el Session Handle para las conexiones de intercambio de datos. Una vez establecida la sesión se envía una solicitud de lectura/escritura bajo el protocolo de Ethernet/IP. En la Tabla 2.1 se observa la lógica del intercambio de datos.

Tabla 2.1. Lógica de intercambio de paquetes de datos entre dos PLCs.

Origen	Destino	Protocolo	Descripción
192.168.1.12	192.168.1.11	TCP	Apertura de conexión
192.168.1.11	192.168.1.12	TCP	
192.168.1.12	192.1681.11	TCP	
192.168.1.12	192.168.1.11	ENIP (EtherNet/IP)	Solicitud de session

192.168.1.11	192.168.1.12	ENIP (EtherNet/IP)	Respuesta a solicitud de sesión
192.168.1.12	192.168.1.11	CIP Connection Manager	Solicitud de apertura de conexión CIP
192.168.1.11	192.168.1.12	CIP Connection Manager	Respuesta a solicitud de apertura CIP
192.168.1.12	192.168.1.11	CIP Ethernet/IP	Solicitud lectura/escritura
192.168.1.11	192.168.1.12	CIP Ethernet/IP	Respuesta a solicitud lectura/escritura
192.168.1.12	192.168.1.11	TCP	Cierre de conexión
192.168.1.11	192.168.1.12	TCP	
192.168.1.12	192.168.1.11	TCP	

A continuación, se detalla la estructura de cada uno de los paquetes de datos que se observan en la tabla anterior.

Para los paquetes de datos con protocolo TCP se cuenta con la estructura que se observa en la Figura 2.1.

Ethernet II	Internet Protocol	TCP

Figura 2.1. Estructura de paquete de datos TCP.

En la estructura de Ethernet II se identifican los campos de la Figura 2.2, que básicamente son las direcciones MAC de origen/destino y la versión del protocolo de red:

ETHERNET II => 14			
# Bytes	6 Bytes	6 Bytes	2 Bytes
Campo	Destination	Source	Type
Valor	00:0f:73:00:a4:6b	00:0f:73:01:90:7c	IPV4

Figura 2.2. Estructura de Ethernet II identificada.

En la Figura 2.3 se observa la trama de Internet Protocol con sus correspondientes valores, los campos principales con las direcciones IP de origen/destino y el protocolo de transporte:

Internet Protocol											
# Bytes	1.00		1.00	2.00	2.00	2.00		1.00	1.00	2.00	4.00	4.00
Campo	Internet Protocol Version	Header length (bytes)	Differentiated Services field	Total length	Identification	Flags	Fragment offset	Time to live	Protocol	Header checksum	Source Address	Destination Address
Valor	4	20	0	126	4	0	0	128	6(TCP)	0XB70E	192.168.1.12	192.168.1.11

Figura 2.3. Estructura de trama de protocolo de red.

En la Figura 2.4 se observa la estructura de la trama TCP donde se define el puerto origen/destino y los números de secuencia y reconocimiento:

Transmission Control Protocol									
# Bytes	2.00	2.00	4.00	4.00	1.00	2.00	2.00	2.00	2.00
Campo	Source Port	Destination Port	Sequence number	Acknowledgment number	Header length (bytes)	Flags	Window	Checksum	Urgent Pointer
Valor	30000	41818	2014071080	1563486759	20	0x18	2000	0xaf89	0

Figura 2.4. Estructura de trama TCP.

Para los paquetes de datos tipo ENIP – EtherNet Industrial Protocol se tiene la estructura de la Figura 2.5.

Ethernet II	Internet Protocol	TCP	Ethernet/IP

Figura 2.5. Estructura de paquete de datos ENIP.

La estructura del paquete de datos ENIP para solicitar apertura de sesión se observa en la Figura 2.6, cabe destacar que en la solicitud se tiene un Session Handle nulo. La respuesta ante la solicitud realizada se observa en la Figura 2.7 la cual es muy similar a la solicitud ENIP, con excepción del campo Session Handle que cuenta con un valor de 0x9535bd5b, que será utilizado como identificador de sesión en los siguientes paquetes de intercambio de datos.

Ethernet/IP								
	Encapsulation header						Command specific data	
# Bytes	2 bytes	2 bytes	4 bytes	4 bytes	8 bytes	4 bytes	2 bytes	2 bytes
Campo	Command	Length	Session handle	Status	Sender Context	Options	Protocol Version	Option Flags
Valor	0x65	4	0x00	0x00	0x00	0x00	1	0x00

Figura 2.6. Estructura Ethernet/IP para paquete de datos ENIP- solicitud.

Ethernet/IP								
	Encapsulation header						Command specific data	
# Bytes	2 bytes	2 bytes	4 bytes	4 bytes	8 bytes	4 bytes	2 bytes	2 bytes
Campo	Command	Length	Session handle	Status	Sender Context	Options	Protocol Version	Option Flags
Valor	0x65	4	0x9535bd5b	0x00	0x00	0x00	1	0x00

Figura 2.7. Estructura de Ethernet/IP para paquete de datos ENIP-respuesta.

La estructura básica del paquete de datos CIP Connection Manager y CIP-Ethernet/IP se observa en la Figura 2.8.

Ethernet II	Internet Protocol	TCP	Ethernet/IP	CIP

Figura 2.8. Campos del protocolo identificados.

La estructura de Ethernet/IP en el paquete de datos CIP Connection Manager se observa en la Figura 2.9, en esta trama se especifica el comando SendRRData utilizado para modelos solicitud/respuesta y se utiliza el valor Session Handle obtenido en la respuesta ENIP de apertura de sesión. Por otro lado, en la sección CIP de la Figura 2.10 se tiene información de gran importancia para la conexión como el Vendor ID, Connection Serial Number, Connection Time, Originator Serial Number y Network Connection.

	Ethernet/IP						Ethernet/IP						
	Encapsulation header						Command specific data						
# Bytes	2 bytes	2 bytes	4 bytes	4 bytes	8 bytes	4 bytes	4 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes
Campo	Command	Length	Session handle	Status	Sender Context	Options	Interface handle	Timeout	Item count	Type ID: Connected Addres Item	Length	UnConnected Data Item	Length
Valor	Send RR Data (0x6f)	62	0x9535bd5b	0x00	c0a8010c00000100	0x00	0x00	1024	2	0	0	0x00b2	46

Figura 2.9. Estructura Ethernet/IP para un paquete de datos Connection Manager.

	Common Industrial Protocol							Command Specific Data					
# Bytes	1 bit	7 bits	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	4 bytes	4 bytes	2 bytes	2 bytes
Campo	Request/ Response	Service	Request Path Size	Path Class Segment	Class	Path Instance Segment	Instance	Priority/Tick time	Time-out	Network connection ID O/D	Network Connection ID D/O	Connection Serial Number	Originator Vendor ID
Valor	Request(0)	0x54	2	0x20	Connection M. 0x06	Logical(1)	0x01	0x0a	0x0e	0x11cad72c	0x97d8e4cc	0xb30d	0x01

Command Specific Data												
4 bytes	1 byte	3 bytes	4 bytes	2 bytes	4 bytes	2 bytes	1 byte	1 byte	2 bytes	1 byte	2 bytes	1 byte
Originator Serial Number	Connection Time Multiplier	Reserved	RPI O/D	Network Connection Parameters O/D	RPI D/O	Network Connection Parameters D/O	Transport type/trigger	Connection Path Size	Class Segment	Class	Instance Segment	Instance
0x7300a46b	0x01	0x00	0x1f8240	0X4302	0x1f8240	0X4302	0xa3	2	0x20	0x02	0x24	0x01

Figura 2.10. Estructura CIP-solicitud para paquete de datos Connection Manager.

	Common Industrial Protocol				Command Specific Data								
# Bytes	1 bit	7 bits	1 byte	1 byte	4 bytes	4 bytes	2 bytes	2 bytes	4 bytes	4 bytes	4 bytes	1 bytes	1 bytes
Campo	Request/ Response	Service	Status	Aditional status size	Network connection ID O/D	Network Connection ID D/O	Connection Serial Number	Originator Vendor ID	Originator Serial Number	O/D API	D/O API	Application Reply Size	Reserved
Valor	Response(1)	0x54	Success (0)	0	0x5bbd0b57	0x97d8e4cc	0xb30d	0x01	0x7300a46b	0x1f8240	0x1f8240	0	0

Figura 2.11. Estructura CIP-respuesta para paquete de datos Connection Manager.

La estructura de Ethernet/IP en el paquete de datos CIP-Ethernet/IP para realizar la lectura de un tag se observa en la **Figura 2.12**. La trama utiliza el comando Send Unit Data para el encapsulamiento de datos conectados, el valor de Session Handle se mantiene y el direccionamiento de los paquetes encapsulados se realiza considerando el tipo “Datos conectados”.

	Ethernet/IP														
	Encapsulation header						Command specific data								
# Bytes	2 bytes	2 bytes	4 bytes	4 bytes	8 bytes	4 bytes	4 bytes	2 bytes	2 bytes	2 bytes	2 bytes	4 bytes	2 bytes	2 bytes	2 bytes
Campo	Command	Length	Session handle	Status	Sender Context	Options	Interface handle	Timeout	Item count	Type ID: Connected Addres Item	Length	Connection ID	Connected Data Item	Length	CIP Sequence Count
Valor	Send UnitData(0x70)	45	0x9535bd5b	0x00	c0a8010c00000100	0x00	0x00	1024	2	0x00a1	4	0x5bbd0b57	0x00b1	25	1

Figura 2.12. Estructura de Ethernet/IP para solicitud de lectura.

En la **Figura 2.13** se observa la estructura CIP para lectura de un tag, se compone de Vendor ID y el Serial Number del dispositivo que genera la solicitud, además se especifica el código de función que indica la acción a realizar 0xa2 para lectura y 0xaa para escritura, el tamaño del dato, el número de registro, el tipo de dato y la dirección dentro del registro. Para la solicitud de escritura se utiliza la misma estructura cambiando el código de función y añadiendo al final 2 bytes con el dato que se quiere escribir.

	CIP									
	Common Industrial Protocol							PCCC Command Data		
# Bytes	1 bit	7 bits	1 byte	1 byte	1 byte	2 bytes	1 byte	1 byte	2 bytes	4 bytes
Campo	Request/Response	Service	Path Size	Path Class Segment	Class	Path Instance Segment	Class	Requestor ID Length	Requestor Vendor ID	Requestor Serial Number
Valor	Request(0)	0x4b	2	0x20	0x67	0x24	0x01	0x07	0x01	0x7300a46b

CIP								
PCCC Command Data								
1 byte	1 byte	2 bytes	1 byte	1 byte	1 byte	2 bytes	1 byte	1 byte
Command code	Status	Transaction Code	Function Code	Byte size	File number	File Type Integer 0x89 Bool 0x85 Flotante 0x8a	Element number	Sub-element Number
0x0f	0x00	0x001	0xa2 (read)	0x02	0x07	0x89	0x00	0x00

Figura 2.13. Estructura CIP para solicitud de lectura.

La estructura de Ethernet/IP de la solicitud se mantiene en la respuesta considerando un Timeout nulo. La respuesta CIP con el dato requerido en la solicitud de lectura se observa en la Figura 2.14. Para la respuesta a una solicitud de escritura se tienen los mismos campos a excepción del campo Data que se suprime.

	CIP										
	Common Industrial Protocol				PCCC Response Data						
# Bytes	1 bit	7 bits	1 byte	1 byte	1 byte	2 bytes	4 bytes	1 byte	1 byte	2 bytes	2 bytes
Campo	Request/ Response	Service	Status	Aditional status size	Requestor ID Length	Requestor Vendor ID	Requestor Serial Number	Response code	Status	Transaction Code	Data
Valor	Response(1)	0x4b	Success (0)	0	0x07	0x01	0x7300a46b	0x4f	0x01	0x001	0x1f40

Figura 2.14. Estructura CIP en respuesta a solicitud de lectura.

2.2. DISEÑO DE DRIVER DE COMUNICACIÓN

La pequeña y mediana industria requiere software de fácil acceso para la supervisión y control de procesos, por tanto, en el presente trabajo se propone el desarrollo de un driver de comunicación con un protocolo industrial que permita almacenar información obtenida de dispositivos lógicos programables para ser centralizada y visualizada en una interfaz de usuario. En consecuencia, primero se definen los requerimientos del software para realizar un diseño inicial y posteriormente determinar la lógica de programación.

2.2.1. Requerimientos

Para el diseño del driver de comunicación se definen los requerimientos necesarios para su funcionamiento a nivel de industria.

- Desarrollo del programa en software de código abierto para garantizar un fácil acceso.
- Utilización de un protocolo de comunicación industrial como Ethernet/IP.
- Interfaz que permita la configuración y visualización del intercambio de datos entre interfaces de usuario de procesos y dispositivos lógicos programables.
- Almacenamiento de información en una base de datos.
- Visualización en una interfaz de usuario-operador de los datos recolectados por el driver.
- Baja inversión económica para ser accesible para la pequeña y mediana industria.
- Funcionamiento continuo tanto para lectura y escritura.
- Capacidad máxima de 16 PLCs y 32 tags por PLC.
- Tiempo de muestreo específico para cada PLC.
- Registro de históricos minuto a minuto durante las últimas 24 horas.

2.2.2. Diseño

Para el cumplimiento de los requerimientos planteados se eligió el software libre JAVA que tiene ventajas de gran utilidad para el presente proyecto como por ejemplo ser un sistema multiplataforma, permitir programación orientada a objetos, fácil desarrollo de interfaces gráficas y extensa bibliografía y documentación. La programación utilizando el protocolo

de comunicación Ethernet/IP para obtener los datos de dispositivos lógicos programables se desarrolló en JAVA. Posteriormente, estos datos se almacenan con la ayuda del gestor de base de datos MySQL, cabe recalcar que es un software de código abierto por tanto se adapta a los requerimientos de este proyecto. Finalmente, la información recolectada estará disponible para ser leída desde la base de datos por software de visualización HMI como Intouch. En la Figura 2.15 se observa un esquema de funcionamiento de hardware y software.

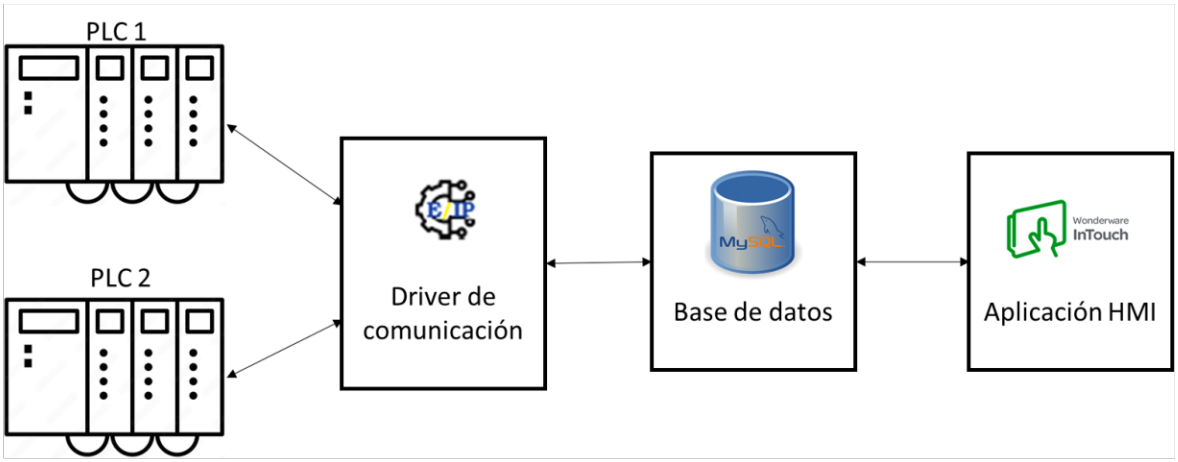


Figura 2.15. Esquema de funcionamiento.

2.2.3. Lógica de programación

Se utilizó programación orientada a objetos para el diseño de esta aplicación para simplificar el manejo de datos mediante el uso de clases, atributos y métodos.

Inicialmente se debe obtener información del PLC por tanto se creó una sección que permita la administración de los dispositivos lógicos programables considerando las funciones de agregar, modificar y eliminar. La información requerida para cada PLC es: nombre, dirección IP, device group y tiempo de ciclo. El nombre y device group permite identificar al dispositivo, la dirección IP posibilita la comunicación con el dispositivo y el tiempo de ciclo determina cada cuanto tiempo se realiza la adquisición de datos desde el PLC. Una vez se ha realizado la validación de la información ingresada y se ha ejecutado la acción agregar, modificar o eliminar, se guardan los datos en la sección de configuración general la cual centraliza la información y se encarga de actualizar todas las clases que requieran de la lista de PLCs y sus atributos.

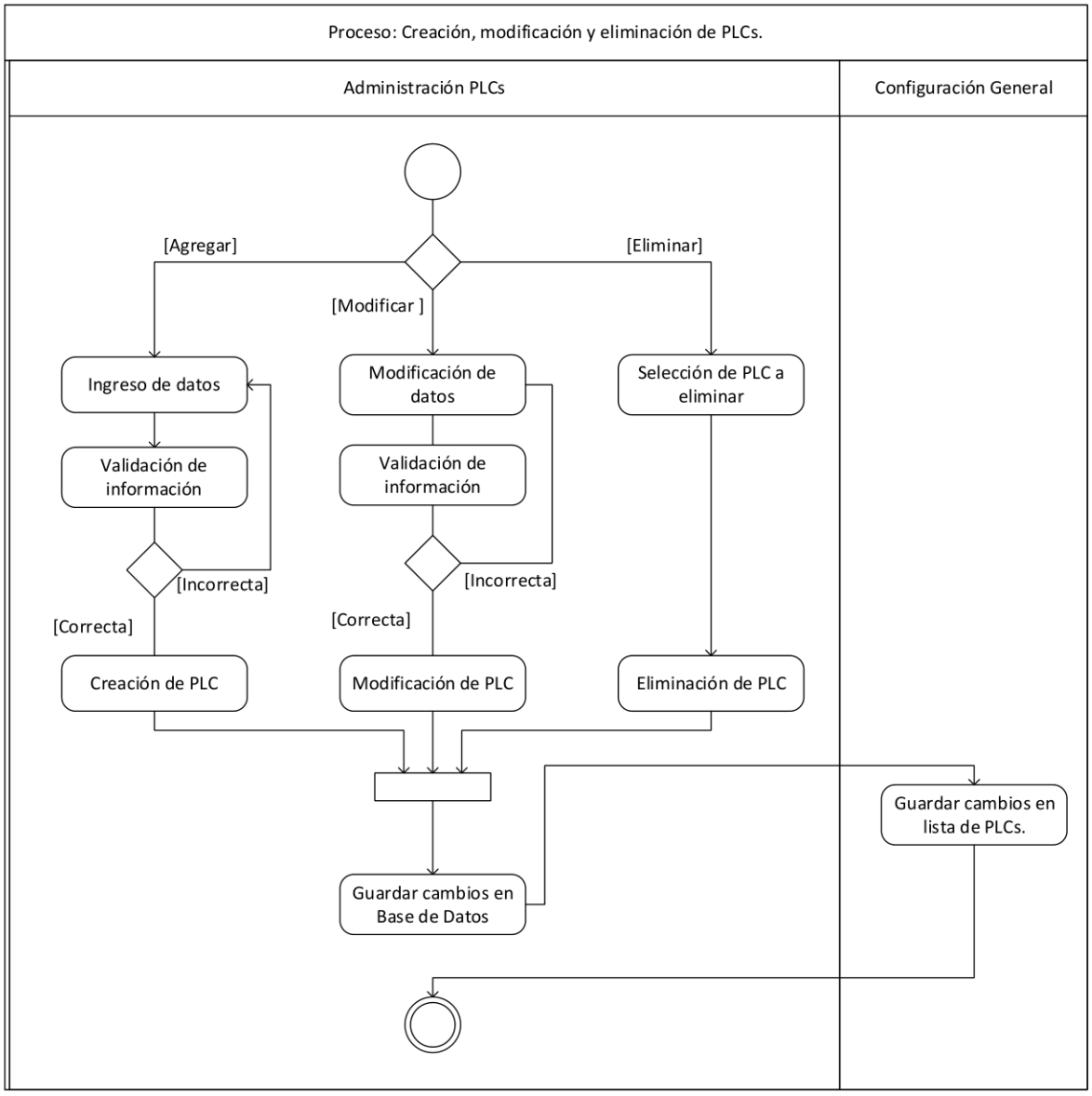


Figura 2.16. Flujo de actividades para creación, modificación y eliminación de PLCs.

Una vez registrados los PLCs se añaden o eliminan tags a cada dispositivo, para lo cual se selecciona el PLC en el cual se realizarán los cambios, posteriormente se ingresan los datos de cada tag que incluye: nombre, dirección en el PLC, tipo de dato, acción de lectura o escritura y si requiere o no registro histórico. La dirección y el tipo de dato permiten obtener el valor del tag especificado, por tanto, se valida la información ingresada comprobando que no existan duplicados del tag y que la dirección sea válida para el tipo de dato, finalmente se guardan los cambios realizados en el objeto y base de datos. En la Figura 2.17 se observa el flujo de actividades para realizar funciones de creación y eliminación de tags.

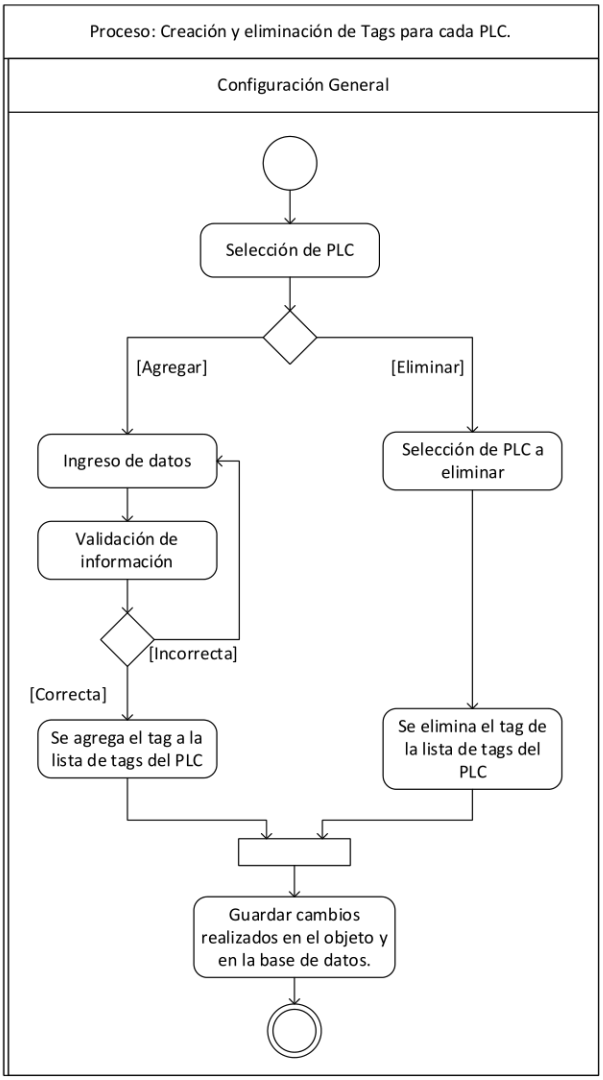


Figura 2.17. Flujo de actividades para creación y eliminación de tags.

Cuando ya se tiene registrada toda la información requerida para la conexión del computador con el PLC, se podrán realizar funciones de lectura/escritura en modo automático o manual. En la Figura 2.18 se observa el flujo de actividades para realizar un intercambio de datos automático. Inicialmente se recibe una lista actualizada de PLCs con todos sus atributos desde la sección de Configuración General, posteriormente se activa el servidor que ejecuta las acciones de apertura de conexión, lectura y escritura que se realizan desde la sección de Conexión PLC. En la rama de lectura se compara el valor leído con el valor guardado, si el dato ha cambiado se actualiza la base de datos, por otro lado, en la rama de escritura se comprueba si el valor leído desde la base de datos ha cambiado para posteriormente escribir el nuevo valor del tag en el PLC. Las acciones de lectura y escritura en la base de datos se realizan en la sección de Administración de base de datos.

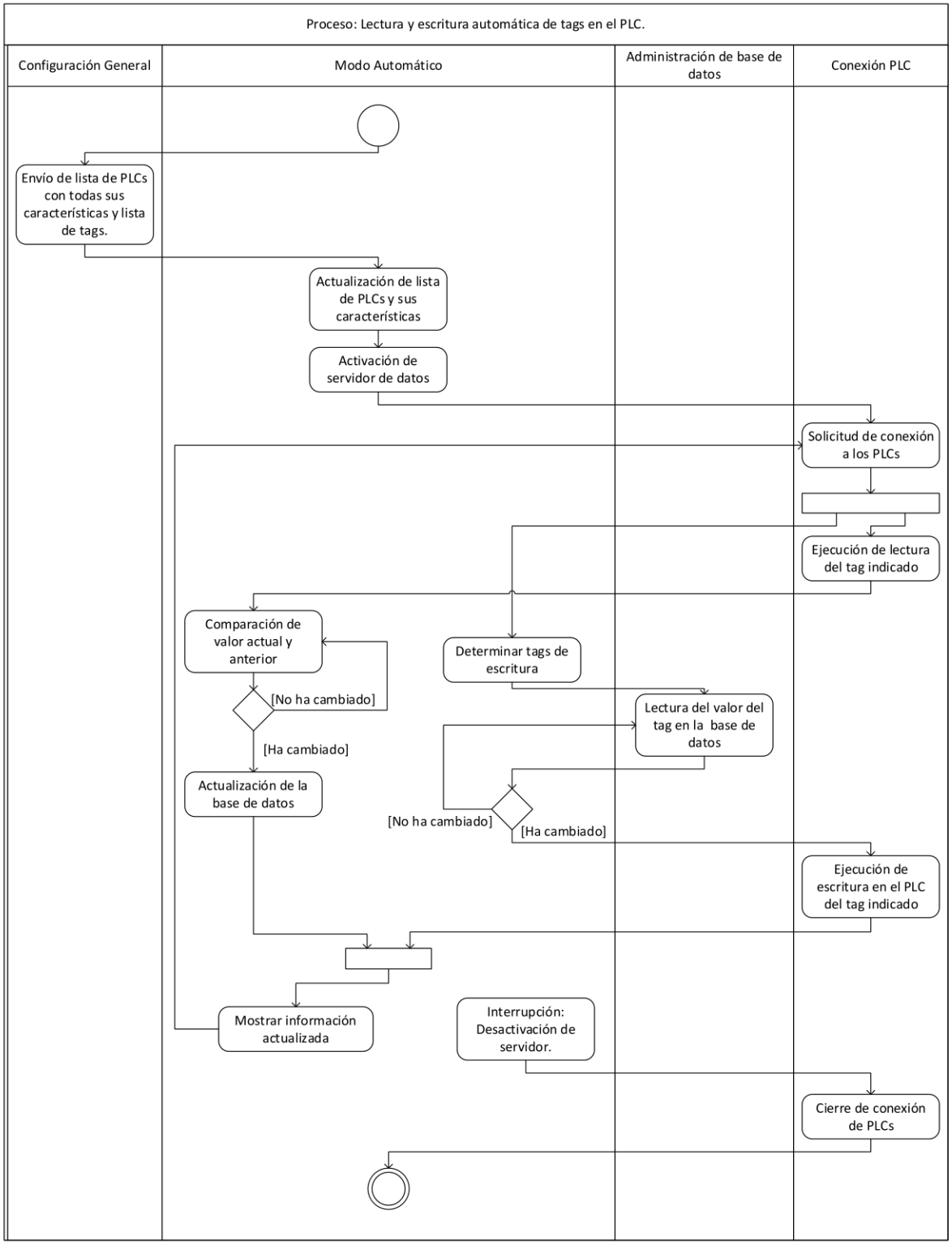


Figura 2.18. Flujo de actividades para lectura y escritura automática.

El modo de escritura manual inicia de la misma forma que el modo automático, la diferencia radica en el origen del dato para la escritura. En el modo manual el valor se toma directamente de la sección de modo manual, mientras en el modo automático el dato se adquiere cíclicamente desde la base de datos, donde previamente fue almacenado por un

software externo de visualización de HMI de procesos. En la Figura 2.19 se observa el flujo de actividades para lectura y escritura manual.

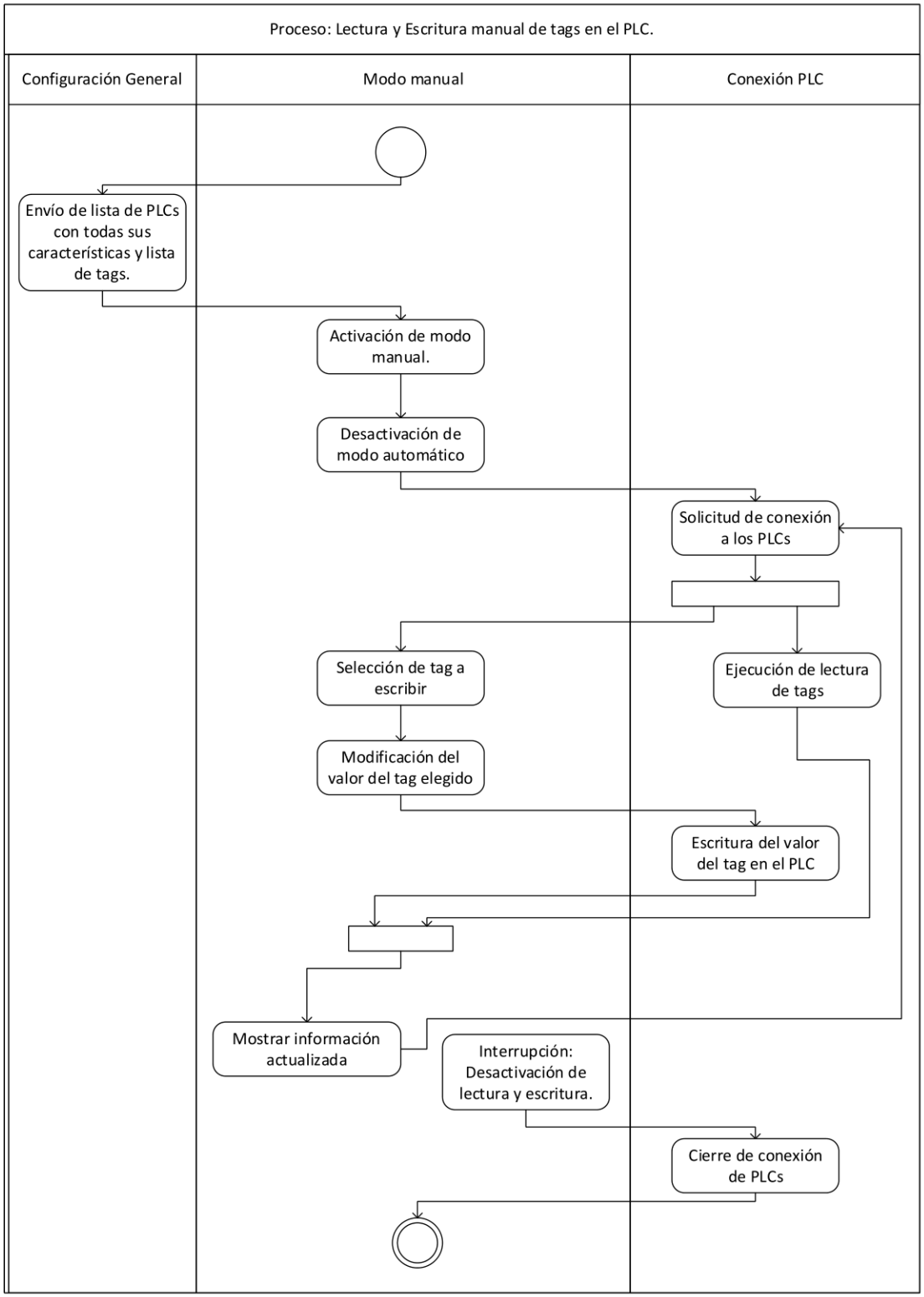


Figura 2.19. Flujo de actividades para lectura y escritura manual.

El registro de históricos se realiza cada minuto almacenando las últimas 24 horas, esto se logra enviando inicialmente la lista de PLCs a la sección Históricos y activando la escritura de los tags históricos en la base de datos, a su vez se crea el evento que se encarga de la eliminación de registros cuando su antigüedad es superior a las 24 horas. El almacenamiento de datos históricos se detiene cuando se desactiva el servidor o se ha presentado algún error en la conexión con los PLCs.

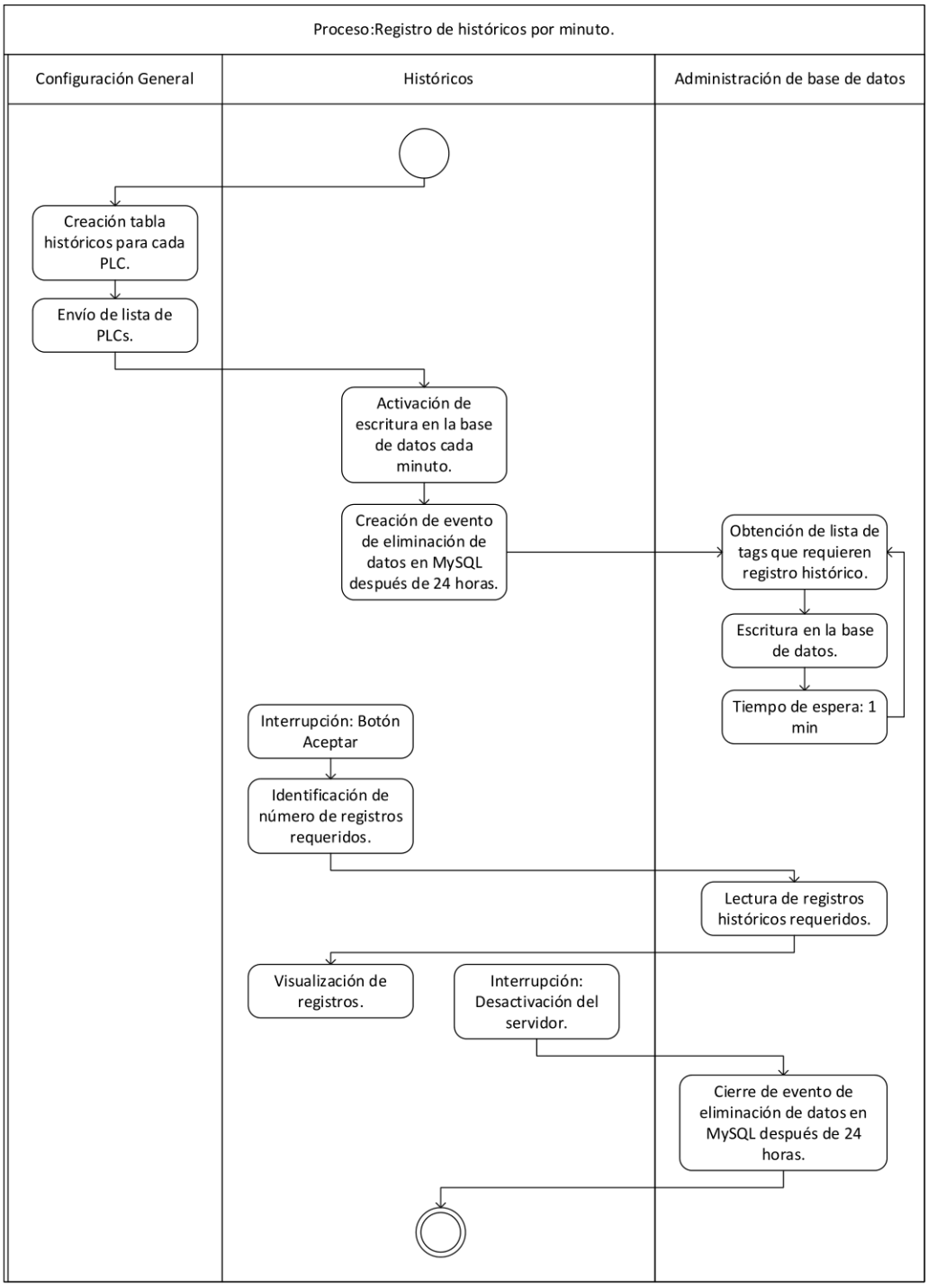


Figura 2.20. Diagrama de actividades para manejo de históricos.

El almacenamiento de información en la base de datos es de gran importancia por tanto se requiere de una sección que permita administrar los parámetros requeridos para la conexión a la base de datos y el esquema con las tablas que se utilizan para almacenar la información del software. En la Figura 2.21 se observa el flujo de actividades para la configuración de la base de datos.

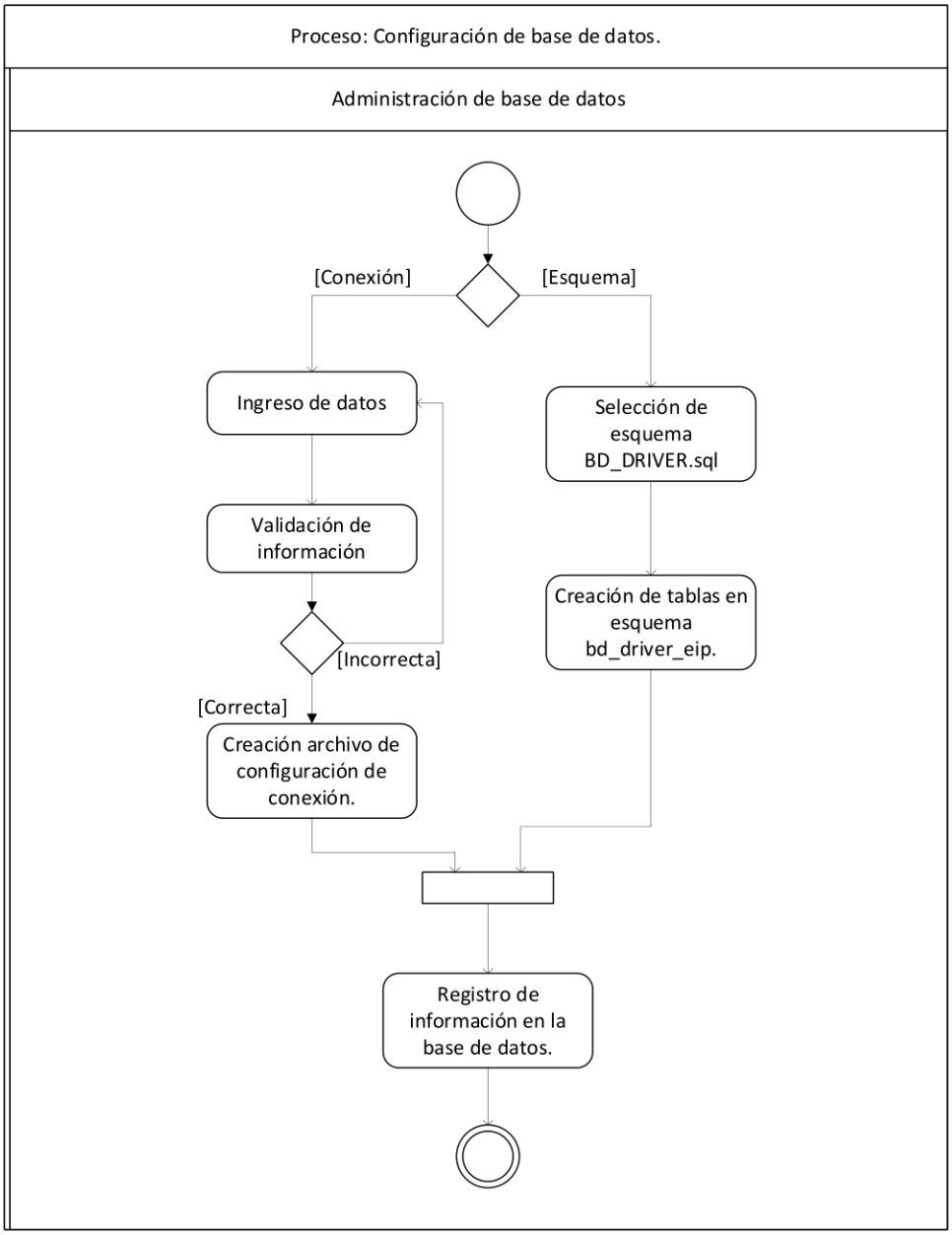


Figura 2.21. Flujo de actividades para configuración de base de datos.

Al ingresar la información de conexión se crea un archivo de texto en un directorio de nombre EIPDriver que guarda estos parámetros para utilizarlos cuando se realice la conexión a la base de datos. La ubicación del archivo es en la carpeta de perfil de usuario.

Cada una de las secciones observadas en los diagramas de actividades anteriores representan la instancia de una clase que maneja y gestiona todo lo relacionado con esa sección.

2.2.3.1. Lectura

El proceso de lectura inicia con la conexión de la PC con el PLC después de haber abierto la sesión mediante mensajes ENIP. El método de conexión *conectar (gateway, nameTag)* primero arma el tag path que se estructura de la siguiente manera para crear la etiqueta que abrirá la conexión:

```
tag = new Tag (tagPath, TIMEOUT)
tagPath = protocol=ab_eip&gateway=192.168.1.11=&plc=micrologix&elem_size=2&name=N7:0
```

El Timeout es el tiempo de espera de la conexión seteado en 5 segundos.

El método de lectura *lecturaTag (tipoDato, bit)* obtiene y retorna el dato solicitado en el tag path y lo almacena en la memoria local de la PC que ejecuta la biblioteca, los argumentos requeridos por este método son el tipo de dato y en el caso de ser booleano el bit que requiere ser leído. Los métodos utilizados para lectura dependiendo del dato son:

- tag.getInt16(tamañoDato): lee un dato tipo entero.
- tag.getBit(tamañoDato): lee un dato tipo bool.
- tag.getFloat32(tamañoDato): lee un dato tipo flotante.

2.2.3.2. Escritura

El proceso de escritura igual que la lectura inicia con la conexión de la PC y el PLC, especificando la dirección IP y la dirección del tag. El método escrituraTag(valorTag, tipoDato, bit) contiene los argumentos valor del tag, tipo de dato y en caso de ser tipo booleano el bit que se requiere escribir, el valor del atributo valorTag se escribe en el tag especificado, utilizando los métodos:

- tag.setInt16 (0, valorTag): escribe un dato tipo entero.
- tag.setBit (bit, valorTag): escribe un dato tipo bool.
- tag.setFloat32 (0, valorTag): escribe un dato tipo flotante.

2.2.4. Base de datos

Para el manejo de base de datos se crea una clase que cuenta con un método de conexión, métodos de lectura y métodos de escritura en la base de datos.

Conexión a base de datos

El gestor de base de datos utilizado es MySQL el cual puede conectarse con otras aplicaciones mediante sentencias SQL, cabe mencionar que en java es necesario utilizar el conector mysql-connector-java-5.1.47. La conexión a la base de datos se realiza mediante el método DriverManager.getConnection(url, usuario, contraseña) con los siguientes datos:

- Class.forName(“com.mysql.jdbc.Driver”): especifica la dirección del driver del conector que establece la conexión.
- URL="jdbc:mysql://localhost:3306/bd_driver_eip?useTimezone=true&serverTimezone=GTM&useSLL=false": especifica el puerto de conexión , el nombre de la base de datos y la zona horaria .
- Usuario=“root”: especifica el nombre de usuario de la conexión.
- Contraseña=”1234”: especifica la contraseña de la conexión.

Lectura de base de datos

El proceso de lectura inicia con la conexión a la base de datos, posteriormente se establece un método que permita construir las sentencias SQL, se ejecuta la solicitud, se obtiene el dato indicando el número de columna en la tabla y finalmente se cierra la conexión.

```
PreparedStatement  pps = this.getConnection().prepareStatement(sentencia)
ResultSet rs = pps.executeQuery()
datosLeido = rs.getString(#columnaTabla)
pps.getConnection().close()
```

Escritura de base de datos

El proceso de escritura inicia con la conexión, posteriormente se ejecuta un método que permite la construcción de la sentencia indicada, se escribe el dato especificando la columna y el valor, se ejecuta la actualización y se cierra la conexión.

```
PreparedStatement  pps = this.getConnection().prepareStatement(sentencia)
pps.setString(#columna, valorEscrito)
pps.executeUpdate()
pps.getConnection().close()
```

Es de gran importancia cerrar la conexión a la base de datos para evitar sobrecargar el servidor provocando que la aplicación colapse o no permita abrir nuevas conexiones al superar el límite aceptado por MySQL.

Diseño de base de datos

Para el diseño de la base de datos es fundamental conocer la información que se almacenará y su relación entre sí. Además, se debe contar con un respaldo de la configuración realizada para cargarla al programa cuando sea ejecutado, por tanto, se almacena la siguiente configuración:

- Nombre del PLC
- Dirección IP
- Device Group
- Tiempo de lectura
- Nombre de tag
- Tipo de tag
- Dirección del tag
- Acción del tag
- Requiere registro histórico
- Valor del tag

Una vez definida la información se realiza la normalización de la base de datos y se obtienen 6 tablas:

- PLCS: registra la información que caracteriza a cada dispositivo.
- Intermedia: es una tabla de unión que enlaza el PLC con los tags correspondientes.
- Tags: registra la información que caracteriza a cada tag.
- Entero: registra el valor del tag tipo entero.
- Bool: registra el valor del tag tipo booleano.
- Flotante: registra el valor del tag tipo flotante.

En la Figura 2.22 se observa la representación visual del diseño de la base de datos por medio de un diagrama entidad relación mejorado conocido como EER donde se visualiza la relación de los atributos de cada tabla entre sí.

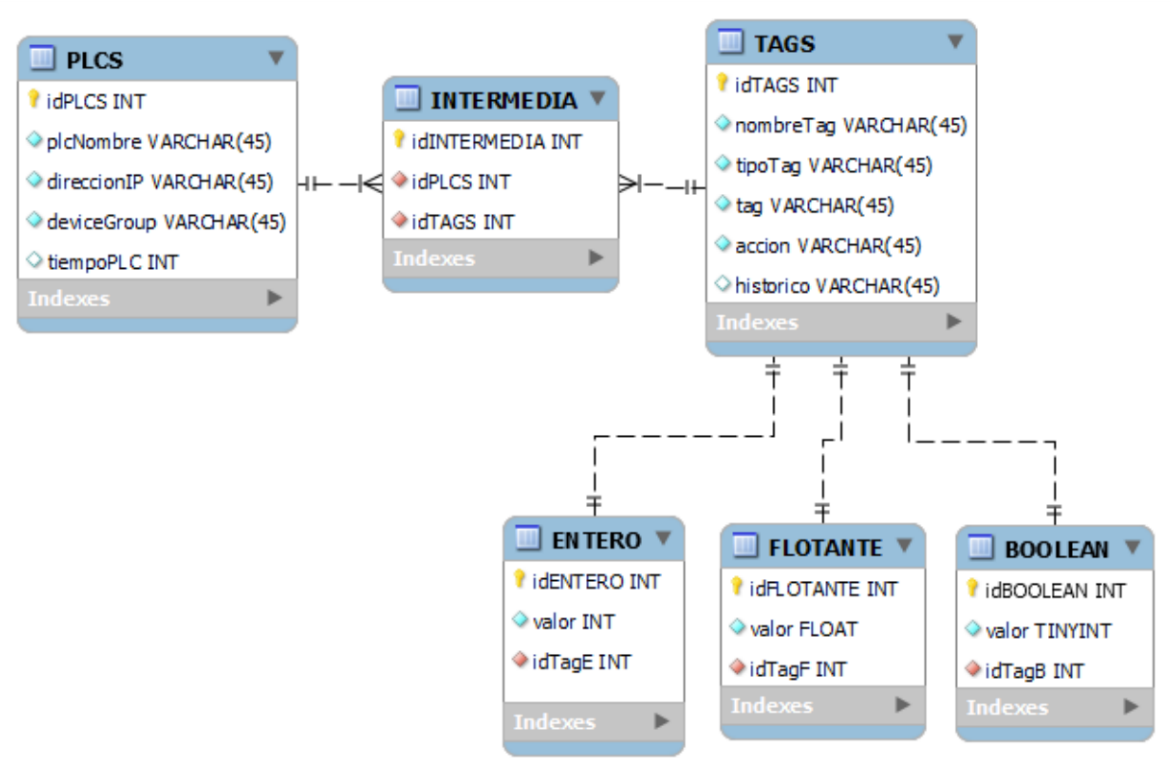


Figura 2.22. Diagrama EER de la base datos diseñada.

En la tabla PLCs se tiene información de un dispositivo con su correspondiente identificador principal, para conocer los tags existentes en el PLC se analiza la tabla Intermedia en la cual se visualiza el identificador del PLC con el correspondiente identificador del tag de la tabla de Tags, en esta tercera tabla en función del tipo del tag y el identificador se accede

a una de las tablas Entero, Flotante o Boolean para obtener el valor de la variable seleccionada.

Para el almacenamiento de datos de registro histórico se requiere la siguiente información:

- Hora y fecha
- Tag que requiere registro histórico

La tabla que almacena la hora y el valor del tag no tiene ninguna relación con las tablas anteriores, es independiente y única para cada PLC, es decir por cada PLC con tags que requieran registros históricos se crea una tabla que contenga los valores de los tags seleccionados en función del tiempo.

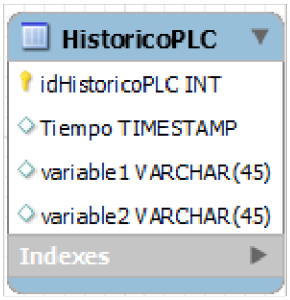


Figura 2.23. Representación de tabla para registro histórico.

2.3. DIAGRAMA DE CLASES

Cada una de las secciones definidas en la lógica de programación representa una clase, por tanto, se tiene: la clase MainInterface que representa a la sección de configuración y a su vez es la ventana principal, la clase AdminPLC que representa a la sección de administración de PLCs, la clase Visualización que representa a la sección de Modo Automático, la clase Históricos que representa a la sección de Manejo de históricos, la clase VentanaEscritura que representa a la sección Modo manual, la clase BaseDatosAdministrador que representa a la sección de Administración de base de datos y la clase PLCConector que representa a la sección Conexión PLC.

Las clases PLC y TagBean son clases simples que definen un objeto y sus atributos privados con sus respectivos getters y setters públicos.

El diagrama de clases permite visualizar las clases, métodos y atributos que forman parte del software y su relación de dependencia entre sí, lo cual hace posible comprender

rápidamente la estructura de la herramienta computacional implementada. En la Figura 2.24 se observa el diagrama completo del software desarrollado.

Figura 2.24. Diagrama de clases completo.

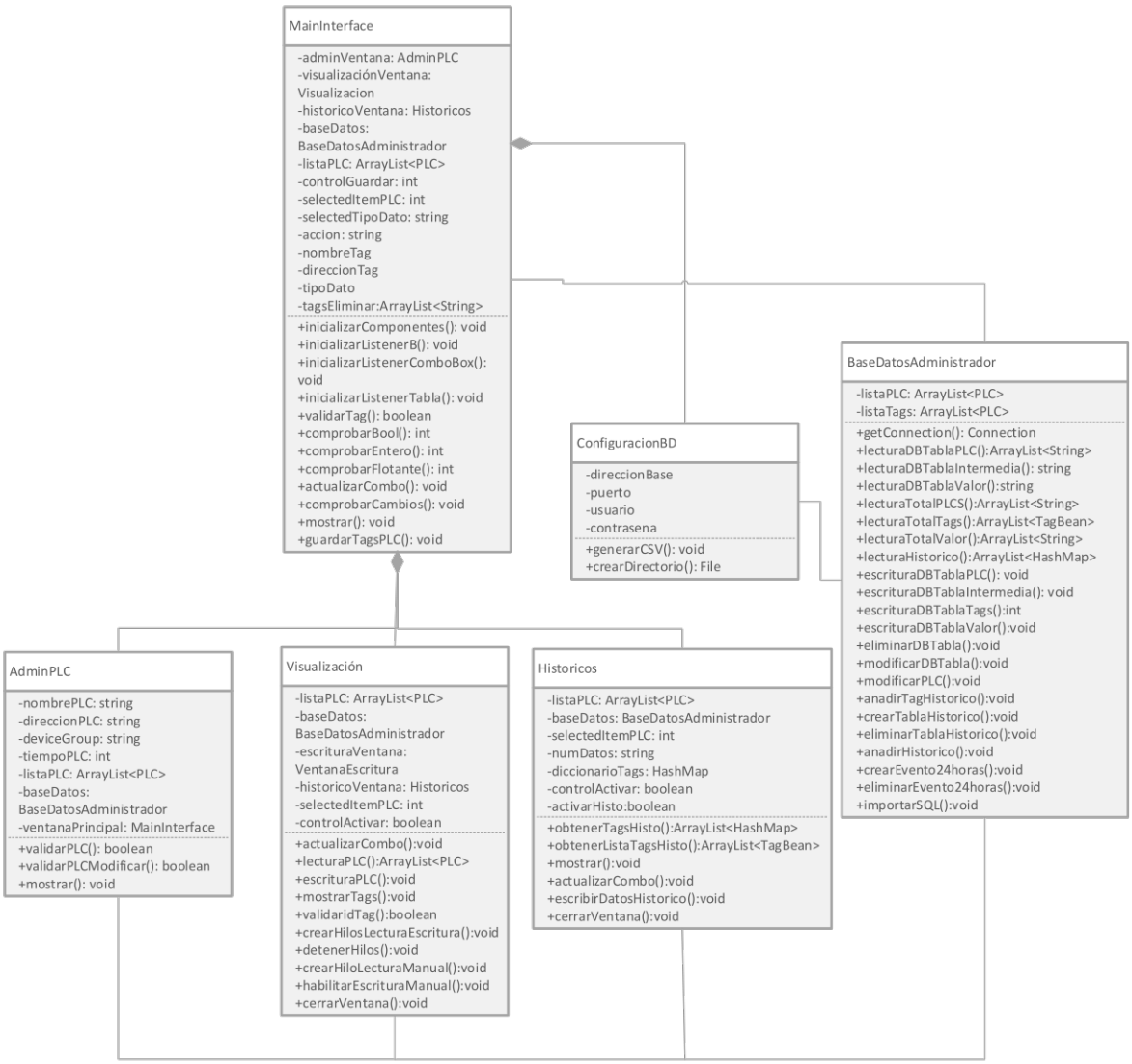


Figura 2.25. Diagrama de clases MainInterface, AdminPLC, Visualización, Históricos y BaseDatosAdministrador.

Las clases AdminPLC, Visualización, Históricos y ConfiguracionBD tienen una relación de composición con la clase MainInterface, es decir su existencia depende directamente de esta clase, si no existiese la sección de configuración no tiene sentido que se realice la lectura, escritura y registro de históricos. A su vez estas clases tienen una relación de asociación con la clase BaseDatosAdministrador que se encarga de gestionar la conexión, lectura y escritura de datos que se requiera. En la Figura 2.25 se observa el diagrama de las clases mencionadas.

Figura 2.26. Diagrama de clases AdminPLC, PLC y TagBean.

La clase AdminPLC se compone de la clase PLC y a su vez ésta tiene una relación de composición con la clase TagBean. En otras palabras, para la administración de dispositivos, se requiere de PLCs con cada uno de sus atributos especificados y cada PLC precisa de tags con sus correspondientes características.

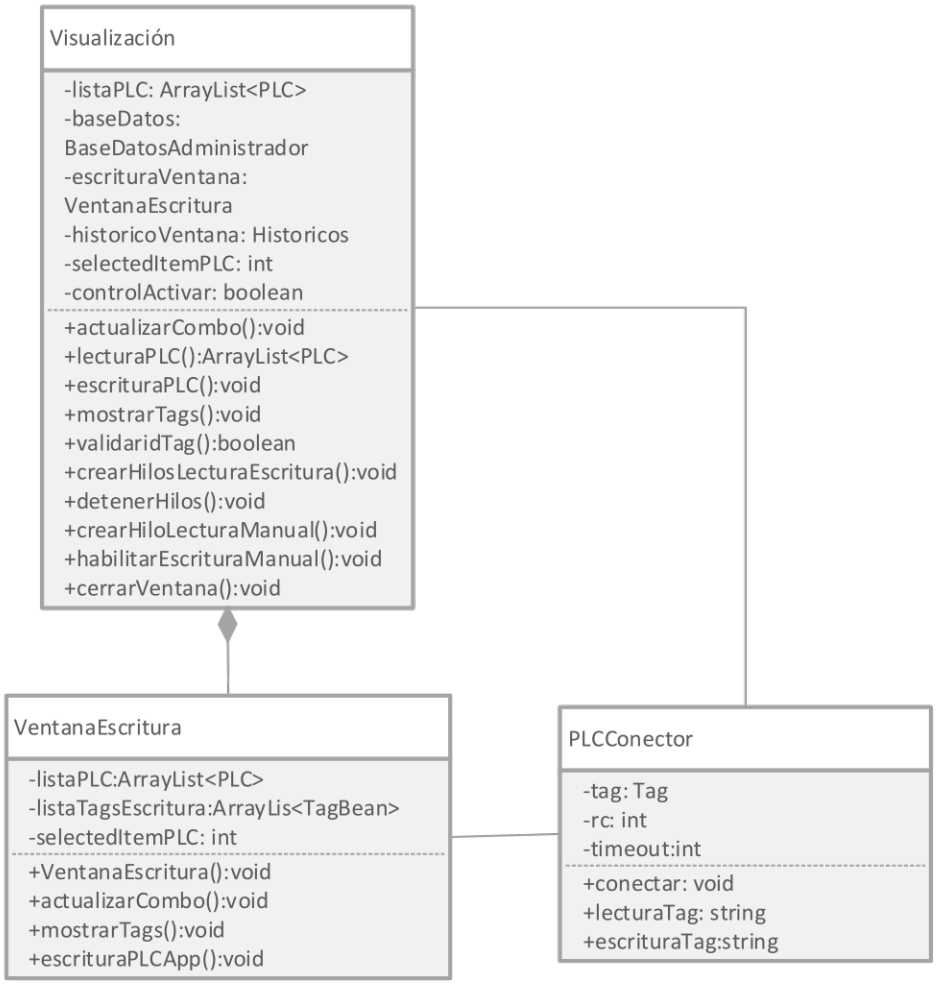


Figura 2.27. Diagrama de clases Visualización, VentanaEscritura y PLCConector.

La clase Visualización se compone de la clase VentanaEscritura y ambas tienen una relación de asociación con PLCConector (Figura 2.27) ya que requieren gestionar la conexión, lectura y escritura con el PLC.

2.4. DISEÑO INTERFAZ GRÁFICA DE USUARIO

La interfaz gráfica permite al usuario configurar y visualizar de una forma más fácil las actividades y procesos que se realizan. El diseño de la interfaz se basa en los siguientes requerimientos:

- Interface simple y de fácil entendimiento.
- Uso de elementos comunes y patrones de funcionamiento en cada ventana.

- Información permanente al usuario de la acción que se está realizando o si se ha presentado algún error.
- Robustez ante errores presentados con la información ingresada o la conexión con los PLCs.
- Visualización continua del estado de los tags y registros históricos.
- Ingreso de información de importancia para establecer comunicación con los dispositivos.
- Navegación entre ventanas.

2.4.1. Ventana de configuración

En la Figura 2.28 se observa la ventana de configuración que a su vez es la ventana principal del software ya que permite el acceso a nuevas ventanas.

- PLC: En la parte izquierda en la sección “PLC” se tiene un cuadro combinado que permite al usuario elegir el dispositivo al cuál desea añadir o eliminar variables y cuenta también con un botón que guarda los cambios efectuados.
- Tags: en la parte central “Tags” se tienen campos de ingreso de texto para el nombre de la variable y la dirección del tag que sigue el modelo B3:0/0, N7:0 y F8:0 para los tipos de dato Bool, Entero, Flotante que se encuentran en el primer cuadro combinado, adicionalmente se brinda la opción de seleccionar la acción de lectura o escritura que se realiza con el tag y definir mediante una casilla de verificación si requiere o no registro histórico. Esta sección incluye el botón añadir que permite al usuario agregar tags y el botón eliminar que descarta tags creados previamente.
- En el área inferior de la pantalla se encuentra la tabla de visualización relacionada con la sección Tags considerando los campos: Variable, Tag, Tipo de Dato, Acción y Registro Histórico. Esta tabla registra la lista de tags del PLC seleccionado en el cuadro combinado.
- Ventanas: esta sección comprende 4 botones utilizados para acceder a las ventanas Configuración de base de datos, Administración de PLCs, Visualización y Registro de históricos.

En la Figura 2.29 se observa el diagrama que explica el funcionamiento de esta pantalla y detalla los procesos que ocurren si se presiona o no cada botón. Con este diagrama se

puede entender también la secuencia de pasos que se realizan para completar un proceso utilizando la interfaz.

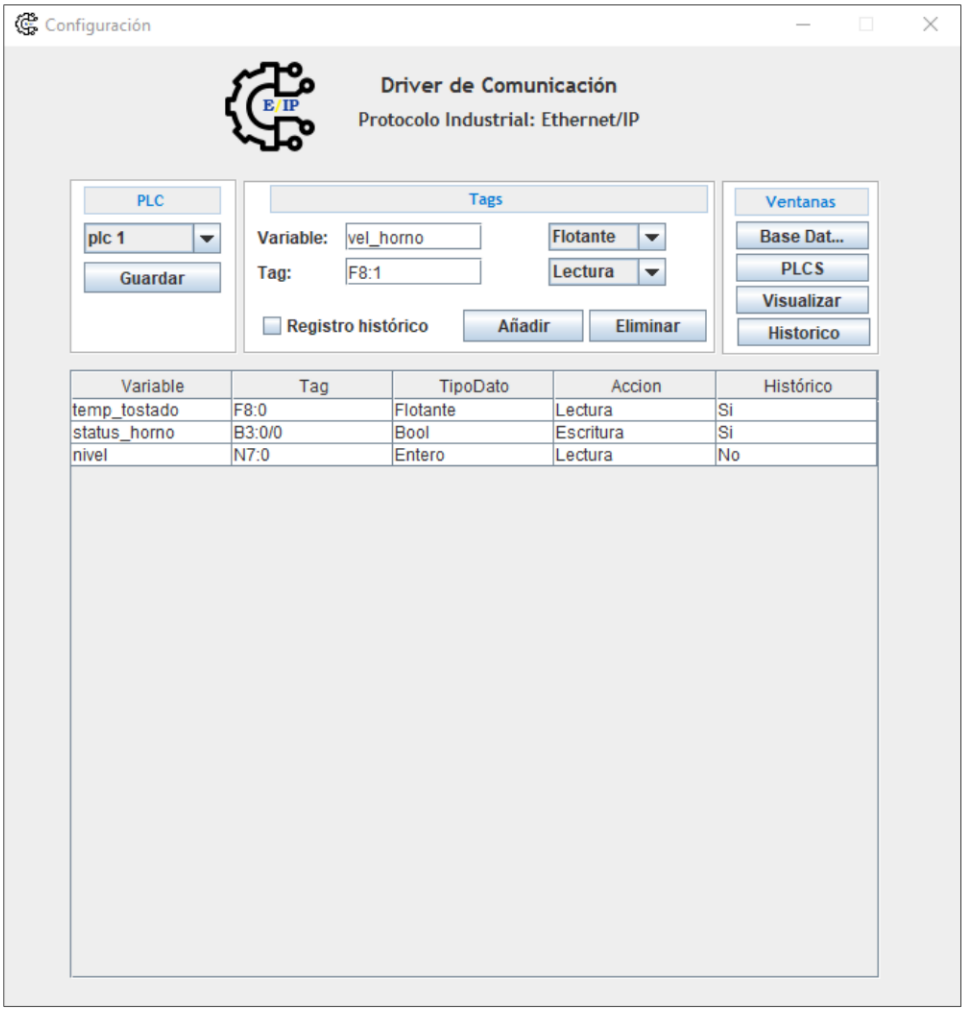


Figura 2.28. Ventana de configuración.

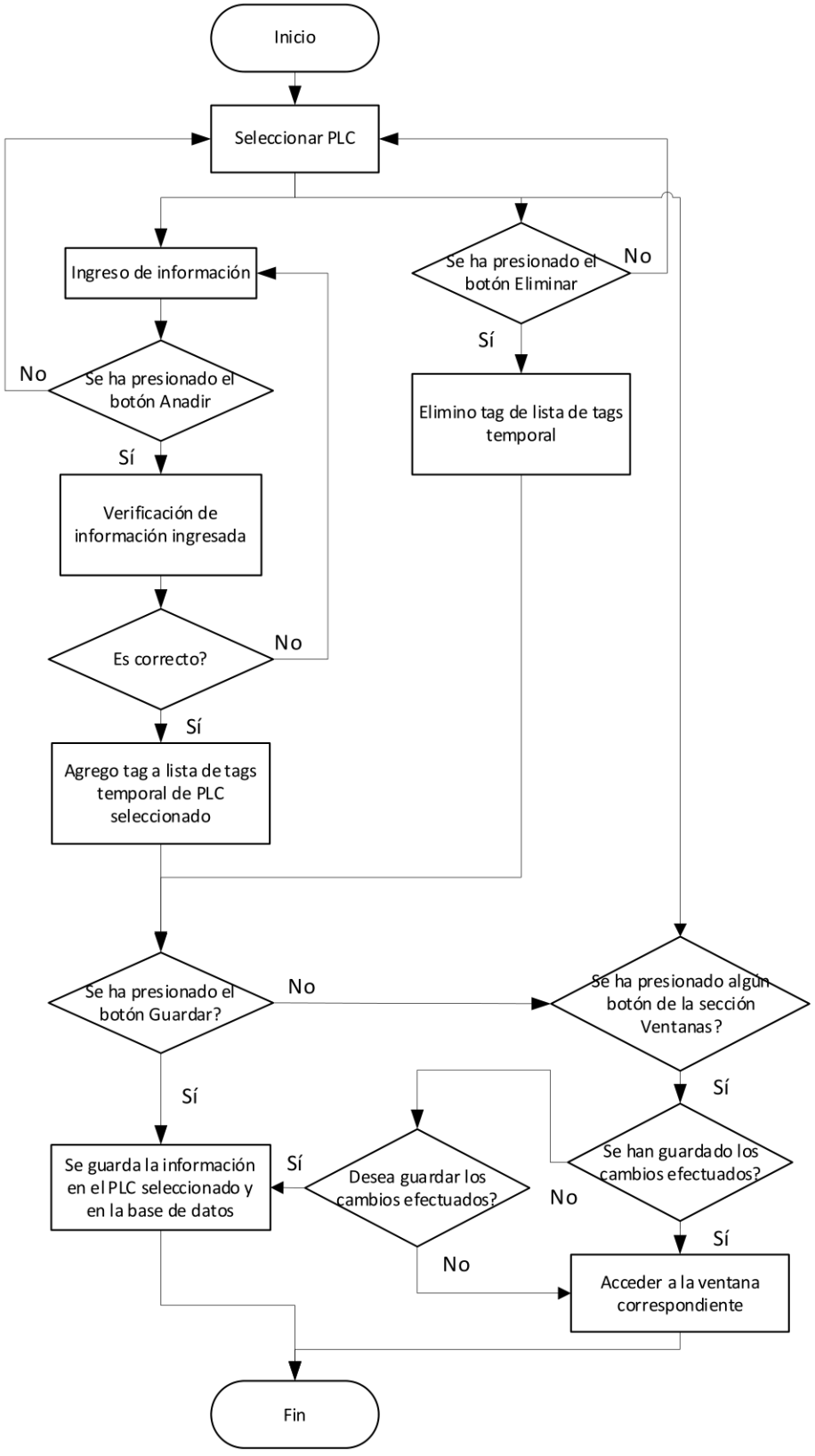


Figura 2.29. Diagrama de flujo de funcionamiento de ventana de configuración.

2.4.2. Ventana de configuración de base de datos

En la Figura 2.30 se observa la ventana de configuración para lograr la conexión con la base de datos, es de gran importancia indicarle al programa a donde se debe conectar para acceder a la información almacenada, por tanto, los datos ingresados deben coincidir con la conexión existente en MySQL. Esta sección comprende de cuatro campos de ingreso de texto:

- Dirección IP: indica la dirección IP donde se encuentra la base de datos.
- Puerto: indica el puerto de acceso a la aplicación, en el caso de MySQL el valor por defecto es 3306.
- Usuario: indica el usuario que se utiliza para la conexión a la base de datos.
- Contraseña: indica la contraseña para acceder a la conexión que permite el ingreso a la base de datos.

Además, tiene dos botones:

- Elegir esquema: al presionar este botón se abre una ventana que permite navegar en los archivos de la PC para elegir el archivo de texto que posibilita la creación de tablas de almacenamiento en la base de datos.
- Aceptar: al presionar este botón la información ingresada en los campos de texto es guardada en un archivo .txt.

Figura 2.30. Ventana de configuración de base de datos.

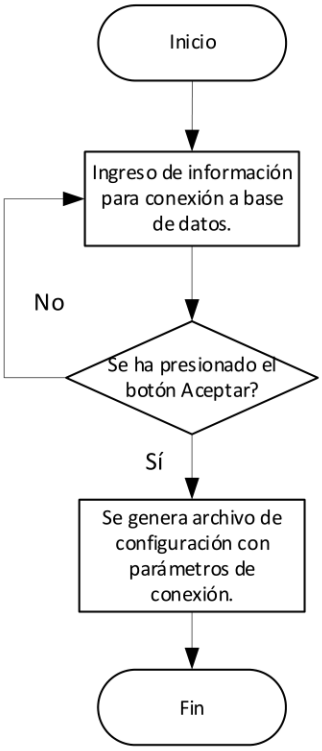


Figura 2.31. Diagrama de flujo de funcionamiento de ventana de configuración de base de datos.

2.4.3. Ventana de administración de PLCs

En la Figura 2.32 se observa la Ventana de administración de PLCs encargada del manejo de dispositivos, brindando funciones de creación, modificación y eliminación. Cuenta con dos secciones:

- Datos PLC: esta sección comprende cuatro campos de ingreso de texto para registrar la información del PLC como nombre, dirección IP, device group y tiempo de muestreo que puede estar entre el rango de 150 ms a 2000 ms, además, contiene el botón Añadir que agrega el PLC con los datos ingresados en los campos mencionados, el botón Eliminar que descarta el PLC seleccionado en la tabla y el botón Modificar que permite editar los datos de los PLCs existentes.
- Tabla: en la parte inferior se tiene una tabla de visualización de la lista de PLCs con sus respectivas características.

En la Figura 2.33 se observa el diagrama que indica el funcionamiento de la pantalla administración de PLCs.

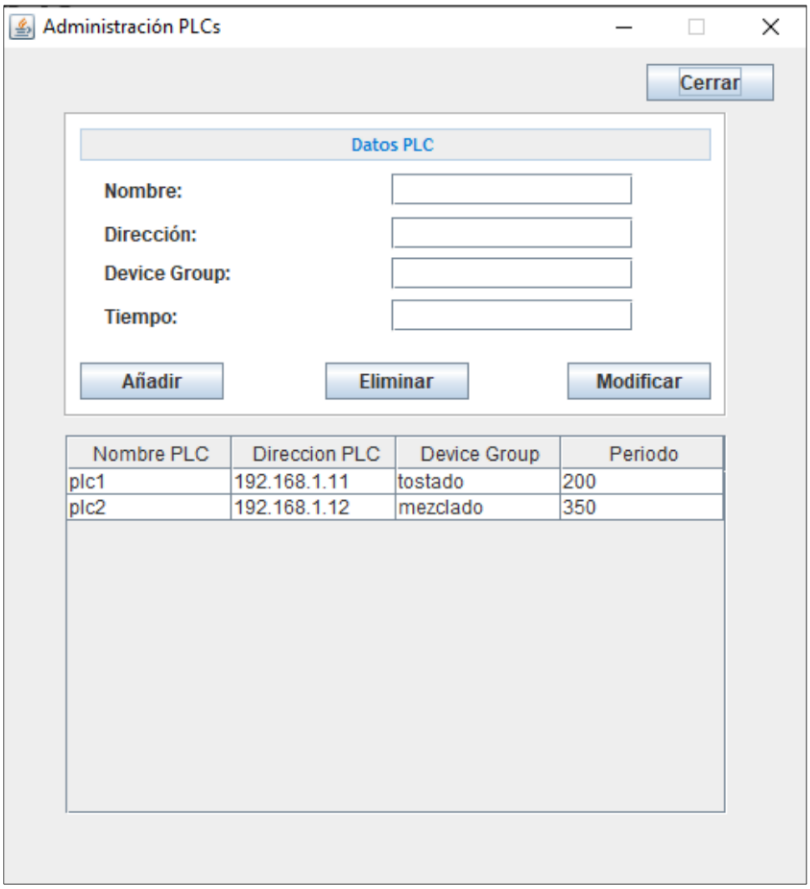


Figura 2.32. Ventana de administración de PLCs.

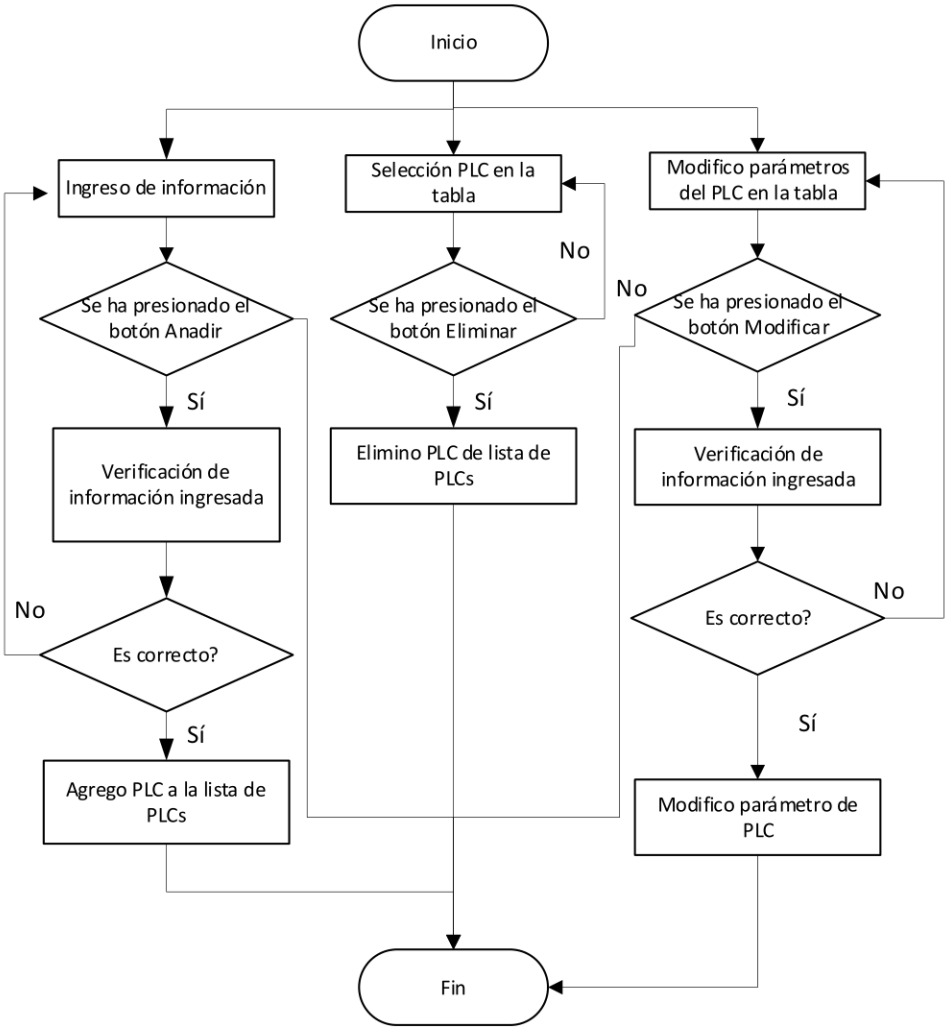


Figura 2.33. Diagrama de flujo de funcionamiento de ventana administración de PLCs.

2.4.4. Ventana de visualización

En la Figura 2.34 se observa la ventana de Visualización la cual permite al usuario observar el funcionamiento del servidor con respecto al intercambio de datos. Presenta cuatro secciones:

- Estado: en esta sección se tiene un cuadro combinado que permite seleccionar el PLC y se indica el estado del funcionamiento del servidor.
- Modo Automático: esta sección contiene el botón Activar que inicia la lectura y escritura automática con el PLC, pone en marcha el registro de históricos cada minuto y la lectura continua de la base de datos de los valores destinados a escritura en el dispositivo, también contiene el botón Desactivar que detiene todas las tareas que se ejecutaron al activar el servidor.

- Escritura Manual: en esta sección se cuenta con una casilla de verificación que habilita el botón de escritura, en caso de encontrarse funcionando el modo automático lo desactiva inmediatamente. El botón de escritura permite el acceso a la ventana Escritura manual.
- Tabla: en la parte inferior se presenta una tabla con la lista de tags del PLC seleccionado, cuenta con los campos Nombre, Tipo de dato, Valor actual de la variable y Acción.

En la Figura 2.35 se observa el diagrama que explica el funcionamiento de la pantalla de visualización y los procesos que habilitan cada uno de los botones.

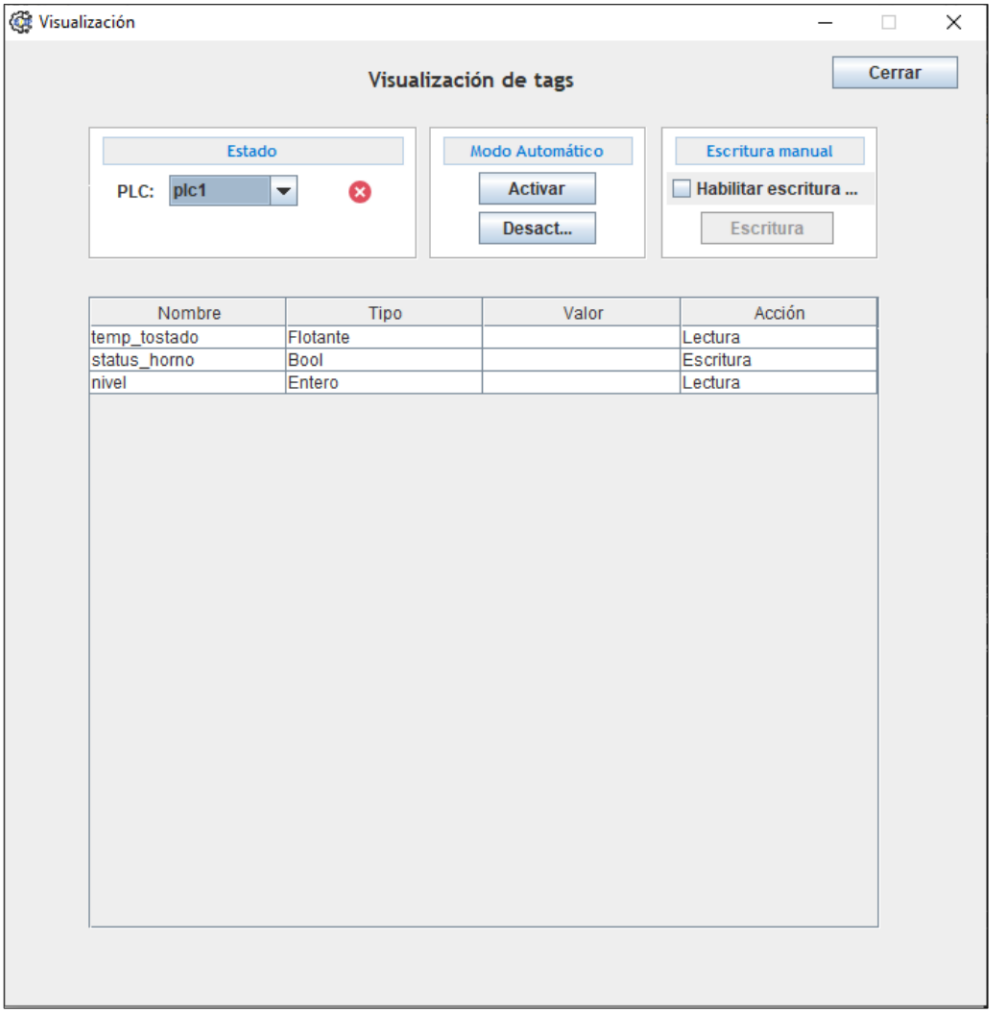


Figura 2.34. Ventana de visualización.

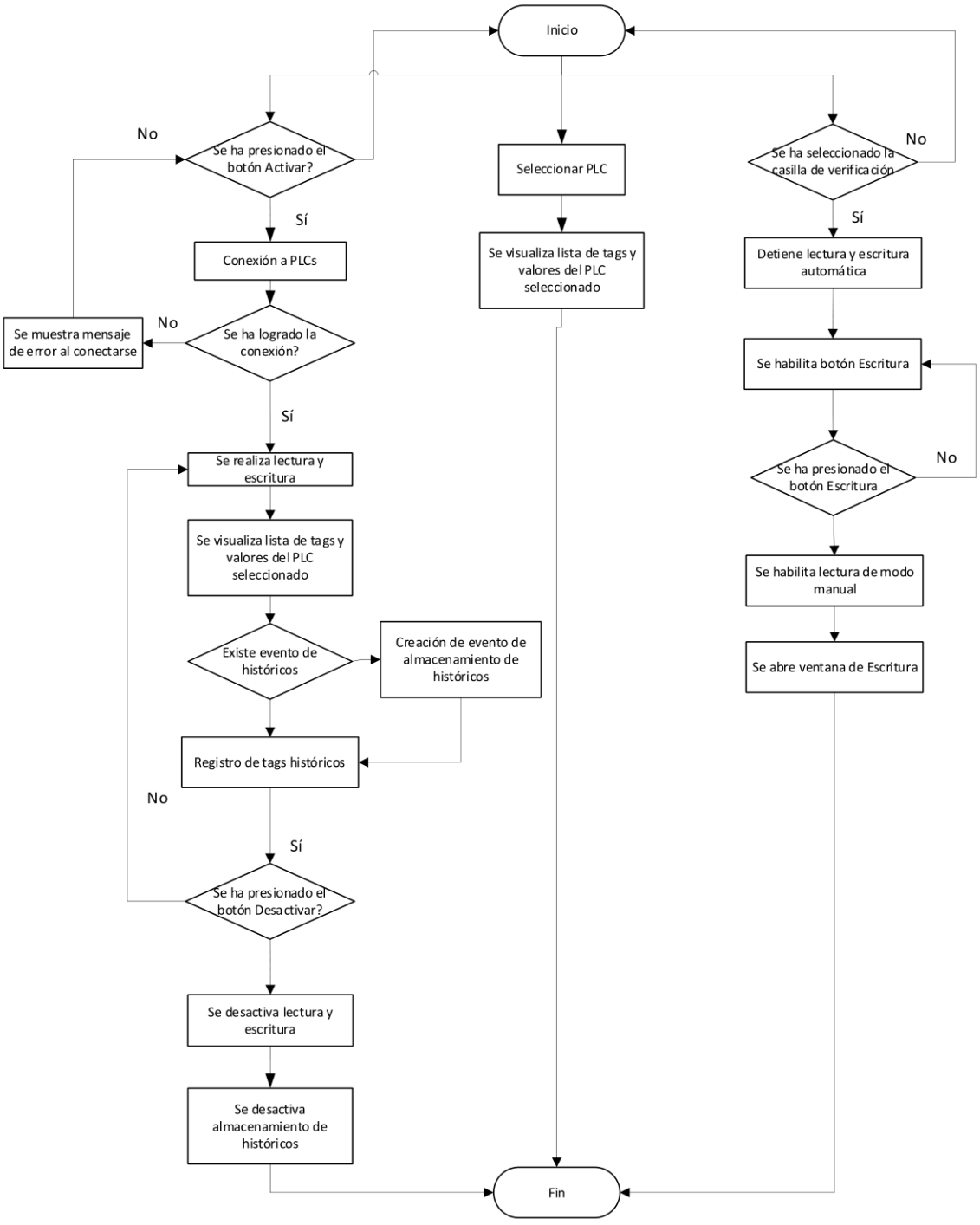


Figura 2.35. Diagrama de flujo de funcionamiento de ventana visualización.

2.4.5. Ventana de escritura en modo manual.

En la Figura 2.37 se observa la ventana Escritura manual que permite al usuario escribir en el PLC desde esta ventana. Al abrirla desde la pantalla de visualización aparece el

mensaje de advertencia que se observa en la Figura 2.36 notificando que se está activando la escritura desde la pantalla manual. En esta pantalla se tienen dos secciones:

- Escritura manual: contiene un cuadro combinado que posibilita la elección del PLC al cual se le requiere escribir el valor del tag que se ha modificado en la tabla y el botón OK que habilita la escritura en el PLC.
- Tabla: permite visualizar la lista de tags de escritura del PLC seleccionado y posibilita el ingreso de datos en la columna Valor para poder realizar la escritura.

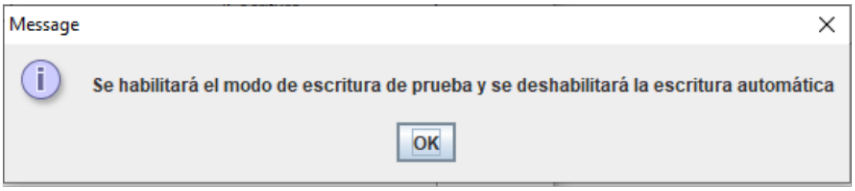


Figura 2.36. Mensaje de advertencia de escritura manual.

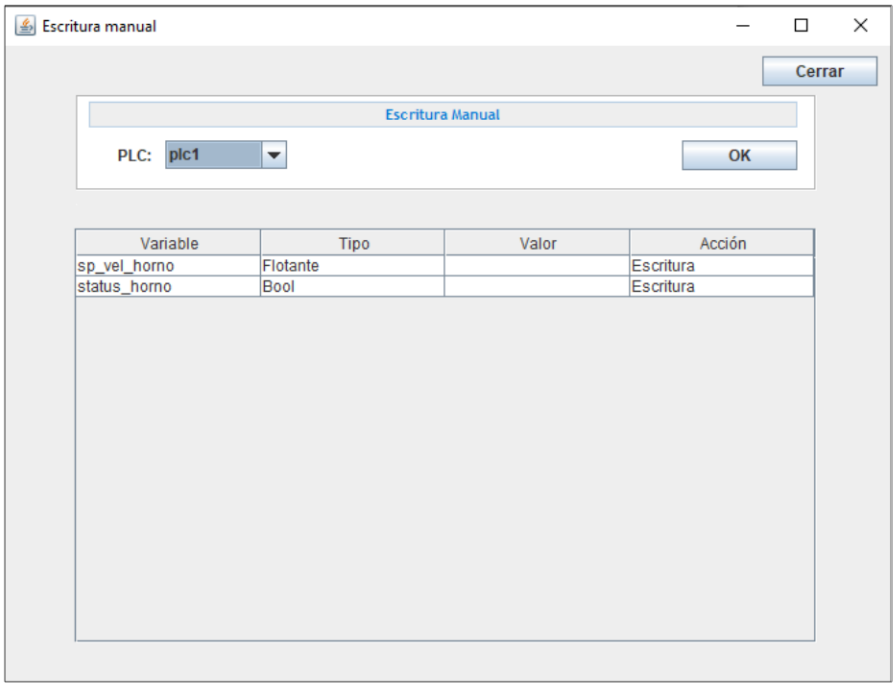


Figura 2.37. Ventana de modo de escritura manual.

En la Figura 2.38 se observa el diagrama de funcionamiento que indica los pasos que se deben seguir para realizar una escritura manual.

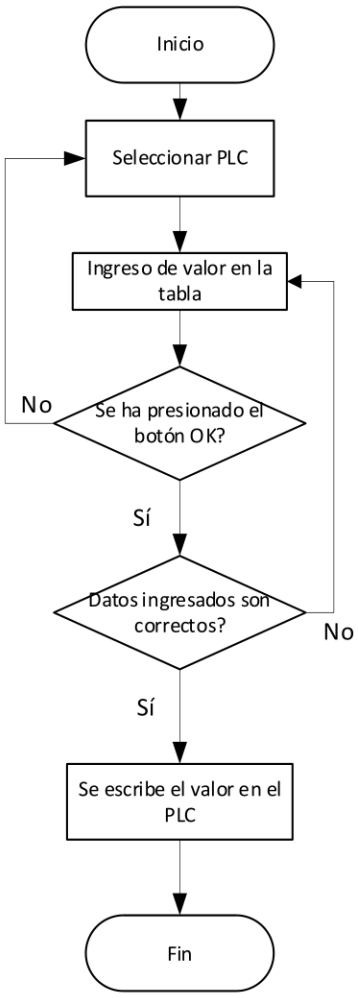
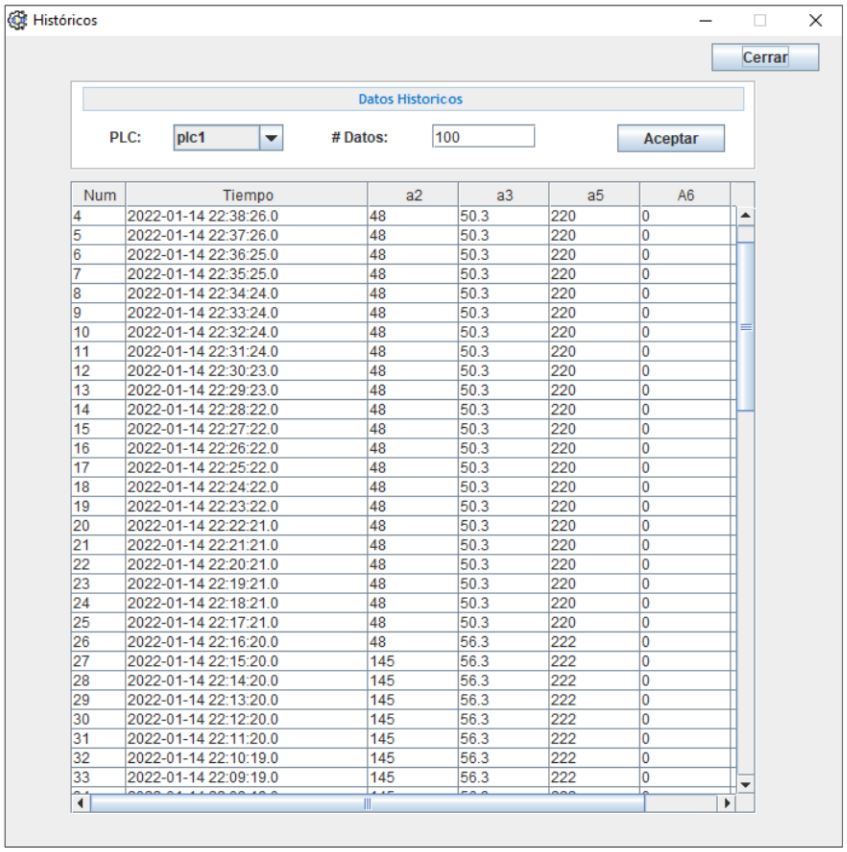


Figura 2.38. Diagrama de flujo de funcionamiento de ventana escritura manual.

2.4.6. Ventana de históricos

En la Figura 2.39 se observa la ventana de históricos la cual permite visualizar el registro histórico de cada PLC en las últimas 24 horas, contiene un cuadro combinado que posibilita la selección del dispositivo, un campo de texto que permite ingresar la cantidad de registros que se requieren visualizar en la tabla y el botón Aceptar que actualiza el número de datos que se ingresó.

En la Figura 2.40 se observa el diagrama de funcionamiento de la ventana que maneja los históricos y el proceso que se debe realizar para visualizar los datos.



Num	Tiempo	a2	a3	a5	A6
4	2022-01-14 22:38:26.0	48	50.3	220	0
5	2022-01-14 22:37:26.0	48	50.3	220	0
6	2022-01-14 22:36:25.0	48	50.3	220	0
7	2022-01-14 22:35:25.0	48	50.3	220	0
8	2022-01-14 22:34:24.0	48	50.3	220	0
9	2022-01-14 22:33:24.0	48	50.3	220	0
10	2022-01-14 22:32:24.0	48	50.3	220	0
11	2022-01-14 22:31:24.0	48	50.3	220	0
12	2022-01-14 22:30:23.0	48	50.3	220	0
13	2022-01-14 22:29:23.0	48	50.3	220	0
14	2022-01-14 22:28:22.0	48	50.3	220	0
15	2022-01-14 22:27:22.0	48	50.3	220	0
16	2022-01-14 22:26:22.0	48	50.3	220	0
17	2022-01-14 22:25:22.0	48	50.3	220	0
18	2022-01-14 22:24:22.0	48	50.3	220	0
19	2022-01-14 22:23:22.0	48	50.3	220	0
20	2022-01-14 22:22:21.0	48	50.3	220	0
21	2022-01-14 22:21:21.0	48	50.3	220	0
22	2022-01-14 22:20:21.0	48	50.3	220	0
23	2022-01-14 22:19:21.0	48	50.3	220	0
24	2022-01-14 22:18:21.0	48	50.3	220	0
25	2022-01-14 22:17:21.0	48	50.3	220	0
26	2022-01-14 22:16:20.0	48	56.3	222	0
27	2022-01-14 22:15:20.0	145	56.3	222	0
28	2022-01-14 22:14:20.0	145	56.3	222	0
29	2022-01-14 22:13:20.0	145	56.3	222	0
30	2022-01-14 22:12:20.0	145	56.3	222	0
31	2022-01-14 22:11:20.0	145	56.3	222	0
32	2022-01-14 22:10:19.0	145	56.3	222	0
33	2022-01-14 22:09:19.0	145	56.3	222	0

Figura 2.39. Ventana de registro de históricos.

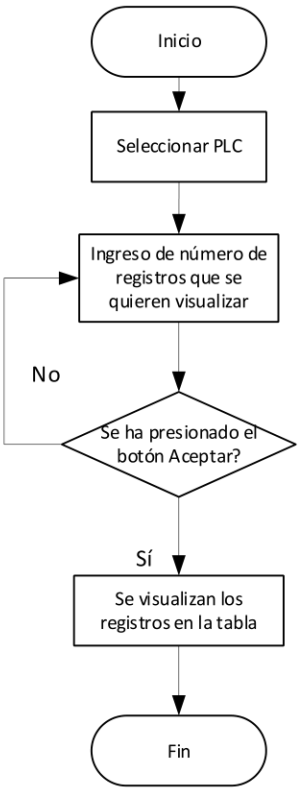


Figura 2.40. Diagrama de flujo de funcionamiento de ventana históricos.

3. RESULTADOS Y DISCUSIÓN

En este capítulo se muestran los resultados obtenidos en las pruebas realizadas. Inicialmente se detalla el funcionamiento del driver en modo manual y automático, posteriormente se realiza la implementación del prototipo simulando el proceso de preparación y producción de chocolate, además, se detallan los subprocesos y variables que requieren ser manejadas por cada uno de los dispositivos lógicos programables. Adicionalmente se realiza una comparación del servidor de datos DASABCIP y la aplicación desarrollada.

La configuración inicia en la ventana de administración de PLCs en la cual se agregan dos dispositivos: ML1100 Serie A y ML1100 Serie B para realizar las pruebas de funcionamiento del programa. Cada PLC contiene la siguiente información:

ML1100 Serie B:

- Nombre: PLC1
- Dirección IP: 192.168.1.11
- Device Group: tostado
- Tiempo de muestreo: 500 ms

ML1100 Serie A:

- Nombre: PLC2
- Dirección IP: 192.168.1.12
- Device Group: mezclado
- Tiempo de muestreo: 300 ms

En el dispositivo “plc1” se agregan distintos tags de tipo booleano, entero y flotante, que realizan funciones de lectura o escritura y almacenamiento de registro histórico. En la Figura 3.1 se observa la ventana de configuración con todos los tags agregados en el “plc1”.

Las direcciones de las etiquetas se deben ingresar bajo el siguiente formato:

Tabla 3.1. Formato de tipo de tag.

TIPO	FORMATO TAG
Booleano	B3:0/0
Entero	N7:0
Flotante	F8:0

En caso de no cumplir el formato considerando el tipo de dato no se permite agregar el tag, finalmente presionando el botón “Guardar” se registra la información en la base de datos.

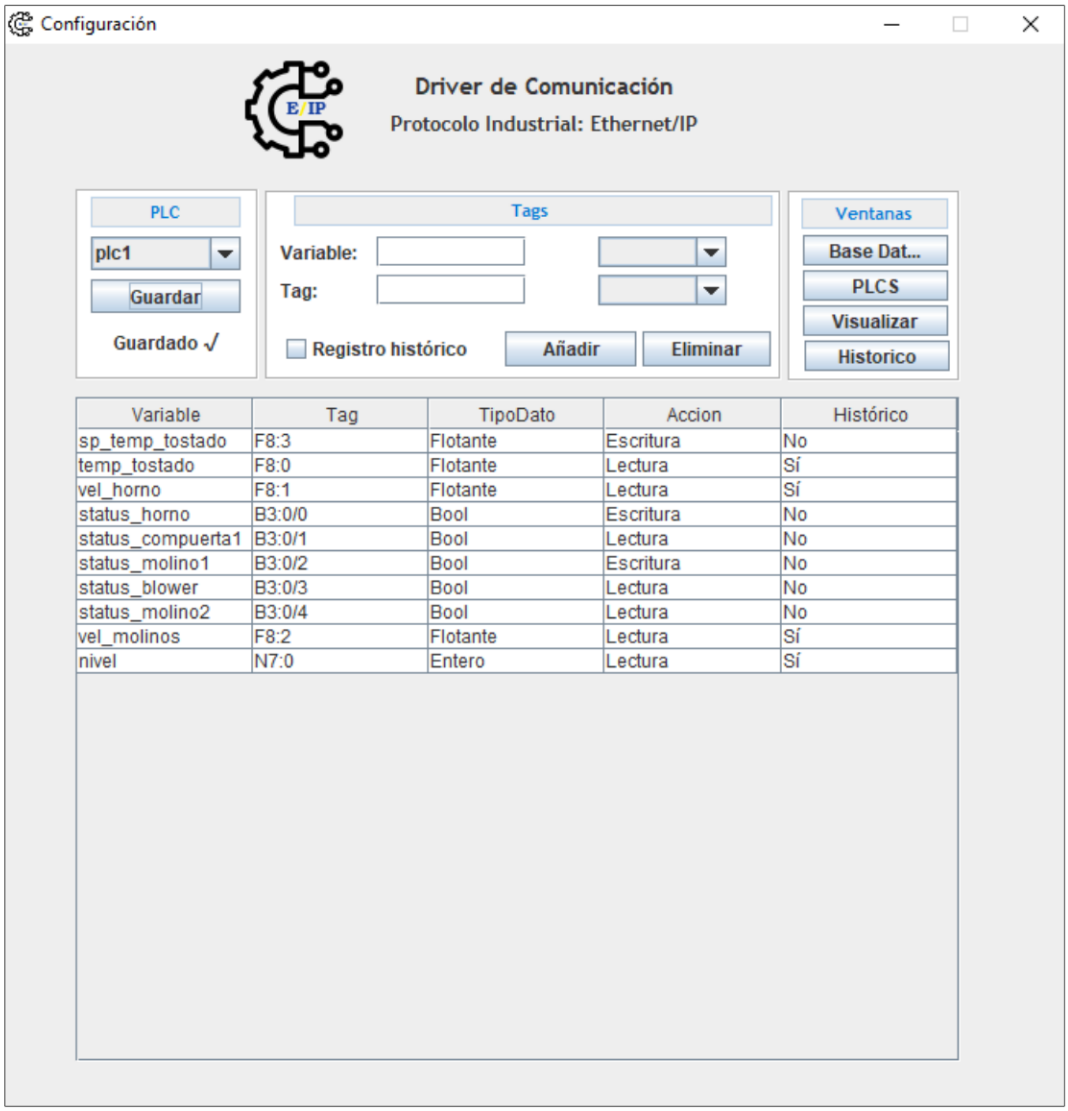


Figura 3.1. Configuración realizada y guardada en el PLC 1.

Una vez ingresadas las variables y terminada la configuración se accede a la ventana de Visualización donde se puede activar el servidor de dos modos: manual y automático.

3.1. MODO MANUAL

El software desarrollado cuenta con un modo manual en el cual se realiza la lectura automática y la escritura manual desde la aplicación, cabe recalcar que este modo se habilita con la casilla de verificación, además, se visualiza un mensaje de advertencia que notifica el cambio en el modo de funcionamiento.

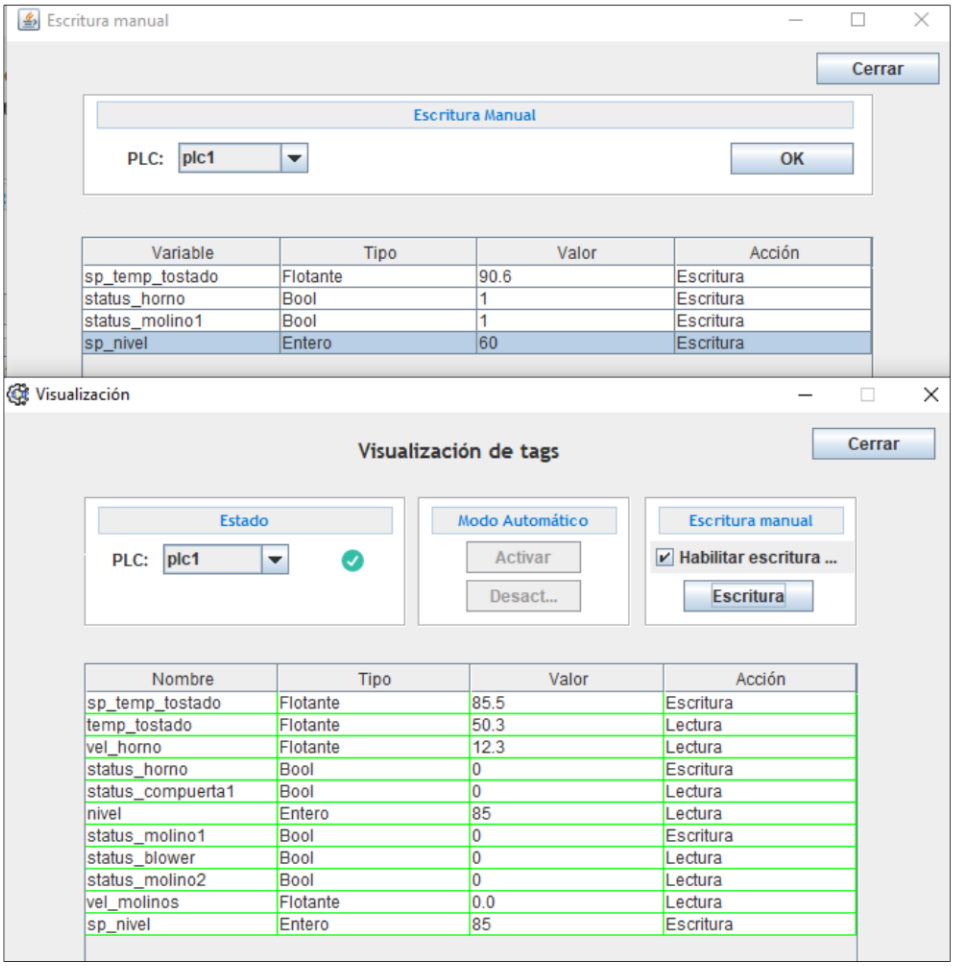


Figura 3.2. Servidor activado y ventana de escritura previo a la escritura.

En la ventana de escritura manual se modifican los valores de las variables destinadas a escritura como se observa en la Figura 3.2., al presionar el botón “OK” se escriben los valores modificados en el PLC y ese cambio se ve reflejado en la tabla de visualización del servidor como se observa en la Figura 3.3 y en los registros del PLC de la Figura 3.4.

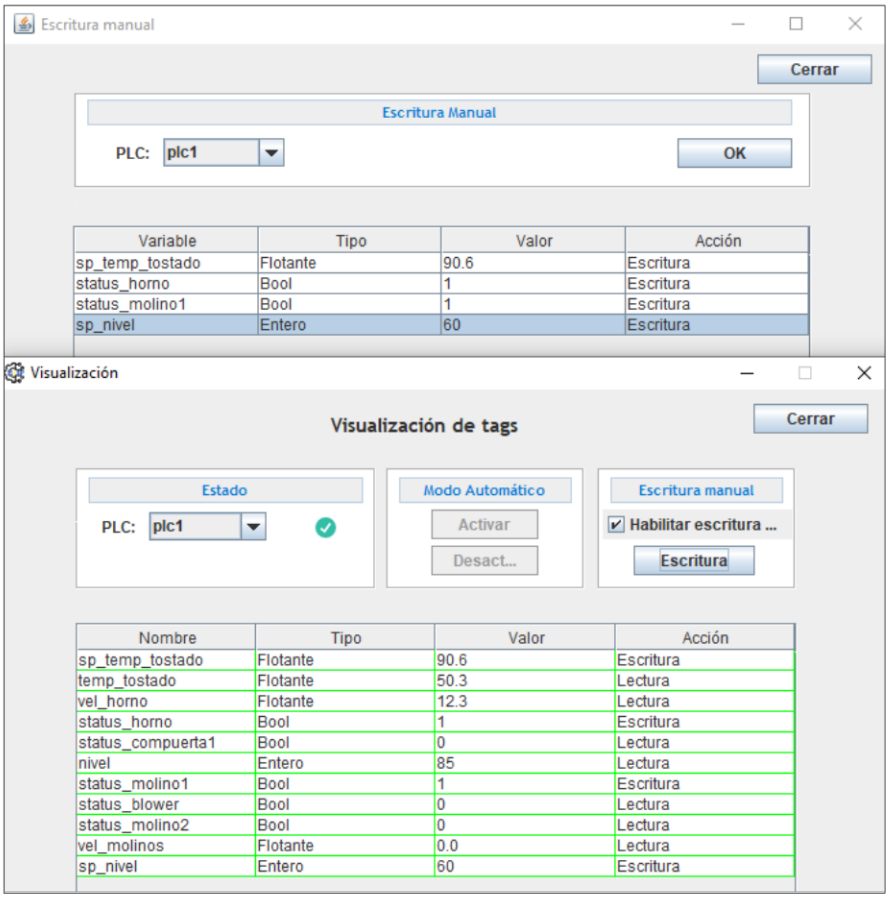


Figura 3.3. Servidor activado y ventana de escritura al realizar la escritura.

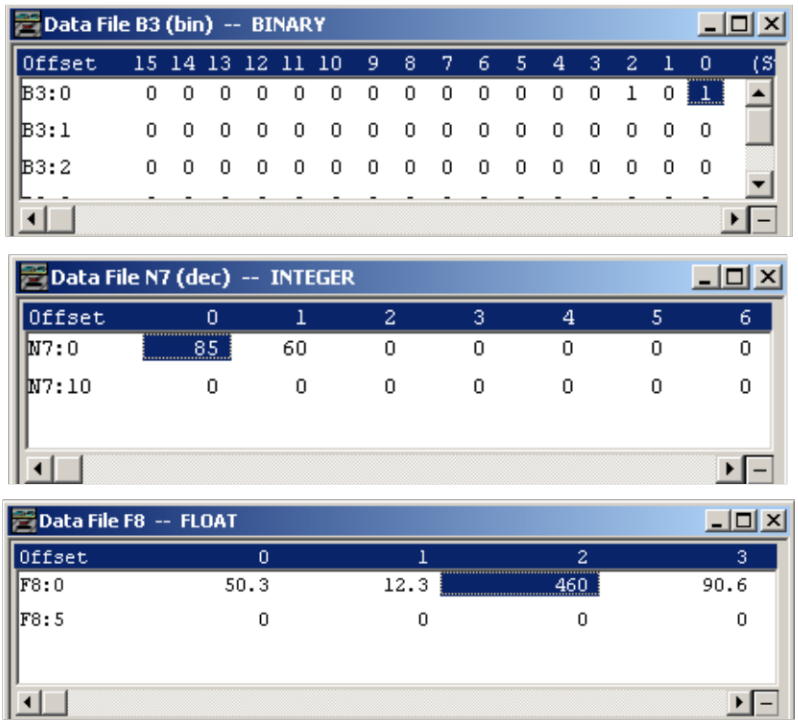


Figura 3.4. Registros en el PLC.

3.2. MODO AUTOMÁTICO

En este modo se realiza la lectura y escritura automáticas, cabe mencionar que los valores destinados a la escritura se toman desde la base de datos. La activación automática inicia al presionar el botón “Activar” como se observa en la Figura 3.5, primero se realiza la lectura de la base de datos únicamente de las variables que requieren ser modificadas en el dispositivo, estos valores se actualizan en la ventana de visualización y posteriormente se escriben en el PLC. Se evidencia la lectura de la base de datos con las tablas que se observan en la Figura 3.6.

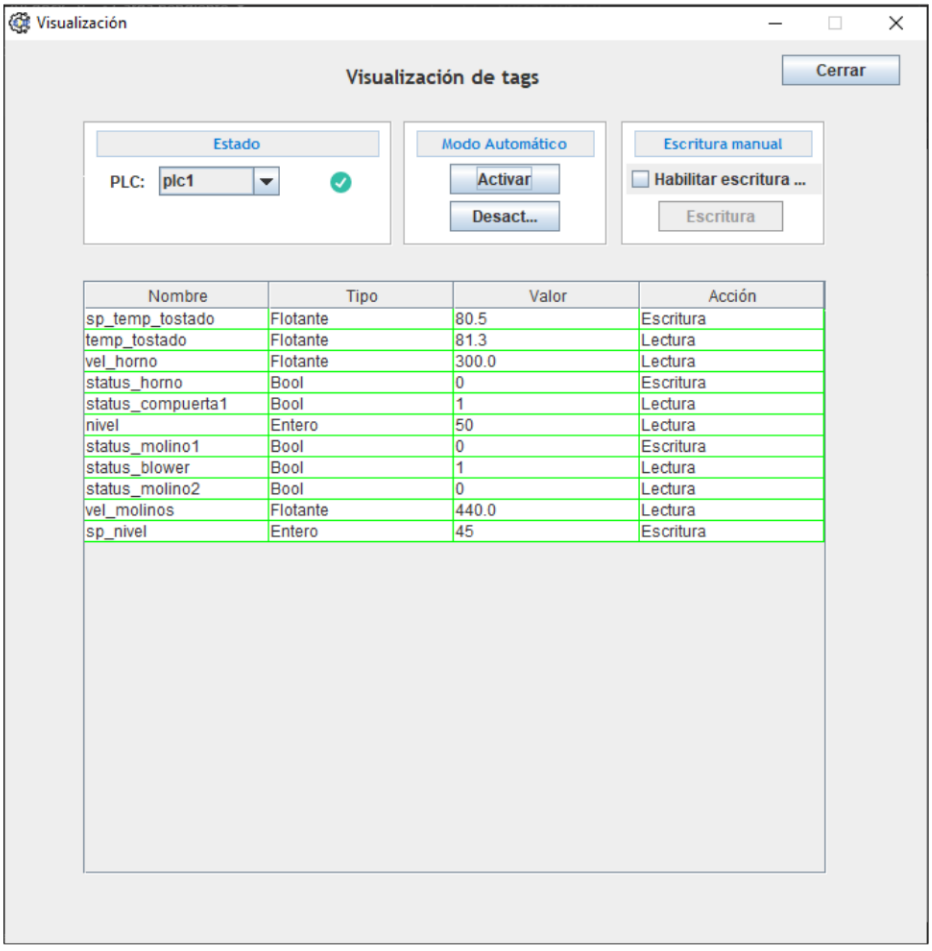


Figura 3.5. Servidor activado para lectura automática.

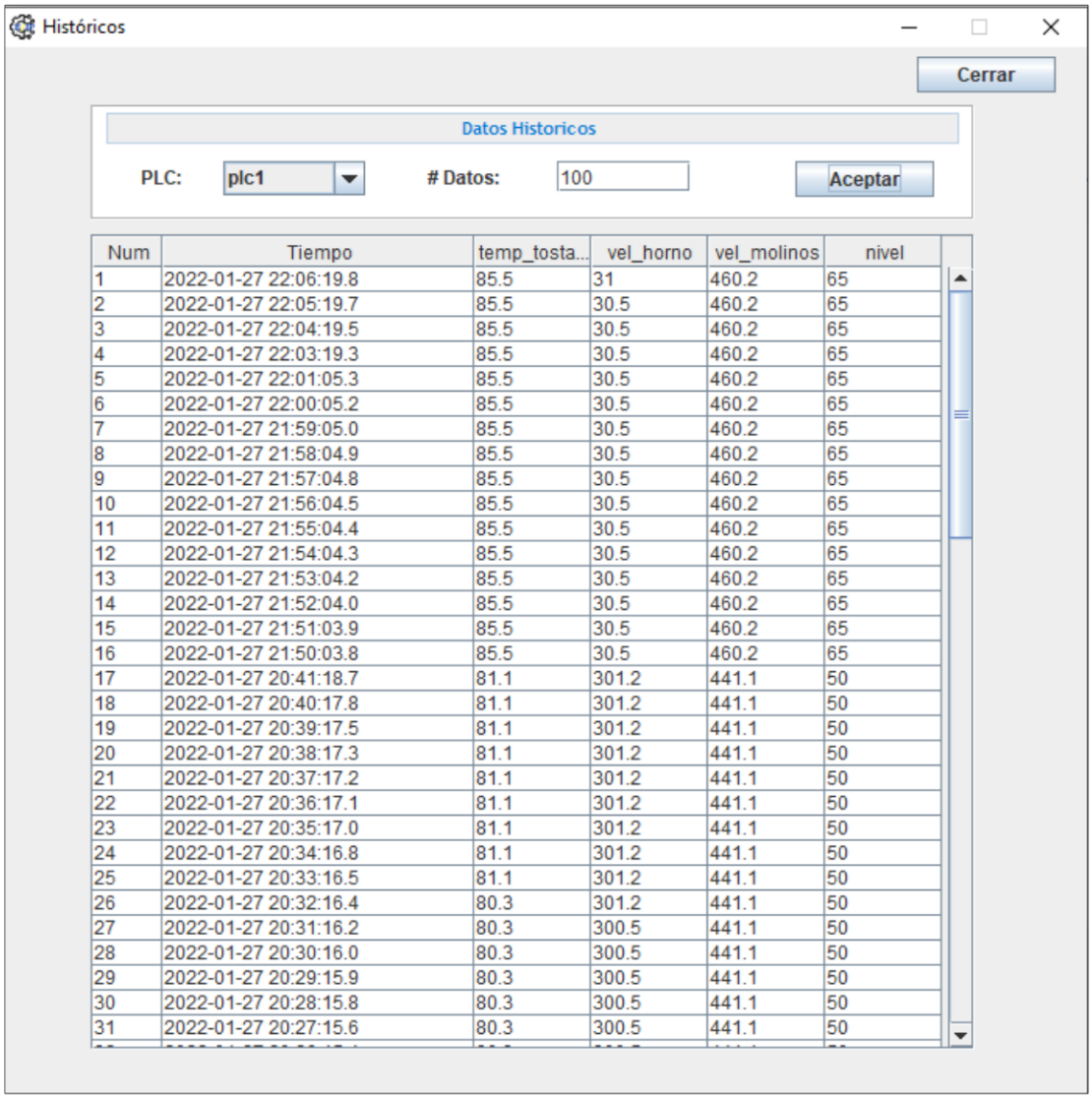
	idBOOLEAN	valor	idTagB	nombreTag
	1	0	5	status_horno
	2	1	6	status_compuerta1
	3	0	9	status_molino1
	4	0	10	status_blower
	5	0	11	status_molino2

	idENTERO	valor	idTagE	nombreTag
	1	50	7	nivel
	2	45	23	sp_nivel

	idFLOTANTE	valor	idTagF	nombreTag
	1	80.5	2	sp_temp_tostado
▶	2	81.3	3	temp_tostado
	3	300	4	vel_horno
	4	440	22	vel_molinos

Figura 3.6. Variables del tipo bool, entero y flotante destinadas para escritura.

En el modo automático también se realiza el registro de históricos minuto a minuto de las variables seleccionadas para hacer el registro. En la Figura 3.7 se observa en la ventana de históricos los últimos 100 registros guardados.



Num	Tiempo	temp_tosta...	vel_horno	vel_molinos	nivel
1	2022-01-27 22:06:19.8	85.5	31	460.2	65
2	2022-01-27 22:05:19.7	85.5	30.5	460.2	65
3	2022-01-27 22:04:19.5	85.5	30.5	460.2	65
4	2022-01-27 22:03:19.3	85.5	30.5	460.2	65
5	2022-01-27 22:01:05.3	85.5	30.5	460.2	65
6	2022-01-27 22:00:05.2	85.5	30.5	460.2	65
7	2022-01-27 21:59:05.0	85.5	30.5	460.2	65
8	2022-01-27 21:58:04.9	85.5	30.5	460.2	65
9	2022-01-27 21:57:04.8	85.5	30.5	460.2	65
10	2022-01-27 21:56:04.5	85.5	30.5	460.2	65
11	2022-01-27 21:55:04.4	85.5	30.5	460.2	65
12	2022-01-27 21:54:04.3	85.5	30.5	460.2	65
13	2022-01-27 21:53:04.2	85.5	30.5	460.2	65
14	2022-01-27 21:52:04.0	85.5	30.5	460.2	65
15	2022-01-27 21:51:03.9	85.5	30.5	460.2	65
16	2022-01-27 21:50:03.8	85.5	30.5	460.2	65
17	2022-01-27 20:41:18.7	81.1	301.2	441.1	50
18	2022-01-27 20:40:17.8	81.1	301.2	441.1	50
19	2022-01-27 20:39:17.5	81.1	301.2	441.1	50
20	2022-01-27 20:38:17.3	81.1	301.2	441.1	50
21	2022-01-27 20:37:17.2	81.1	301.2	441.1	50
22	2022-01-27 20:36:17.1	81.1	301.2	441.1	50
23	2022-01-27 20:35:17.0	81.1	301.2	441.1	50
24	2022-01-27 20:34:16.8	81.1	301.2	441.1	50
25	2022-01-27 20:33:16.5	81.1	301.2	441.1	50
26	2022-01-27 20:32:16.4	80.3	301.2	441.1	50
27	2022-01-27 20:31:16.2	80.3	300.5	441.1	50
28	2022-01-27 20:30:16.0	80.3	300.5	441.1	50
29	2022-01-27 20:29:15.9	80.3	300.5	441.1	50
30	2022-01-27 20:28:15.8	80.3	300.5	441.1	50
31	2022-01-27 20:27:15.6	80.3	300.5	441.1	50

Figura 3.7. Registro histórico de tags de plc1.

	id	tiempo	temp_tostado	vel_horno	vel_molinos	nivel
	127	2022-01-27 20:39:17.5	81.1	301.2	441.1	50
	128	2022-01-27 20:40:17.8	81.1	301.2	441.1	50
	129	2022-01-27 20:41:18.7	81.1	301.2	441.1	50
	131	2022-01-27 21:50:03.8	85.5	30.5	460.2	65
	132	2022-01-27 21:51:03.9	85.5	30.5	460.2	65
	133	2022-01-27 21:52:04.0	85.5	30.5	460.2	65
	134	2022-01-27 21:53:04.2	85.5	30.5	460.2	65
	135	2022-01-27 21:54:04.3	85.5	30.5	460.2	65
	136	2022-01-27 21:55:04.4	85.5	30.5	460.2	65
	137	2022-01-27 21:56:04.5	85.5	30.5	460.2	65
	138	2022-01-27 21:57:04.8	85.5	30.5	460.2	65
	139	2022-01-27 21:58:04.9	85.5	30.5	460.2	65
	140	2022-01-27 21:59:05.0	85.5	30.5	460.2	65
	141	2022-01-27 22:00:05.2	85.5	30.5	460.2	65
	142	2022-01-27 22:01:05.3	85.5	30.5	460.2	65
	143	2022-01-27 22:03:19.3	85.5	30.5	460.2	65
	144	2022-01-27 22:04:19.5	85.5	30.5	460.2	65
	145	2022-01-27 22:05:19.7	85.5	30.5	460.2	65
	146	2022-01-27 22:06:19.8	85.5	31	460.2	65

Figura 3.8. Registro histórico de plc1 en MySQL.

El software desarrollado también fue testeado en Ubuntu 20.04 y se comprueba su funcionamiento, conexión y adquisición de datos con el PLC. En este sistema operativo se instaló java y MySQL para realizar las pruebas correspondientes, el comando que se ejecutó para abrir el programa es:

```
alisson@alisson-virtual-machine:~/Documents$ java -jar ServidorEIP.jar
```

En las siguientes figuras se observa el software funcionando en el sistema operativo de Linux.

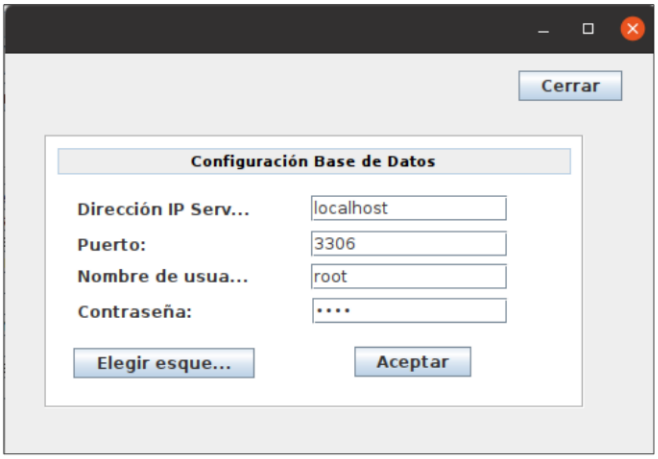


Figura 3.9. Ventana de configuración de base de datos en Ubuntu.

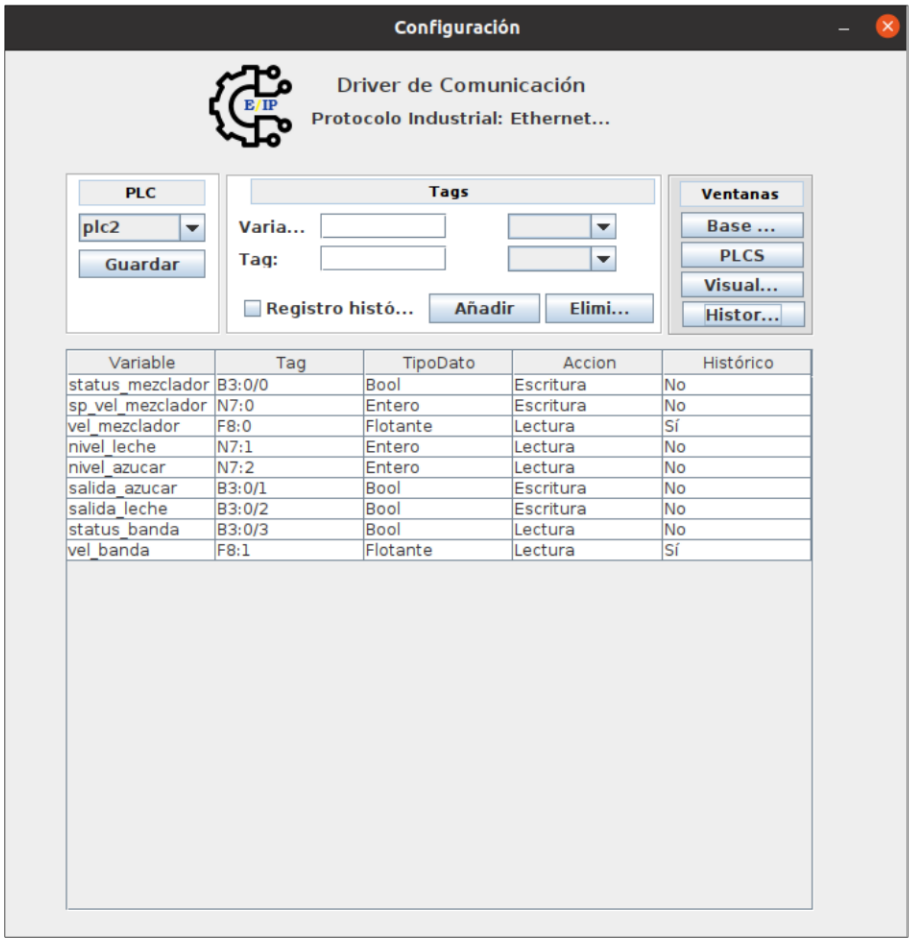


Figura 3.10. Ventana de configuración en Ubuntu.

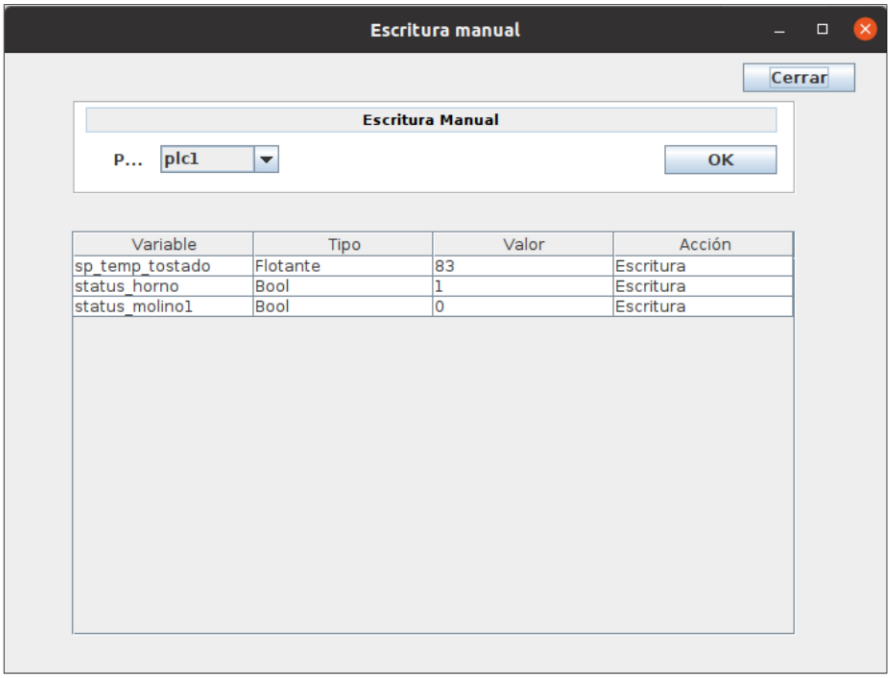


Figura 3.11. Ventana de escritura manual en Ubuntu.

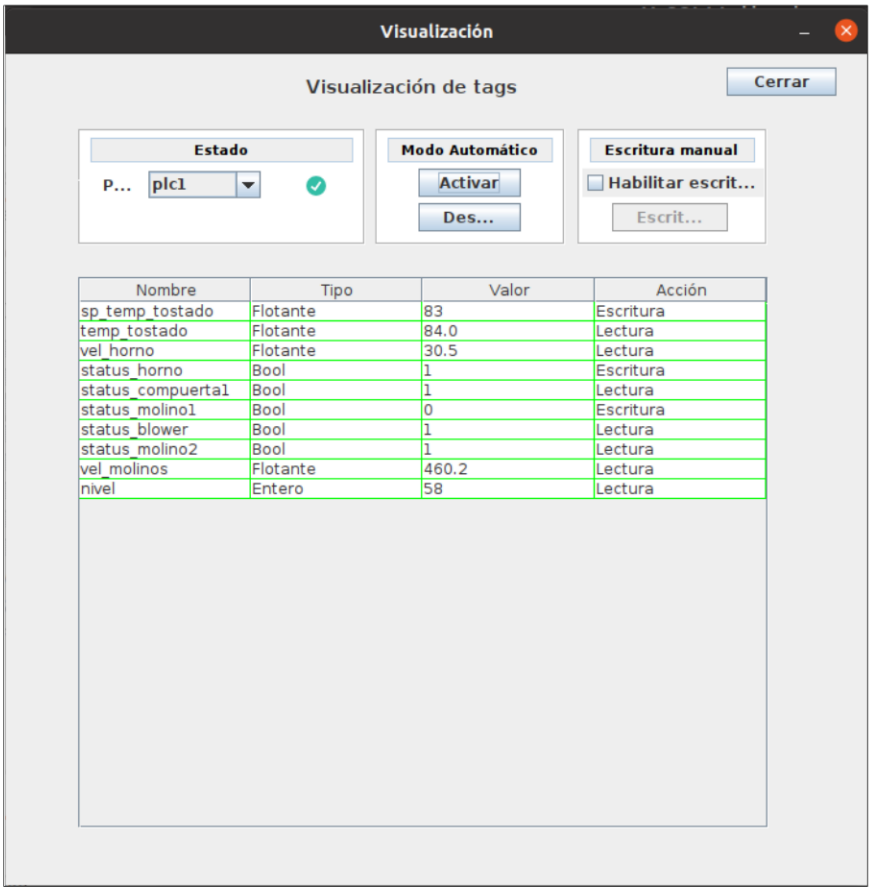


Figura 3.12. Ventana de visualización en Ubuntu.

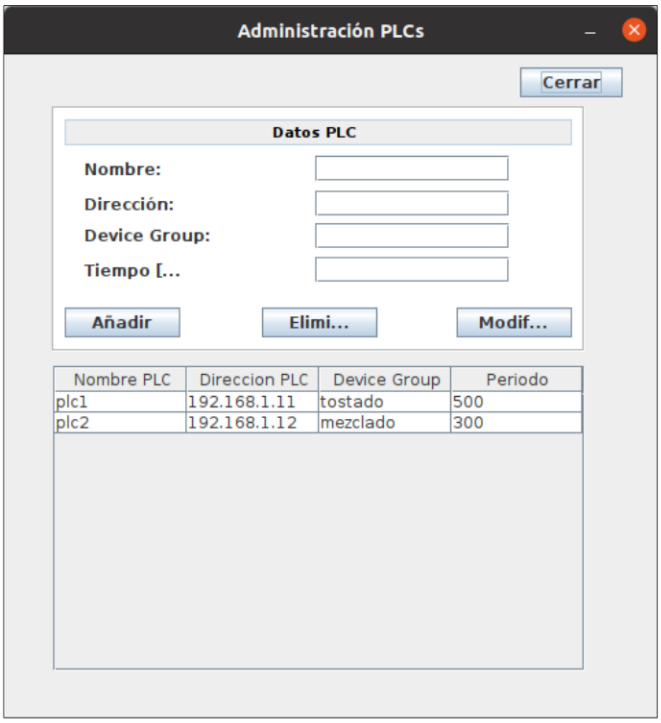
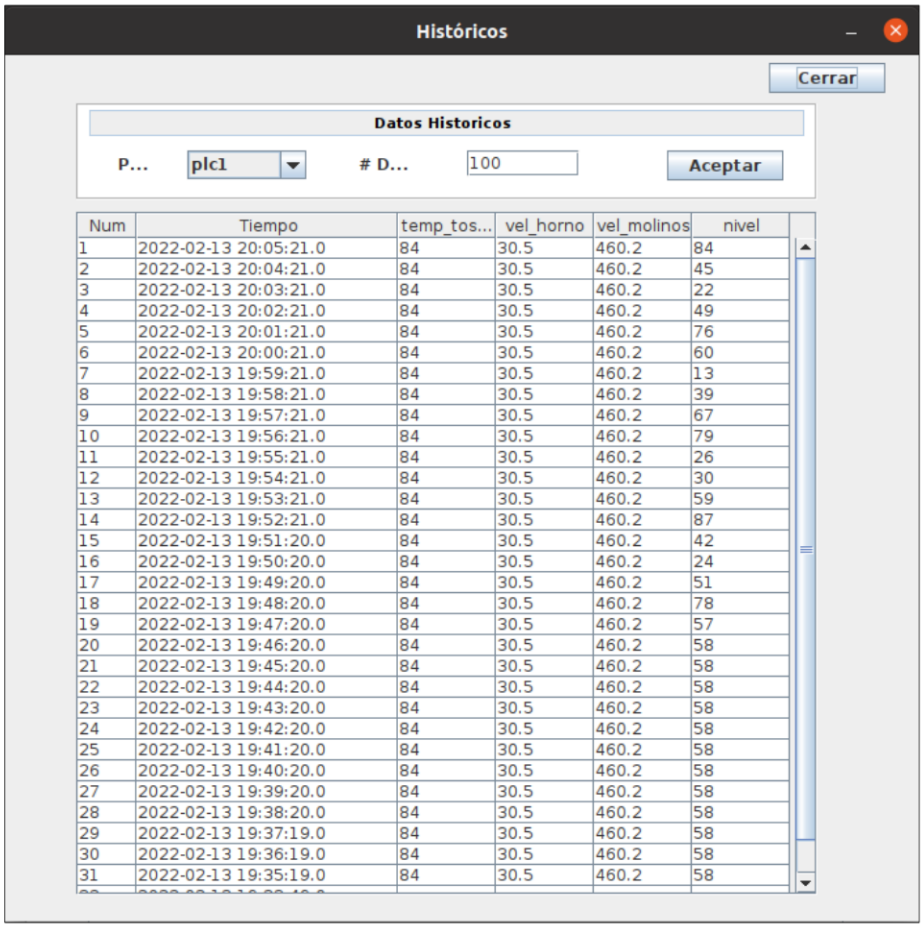


Figura 3.13. Ventana de administración de PLCs en Ubuntu.



Num	Tiempo	temp_tos...	vel_horno	vel_molinos	nivel
1	2022-02-13 20:05:21.0	84	30.5	460.2	84
2	2022-02-13 20:04:21.0	84	30.5	460.2	45
3	2022-02-13 20:03:21.0	84	30.5	460.2	22
4	2022-02-13 20:02:21.0	84	30.5	460.2	49
5	2022-02-13 20:01:21.0	84	30.5	460.2	76
6	2022-02-13 20:00:21.0	84	30.5	460.2	60
7	2022-02-13 19:59:21.0	84	30.5	460.2	13
8	2022-02-13 19:58:21.0	84	30.5	460.2	39
9	2022-02-13 19:57:21.0	84	30.5	460.2	67
10	2022-02-13 19:56:21.0	84	30.5	460.2	79
11	2022-02-13 19:55:21.0	84	30.5	460.2	26
12	2022-02-13 19:54:21.0	84	30.5	460.2	30
13	2022-02-13 19:53:21.0	84	30.5	460.2	59
14	2022-02-13 19:52:21.0	84	30.5	460.2	87
15	2022-02-13 19:51:20.0	84	30.5	460.2	42
16	2022-02-13 19:50:20.0	84	30.5	460.2	24
17	2022-02-13 19:49:20.0	84	30.5	460.2	51
18	2022-02-13 19:48:20.0	84	30.5	460.2	78
19	2022-02-13 19:47:20.0	84	30.5	460.2	57
20	2022-02-13 19:46:20.0	84	30.5	460.2	58
21	2022-02-13 19:45:20.0	84	30.5	460.2	58
22	2022-02-13 19:44:20.0	84	30.5	460.2	58
23	2022-02-13 19:43:20.0	84	30.5	460.2	58
24	2022-02-13 19:42:20.0	84	30.5	460.2	58
25	2022-02-13 19:41:20.0	84	30.5	460.2	58
26	2022-02-13 19:40:20.0	84	30.5	460.2	58
27	2022-02-13 19:39:20.0	84	30.5	460.2	58
28	2022-02-13 19:38:20.0	84	30.5	460.2	58
29	2022-02-13 19:37:19.0	84	30.5	460.2	58
30	2022-02-13 19:36:19.0	84	30.5	460.2	58
31	2022-02-13 19:35:19.0	84	30.5	460.2	58

Figura 3.14. Ventana de históricos en Ubuntu.

3.3. IMPLEMENTACIÓN DE PROTOTIPO

3.3.1. Descripción de Proceso

En ecuador se tienen plantaciones de cacao de excelente calidad que se utilizan para la preparación y producción de chocolate, este proceso cuenta con las etapas que se detallan a continuación:

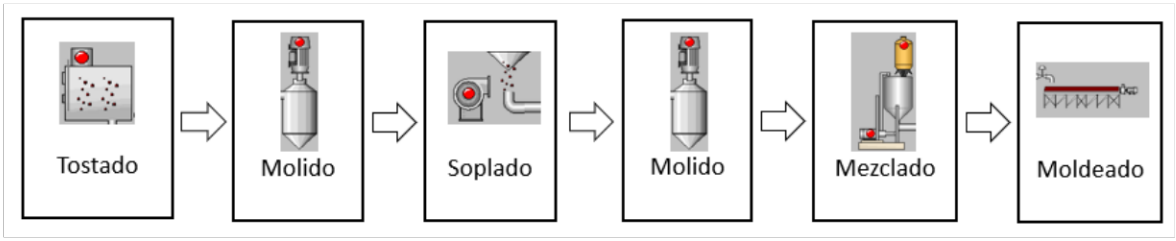


Figura 3.15. Etapas de preparación y producción de cacao.

- Etapa de tostado: en esta etapa se coloca el cacao puro y seco en el interior de un horno giratorio con el objetivo de tostarlo uniformemente para que su cáscara se desprenda fácilmente. La etapa de tostado tiene una duración de 45 minutos a una temperatura dentro del rango de 80°C – 90°C.
- Etapa de molido: en esta etapa se tritura levemente el cacao para separar la cascara del producto, este proceso se lleva a cabo después de haber sido tostado el cacao.
- Etapa de soplado: en esta etapa se utiliza un soplador que separa completamente el cacao de la cáscara, aprovechando la diferencia de densidad para obtener un producto listo para ser utilizado.
- Segunda etapa de molido: en esta etapa se introduce el cacao dentro de un molino que lo tritura finamente para obtener una pasta blanda uniforme.
- Etapa de mezclado: en esta etapa se agrega la pasta obtenida, azúcar y lácteos dentro de un tanque de mezclado para posteriormente colocarlo en moldes que son transportados por una banda transportadora vibratoria.

La supervisión y control requiere del conocimiento del estado del proceso, por tanto, se precisa de adquisición y visualización de variables, como también la posibilidad de cambiar la referencia de alguna variable en específico desde la interfaz de usuario.

Por consiguiente, el prototipo desarrollado se utiliza para gestionar la conexión a dos dispositivos Micrologix para simular el proceso de producción de chocolate de Ambato y visualizarlo en una interfaz HMI creada en Intouch.

3.3.2. Subproceso por PLC

El primer dispositivo “plc1” es un Micrologix 1100 serie B que controla las etapas de tostado, molido, soplado y segunda etapa de molido. Este PLC requiere de las variables que se observan en la Tabla 3.2, cada elemento tiene su dirección correspondiente en el PLC y el nombre asignado a la etiqueta en Intouch.

Tabla 3.2. Mapa de direcciones de PLC 1.

VARIABLE	TIPO TAG	DIRECCIÓN PLC	ETIQUETA HMI
sp_temp_tostado	FLOAT	F8:3	sp_temp_tostado
temp_tostado	FLOAT	F8:0	temp_tostado
vel_horno	FLOAT	F8:1	vel_horno

status_horno	BOOL	B3:0/0	status_horno
status_compuerta1	BOOL	B3:0/1	status_compuerta1
nivel	INT	N7:0	nivel
status_molino1	BOOL	B3:0/2	status_molino1
status_blower	BOOL	B3:0/3	status_blower
status_molino2	BOOL	B3:0/4	status_molino2
vel_molinos	FLOAT	F8:2	vel_molinos

El segundo dispositivo “plc2” es un Micrologix 1100 serie A que controla las etapas de mezclado y moldeado, las cuales requieren de las variables que se observan en la Tabla 3.3.

Tabla 3.3. Mapa de direcciones de PLC 2.

VARIABLE	TIPO TAG	DIRECCIÓN PLC	ETIQUETA HMI
status_motor	BOOL	B3:0/0	status_motor
SP_vel_motor	INT	N7:0	SP_vel_motor
vel_motor	FLOAT	F8:0	vel_motor
nivel_leche	INT	N7:1	nivel_leche
nivel_azucar	INT	N7:2	nivel_azucar
salida_azucar	BOOL	B3:0/1	salida_azucar
salida_leche	BOOL	B3:0/2	salida_leche
status_banda	BOOL	B3:0/3	status_banda
velocidad_banda	FLOAT	F8:1	velocidad_banda

3.3.2. Simulación de proceso

Se realizan pruebas simulando el proceso de preparación y producción de chocolate de Ambato, para lo cual se desarrolla una interfaz HMI en Intouch.

La primera pantalla permite visualizar el estado de las etapas de tostado, molido y soplado como se observa en la Figura 3.16. La interfaz lee la información por medio de un script con sentencias SQL de lectura y escritura que se ejecuta cada 100 ms. Se lee temperatura, velocidad del horno, velocidad de molinos, nivel, estados de molinos, soplador y compuerta, también se realiza la escritura del setpoint de la temperatura de tostado y el estado del molino 1. En la Figura 3.17 se observan los valores de estas variables obtenidas por el driver de comunicación, que coinciden con las visualizadas en la primera pantalla de proceso.

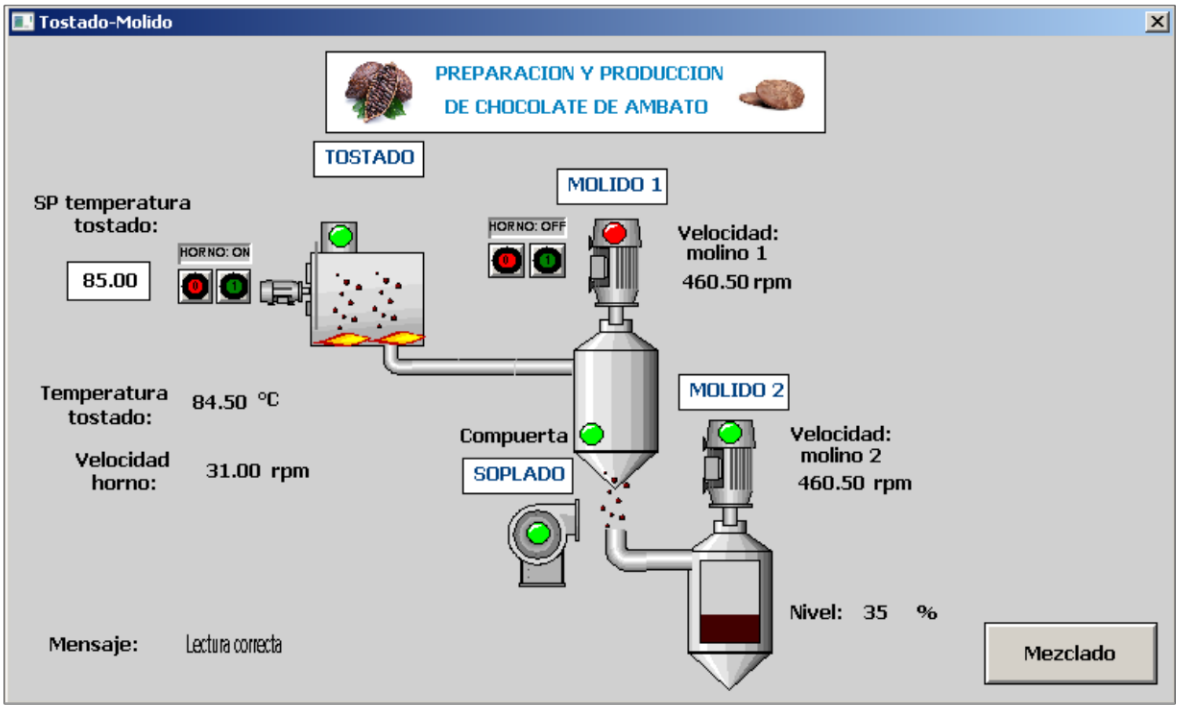
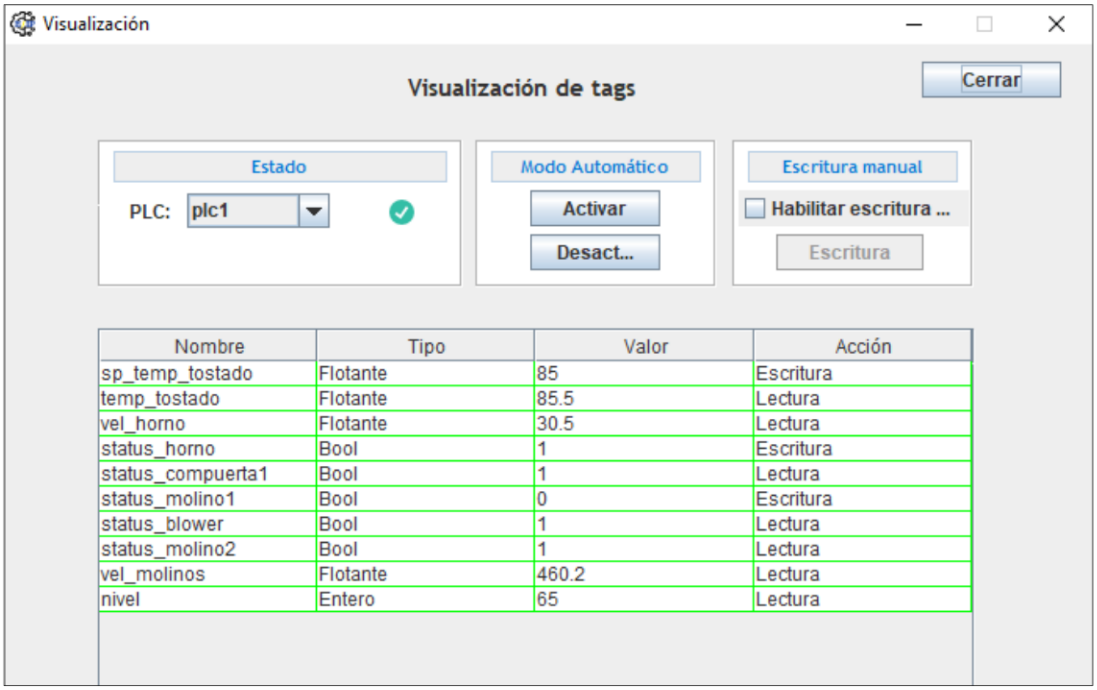


Figura 3.16. Pantalla de proceso de tostado y molido.



Nombre	Tipo	Valor	Acción
sp_temp_tostado	Flotante	85	Escritura
temp_tostado	Flotante	85.5	Lectura
vel_horno	Flotante	30.5	Lectura
status_horno	Bool	1	Escritura
status_compuerta1	Bool	1	Lectura
status_molino1	Bool	0	Escritura
status_blower	Bool	1	Lectura
status_molino2	Bool	1	Lectura
vel_molinos	Flotante	460.2	Lectura
nivel	Entero	65	Lectura

Figura 3.17. Variables de etapa de tostado y molido.

En la segunda pantalla que se visualiza en la Figura 3.18 se observan los subprocesos de mezclado, moldeado y dosificación de leche y azúcar, en los cuales se requiere la adquisición de la velocidad del mezclador, velocidad de la banda transportadora, nivel de leche, nivel de azúcar y estado de la banda, además se requiere escribir en el plc el setpoint de la velocidad del mezclador y habilitar la etapa de mezclado, dosificación de leche y

dosificación de azúcar. El cambio de los valores en la interfaz HMI destinados a escritura se reflejan en el servidor de datos que se observa en la Figura 3.19.

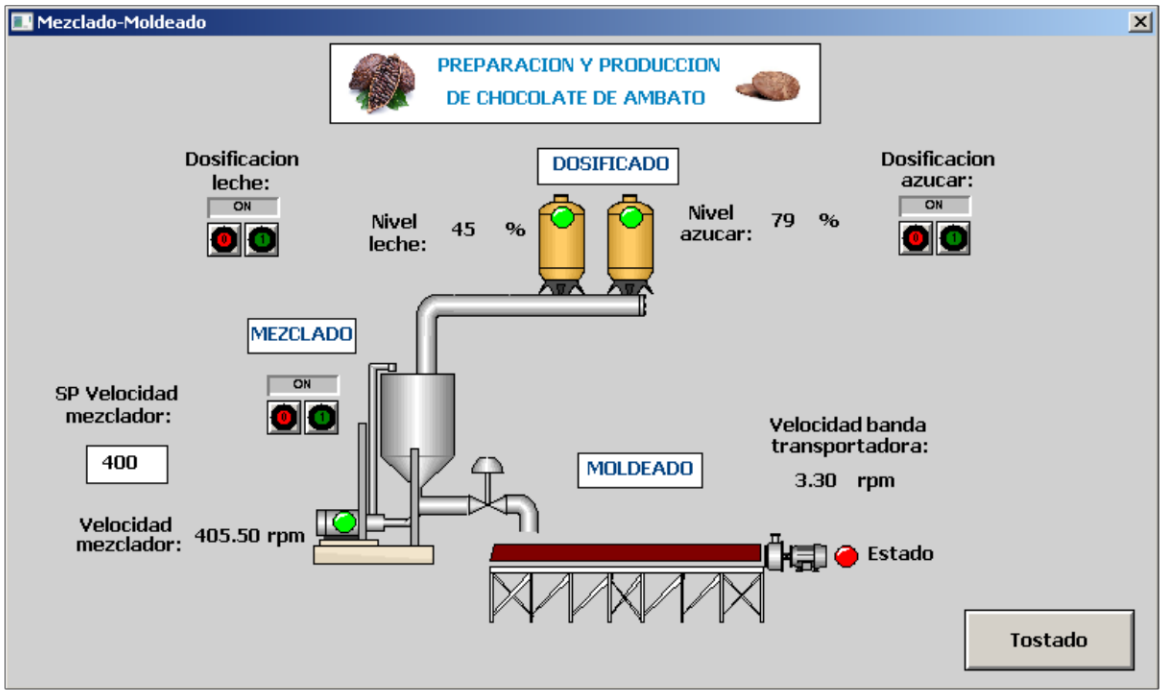


Figura 3.18. Pantalla de proceso de mezclado y moldeado.

Nombre	Tipo	Valor	Acción
status_mezclador	Bool	1	Escritura
sp_vel_mezclador	Entero	400	Escritura
vel_mezclador	Flotante	405.5	Lectura
nivel_leche	Entero	45	Lectura
nivel_azucar	Entero	79	Lectura
salida_azucar	Bool	1	Escritura
salida_leche	Bool	1	Escritura
status_banda	Bool	0	Lectura
vel_banda	Flotante	3.3	Lectura

Figura 3.19. Variables de etapa de mezclado y moldeado.

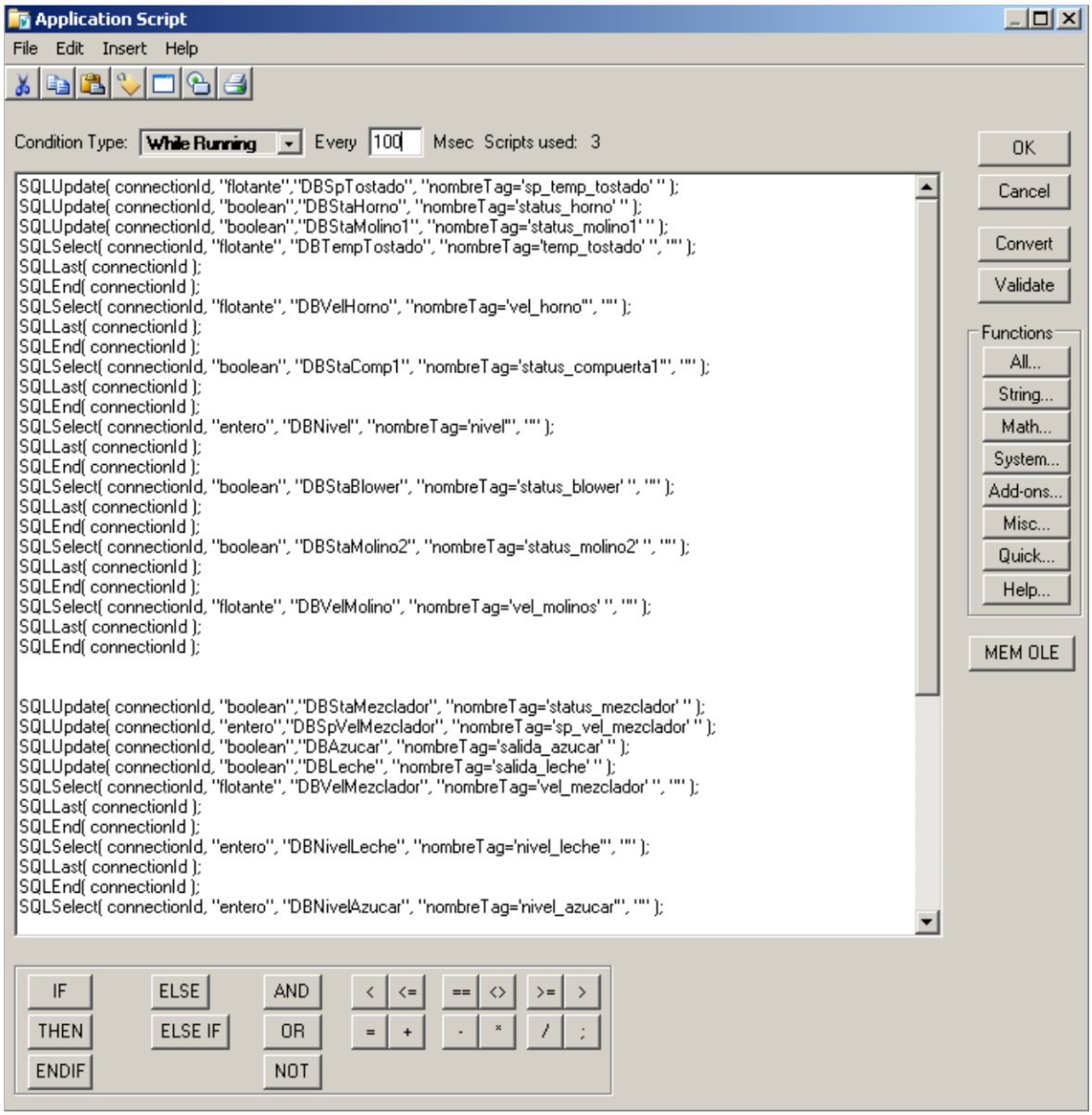


Figura 3.20. Script en Intouch con sentencias SQL de lectura y escritura.

3.4. COMPARACIÓN DE SERVIDOR CON DASABCIP

En esta sección se realiza una comparación del software desarrollado funcionando en modo automático con el servidor de datos comercial DASABCIP desarrollado por Wonderware, el cual se encarga de la comunicación de PLCs de Allen Bradley bajo el protocolo CIP. En la tabla se visualiza un cuadro comparativo con las principales características.

Tabla 3.4. Cuadro comparativo de DASABCIP y Servidor EIP desarrollado.

CARACTERÍSTICA	DASABCIP	Servidor EIP desarrollado
Plataforma	Windows	Windows y Linux

Conexión a PLC	Se realiza a través de dirección IP	Se realiza a través de dirección IP
Conexión con otras aplicaciones	Por medio de DDE, OPC y SuiteLink.	Por medio de base de datos MySQL
Tiempo de espera por PLC	Valor configurable (2000 ms por defecto)	Valor = 5000 ms
Tiempo de muestreo	Un solo valor para todos los PLCs	Valor configurable para cada uno de los PLCs
Licencia	Licencia pagada	Licencia gratuita
Device Group por PLC	Varios	Uno
Redes	ControlNet DeviceNet Ethernet/IP	Ethernet/IP
Permite enlazar ítems con la dirección de tag	Sí, permite añadir	Sí, permite añadir
Número máximo de PLCs	PLCs>16	PLC = 16
Número máximo de ítems	Se admiten más de 32 tags por PLC	Se admiten 32 tags por PLC
Visualización	Se visualizan: dirección del tag, valor, tiempo, código de error y estado.	Amigable con el usuario: nombre del tag, tipo de tag, valor y acción.
PLCs soportados	ControlLogix CompactLogix MicroLogix PLC-5 SLC500	MicroLogix
Funcionamiento	Continuo bajo licencia pagada, 120 minutos con licencia temporal.	Continuo

Uno de los requerimientos de este proyecto es entregar un software que cumpla con las funciones de la Tabla 3.4 y que no demande una alta inversión para que pueda ser accesible para las pequeñas y medianas empresas, por tanto, en las siguientes tablas se realiza una descripción de puntos importantes para el desarrollo del software y su respectiva inversión económica, para finalmente determinar el costo del producto final.

Tabla 3.5. Costo de herramientas utilizadas para el desarrollo del software.

Descripción	Software	Precio
Pruebas de comunicación entre PLCs para identificación de tramas	Wireshark	$0
Driver de comunicación Ethernet/IP	IDE de Java: Eclipse	$0
Base de datos	MySQL 8.0	$0
Visualización de datos obtenidos en interfaces industriales	Aplicación de visualización de HMI.	El costo depende del software escogido para visualizar la información obtenida.

Los programas utilizados para el desarrollo del servidor son gratuitos, por tanto, el costo de recursos de software es nulo. En el caso del programa de visualización HMI el valor económico depende del software que elija el usuario.

Tabla 3.6. Costo del desarrollo e instalación del software.

Actividad	Descripción	Precio
Desarrollo de servidor de datos industrial	Se determinan las horas invertidas para obtener el costo de mano de obra. Horas: 400 Costo por hora: $6	$2400
Instalación	Seguimiento de indicaciones del manual por parte del usuario.	$0

Con respecto al costo del trabajo del desarrollador de software, se determina en función del tiempo invertido y el costo por hora, cabe recalcar que este costo es cubierto por la autora del presente trabajo de titulación como parte del desarrollo del proyecto. La instalación y funcionamiento del software son detallados en el manual de usuario, el cual no tiene costo, por tanto, considerando los recursos de software y la mano de obra el software es totalmente gratuito.

4. CONCLUSIONES Y RECOMENDACIONES

A partir del trabajo y pruebas realizadas en las secciones anteriores se obtienen las siguientes conclusiones y recomendaciones

4.1. CONCLUSIONES

- Se analizó la comunicación entre dos PLCs con protocolo Ethernet/IP con la finalidad de identificar los campos más representativos para la comunicación, así como también sus principales características, a partir de este análisis se comprende el proceso de intercambio de comunicación para realizar tareas de lectura y escritura.

- Se seleccionó el lenguaje de programación JAVA para el desarrollo del servidor de datos, programación de driver de comunicación, manejo de base de datos y construcción de una interfaz de usuario, debido a que permite portabilidad y fácil ejecución del programa en cualquier plataforma.

- Se diseñó e implementó el driver de comunicación en base a requerimientos de desarrollo en software de código abierto, uso de protocolo industrial de comunicación Ethernet/IP, almacenamiento de información, baja inversión económica y funcionamiento continuo.

- Se diseñó e implementó una interfaz de usuario que permita la configuración de los parámetros que requiere el driver para la comunicación con el PLC y posibilite la visualización de la información obtenida del dispositivo. La interfaz cuenta con dos ventanas de configuración de PLCs y tags, una ventana de visualización, una ventana de escritura manual y una ventana para visualizar los registros históricos.

- Se diseñó e implementó una base de datos que permita el almacenamiento de información de configuración, escritura en la base de datos de los tags leídos desde el PLC, lectura de valores destinados a ser escritos en el PLC y almacenamiento de registro histórico cada minuto durante las últimas 24 horas.

- Se realizó una comparación del software desarrollado con el servidor de datos DASABCIP, validando su funcionamiento y confirmando que puede ser utilizado por pequeñas y medianas empresas para lectura y escritura de datos en dispositivos de campo como PLCs.

4.2. RECOMENDACIONES

- Se recomienda desarrollar drivers de comunicación con otros protocolos industriales como DeviceNet o ControlNet que son redes adaptadas para el protocolo CIP, de esta forma contar con un software robusto que abarque distintos protocolos industriales de comunicación.
- Se recomienda continuar el desarrollo del driver de comunicación bajo el protocolo Ethernet/IP para dispositivos lógicos programables de diferentes marcas.
- Se recomienda realizar una plataforma de software en código abierto para desarrollo de aplicaciones de control supervisorio que permita la construcción de interfaces de usuario industriales, además que sea capaz de leer o escribir en la base de datos MySQL y trabaje en conjunto con el driver de Ethernet/IP desarrollado.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] J. L. Val, “Industria 4.0: la transformación digital de la industria,” Facultad de Ingenieria Universidad de Deusto, 2016.

[2] J. Colomer, J. Ayza, and J. Meléndez, “Sistemas de Supervisión.” CEA-IFAC, 2000.

[3] A. Chesney, “Introduction to Ethernet,” *The Industrial Ethernet Book*, vol. 1, no. 3, p. 6, 2010.

[4] W. Stallings, *Comunicaciones y Redes de computadoras.*, Séptima. Madrid: Person Education, 2004.

[5] G. Thomas, “Introduction to the Internet Protocol,” *The extension*, vol. 1, no. 4, p. 6, 2010.

[6] Toad, “Introducción a TCP/IP.” Creative common licences, Feb. 2005. [Online]. Available: https://www.um.es/docencia/barzana/DIVULGACION/INFORMATICA/Introduccion_a_TCPIP.pdf

[7] F. A. C. Herías and J. P. Baeza, “Protocolos de Transporte TCP y UDP.” Universidad de Alicante, 2009. [Online]. Available: https://rua.ua.es/dspace/bitstream/10045/11606/1/Pr3-2009-10.pdf

[8] D. Vasko and O. Haya, “CIP Safety.” ODVA, Inc, Nov. 17, 2020. [Online]. Available: https://www.odva.org/wp-content/uploads/2020/11/PUB00110R4_CIP_Safety.pdf

[9] V. Schiffer, “Common Industrial Protocol (CIP™) and the Family of CIP Networks,” in *Industrial Communication Technology Handbook*, 2nd ed., R. Zurawski, Ed. CRC Press, 2017, pp. 9-1-9–100. doi: 10.1201/b17365-10.

[10] D. Collins, “What are explicit and implicit messaging in EtherNet/IP?,” *Motion Control Tips*, Apr. 23, 2020. https://www.motioncontroltips.com/what-are-explicit-and-implicit-messaging-in-ethernet-ip/ (accessed Nov. 30, 2021).

[11] ODVA and ControlNet International, “EtherNet/IP Adaptation of CIP.” ODVA, Inc, Nov. 2007.

[12] J. Pimentel and G. Schneider, “Network Topology Protocol Change from ControlNetTM to Ethernet/IPTM for a Master Control Station in a Subsea Production System,” Universidad Tecnológica Federal de Paraná, Dec. 2016, p. 15. [Online]. Available: https://www.researchgate.net/publication/311517871_Network_Topology_Protocol_Change_from_ControlNet_to_EthernetIP_for_a_Master_Control_Station_in_a_Subsea_Production_System

[13] P. Sen, “JAVA Overview.” Tutorials Point, Mar. 26, 2018. [Online]. Available: https://www.tutorialspoint.com/java/java_tutorial.pdf

[14] J. Martinez, “Fundamentos de programacion en Java.” Univesidad Complutense de Madrid, 2011. [Online]. Available: https://www.tesuva.edu.co/phocadownloadpap/Fundamentos%20de%20programcion%20en%20Java.pdf

[15] S. Prasanna, “Eclipse: integrated development environment.” Tutorials Point, Feb. 20, 2018. [Online]. Available: https://www.tutorialspoint.com/eclipse/eclipse_tutorial.pdf

[16] Javatpoint, “Java Swing Tutorial - javatpoint,” *www.javatpoint.com*. https://www.javatpoint.com/java-swing (accessed Jan. 09, 2022).

[17] Oracle Corporation, “Worker Threads y SwingWorker,” Java Documentation. https://docs.oracle.com/javase/tutorial/uiswing/concurrency/worker.html (accessed Jan. 20, 2022).

[18] Kyle Hayes, “API · libplctag/libplctag Wiki,” *GitHub*, Nov. 07, 2020. https://github.com/libplctag/libplctag (accessed Jan. 09, 2022).

[19] IONOS, “¿Qué es un wrapper en programación?,” *IONOS Digitalguide*, Sep. 14, 2020. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-un-wrapper/ (accessed Jan. 09, 2022).

[20] Massimo Felici, “Diagrama de actividad.” School of informatics, 2009. [Online]. Available: https://www.inf.ed.ac.uk/teaching/courses/seoc/2009_2010/notes/10_notes.pdf

[21] Michael Haim, “UML Diagrama de clases.” Abelski, 2008.

[22] E. Derclaye, “What is a database?,” *The Journal of World Intellectual Property*, p. 18, Nov. 2005, doi: 10.1111/j.1747-1796.2002.tb00189.x.
[23] R. Paré, L. Casillas, D. Costal, M. Gilbert, C. Martín, and Perez, *Bases de datos*, Primera. UOC, 2005.
[24] Código Compilado, México. *Base de datos*, (Sep. 2015). [Centro de formación.]. Available: https://www.youtube.com/playlist?list=PLs1sXiNvW4OyJCZs5WR3OjPZTlIqNcvQi
[25] Oracle Corporation, “MySQL 8.0 Reference Manual,” 2021. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/preface.html
[26] Nishadha, “Ultimate Entity Relationship Diagram Tutorial (ER Diagrams),” *Creately*, Sep. 2021. https://creately.com/blog/diagrams/er-diagrams-tutorial/#benefits
[27] Invensys Systems, “Wonderware ABCIP DAServer User’s Guide,” *Wonderware Invensys*, p. 234, 2013.

ANEXO A: MANUAL DE USUARIO

DRIVER DE COMUNICACIÓN ETHERNET/IP

MANUAL DRIVER DE COMUNICACIÓN

La aplicación EIP desarrollada para uso industrial es un software que permite la conexión a PLCs Micrologix de la familia Allen Bradley para adquisición y escritura de datos a nivel de campo. Utiliza el protocolo de capas superiores CIP con la red Ethernet/IP para adquirir información y escribirla en la base de datos MySQL con el propósito de ser utilizada por un software de visualización HMI. Para el funcionamiento del programa se recomienda instalar:

- Java JDK 8
- MySQL 8.0: es el motor de la base de datos.
- MySQL Workbench 8.0: es una herramienta que permite la administración de base de datos.
- MySQL Connector/ODBC 8.0 Unicode Driver: es un driver que permite la conexión de la base de datos en MySQL con el software de visualización HMI.

Además, se requiere instalar el programa en una computadora con sistema operativo Windows 10 o Ubuntu y memoria RAM disponible mínima de 4GB. Cabe mencionar que los dispositivos deben encontrarse en red con la computadora que tiene el software.

1. CONFIGURACIÓN DE BASE DE DATOS

1.1. Creación de conexión y base de datos

Una vez instalado MySQL 8.0 se debe considerar la información de configuración ingresada en la instalación para crear la conexión en MySQL Workbench.

En el software MySQL Workbench se crea la conexión dando clic en el ícono de apertura de conexión que se muestra en la Figura 1.1, posteriormente aparece la ventana de la Figura 1.2 de configuración. El nombre queda a elección del usuario, el puerto por defecto es el 3306, se ingresa el nombre de usuario y contraseña configurados en la instalación para acceder a la conexión. Al terminar la configuración se da clic en “Test Connection” para verificar la conexión mediante un mensaje “Succesfully made the MySQL connection” y finalmente se da click en OK para crear la conexión.

Figura 1.1. Creación de conexión.

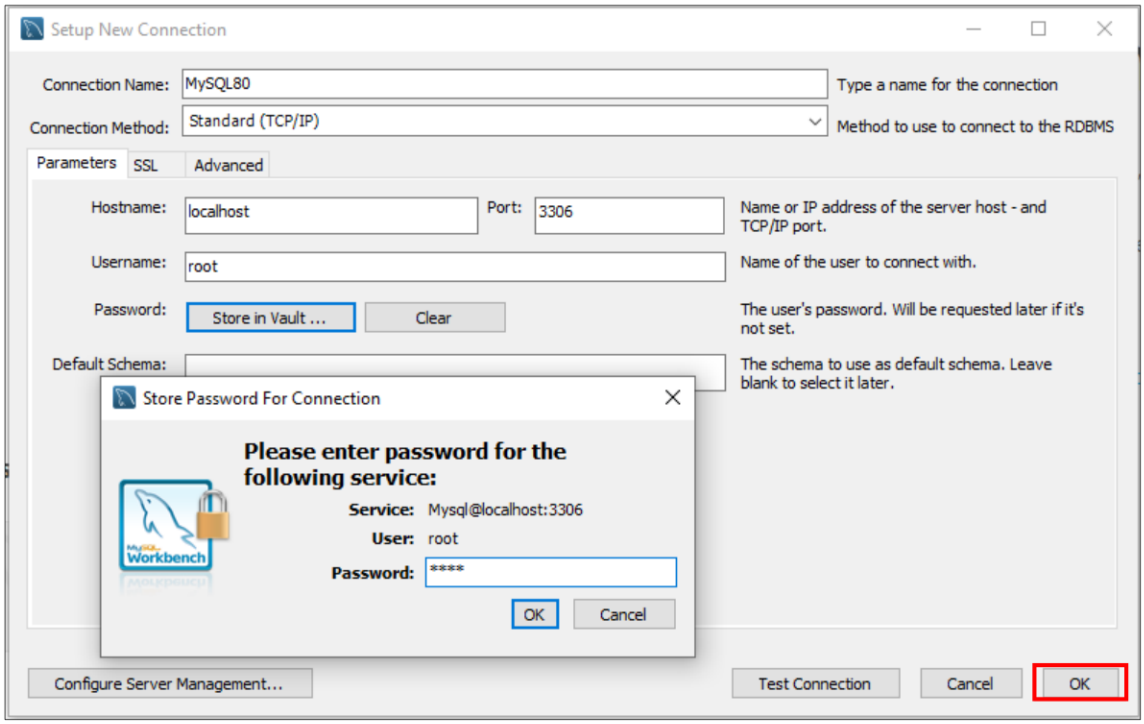


Figura 1.2. Configuración de nueva conexión.

Una vez creada, se ingresa a la conexión dando clic en el área señalada en la Figura 1.3 y se procede a crear la base de datos para lo cual se da clic en el ícono “Create a new schema in the connected server” como se observa en la Figura 1.4.

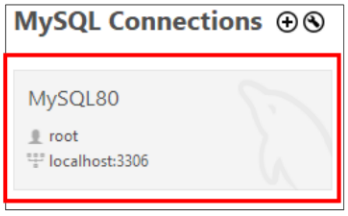


Figura 1.3. Ingreso a la conexión.

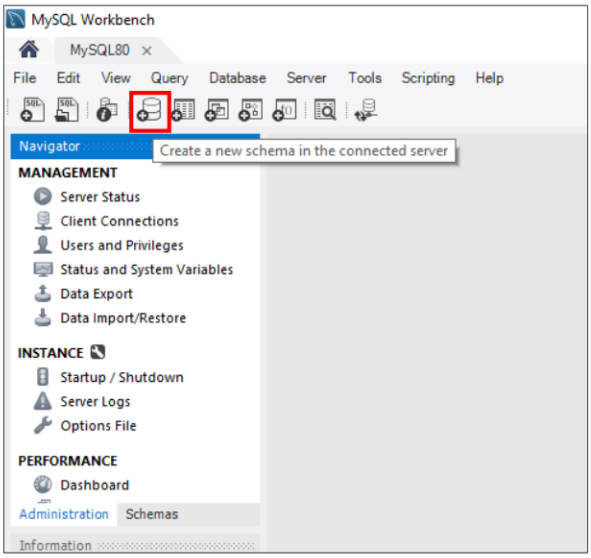


Figura 1.4. Ícono de creación de base de datos.

Al dar clic en el ícono aparecerá la pantalla de la Figura 1.5, en la cual se ingresa el nombre “bd_driver_eip” y se da clic en “Apply” para guardar los cambios efectuados, a continuación, aparece una ventana que indica las sentencias que se ejecutarán por lo cual se da clic en “Apply” y “Finish”.

Nota: El nombre de la base de datos debe ser estrictamente “bd_driver_eip” debido a que las configuraciones del driver se realizan considerando esta información.

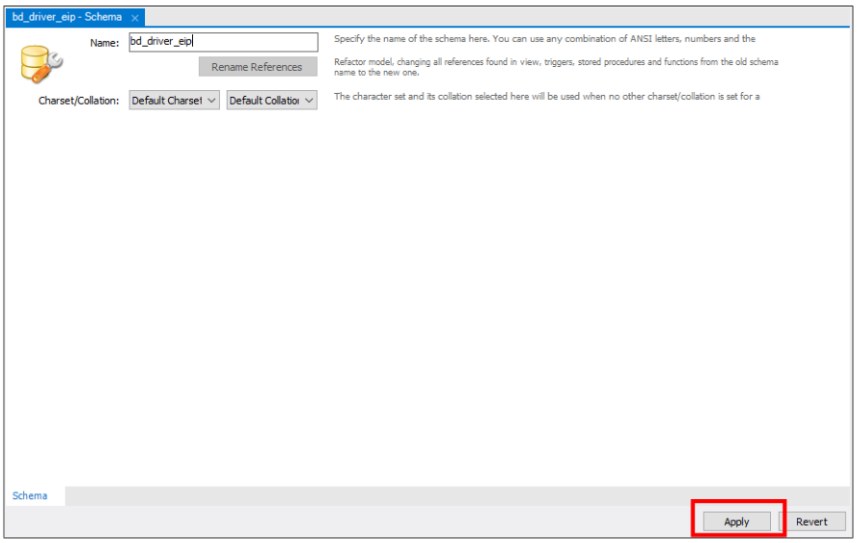


Figura 1.5. Creación base de datos bd_driver_eip.

1.2. Creación de usuario y permitir privilegios

Para el manejo de la base de datos desde una aplicación externa se requiere crear un usuario y habilitar distintos permisos de acceso a funciones de la base de datos. Se realizan los siguientes pasos para lograr este propósito.

1. En la sección de Administración seleccionar la opción “Users and Privileges” que habilita la pantalla que se visualiza en la Figura 1.6. y dar clic en el botón “Add account” para crear el usuario.

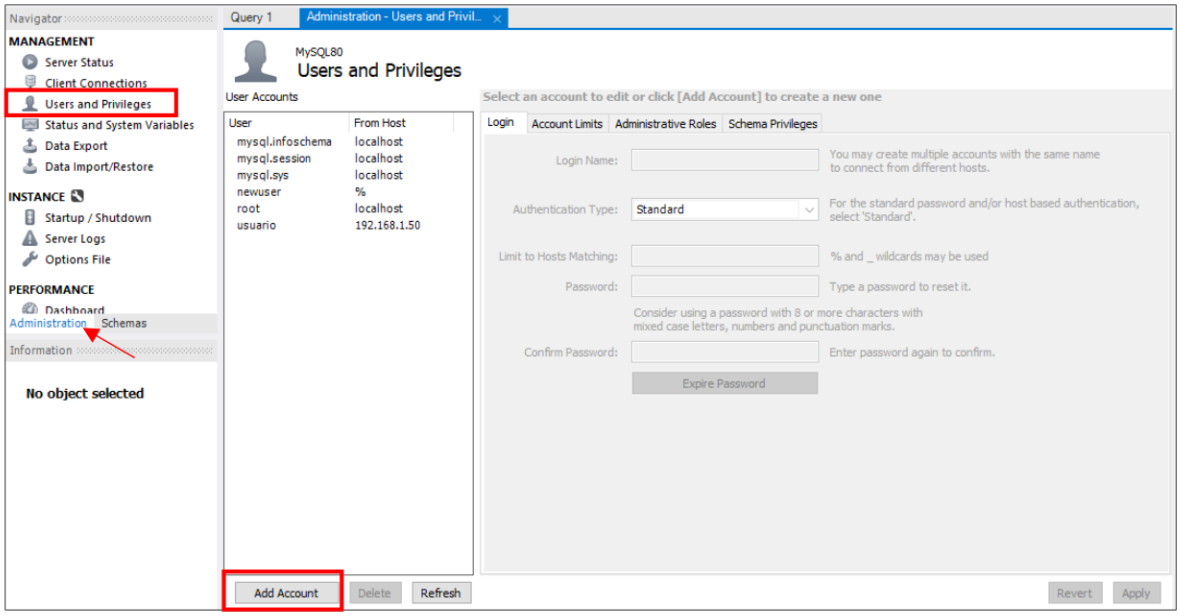


Figura 1.6. Manejo de usuarios y privilegios.

2. En la pantalla de la Figura 1.7 configurar la información de usuario, para lo cual, en la pestaña de “Login” ingresar el nombre “usuario”, la dirección IP de la computadora que contiene el programa que accederá a la base de datos y la contraseña “1234” y dar clic en “Apply”.

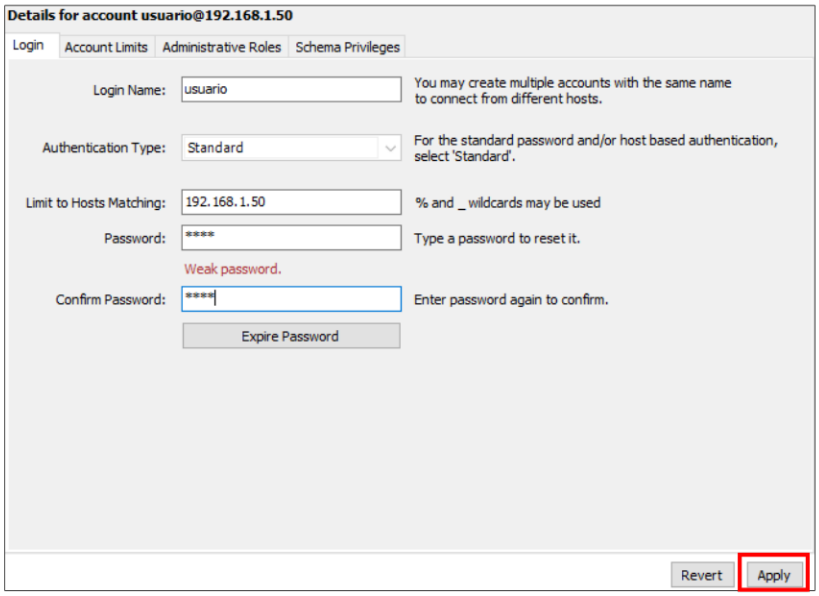


Figura 1.7. Configuración de la cuenta.

3. En la pestaña “Schema Privileges” dar clic en “Add Entry” y seleccionar el esquema bd_driver_eip, para agregar los permisos de acceso como se observa en la Figura 1.9.

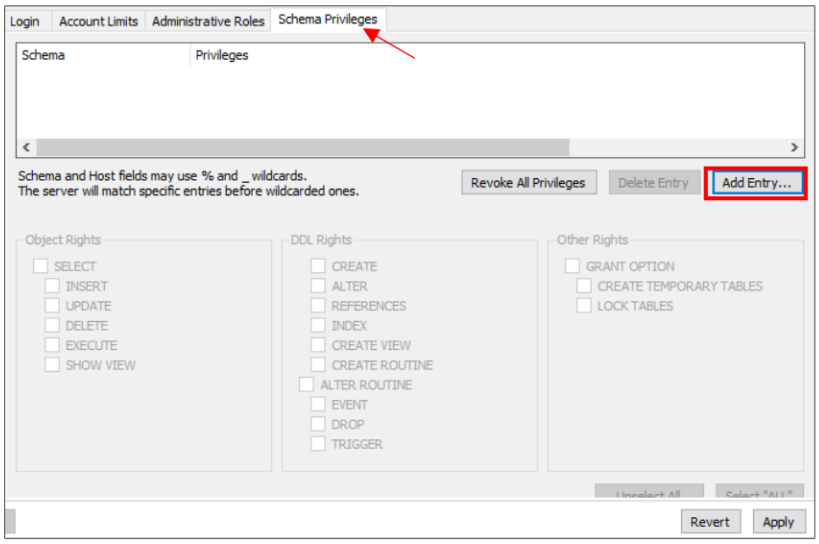


Figura 1.8. Pestaña de configuración de privilegios de usuario.

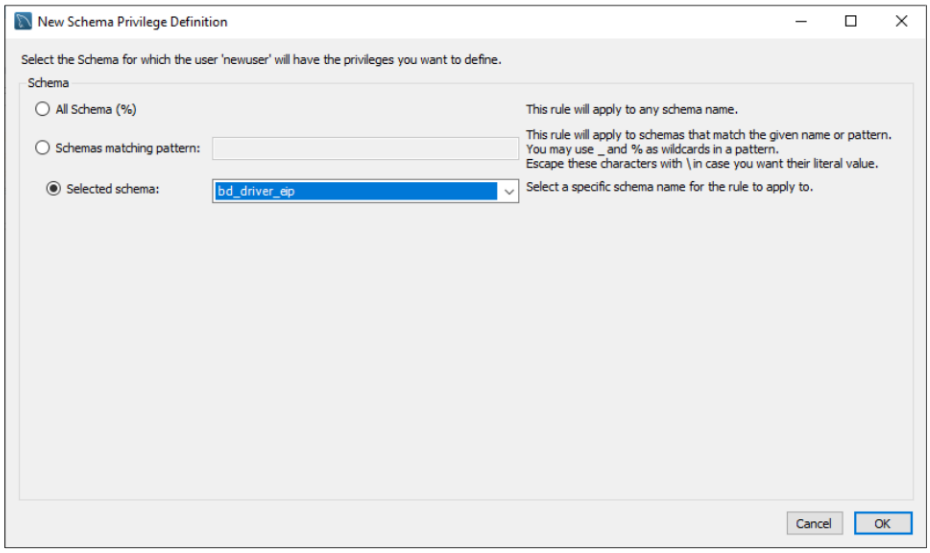


Figura 1.9. Selección de esquema.

4. Finalmente, en la parte inferior de la pestaña Schema Privileges seleccionar “Select ALL”, adicionalmente, en la pestaña Administrative Roles dar clic en DBA y presionar “Apply” para guardar los cambios efectuados.

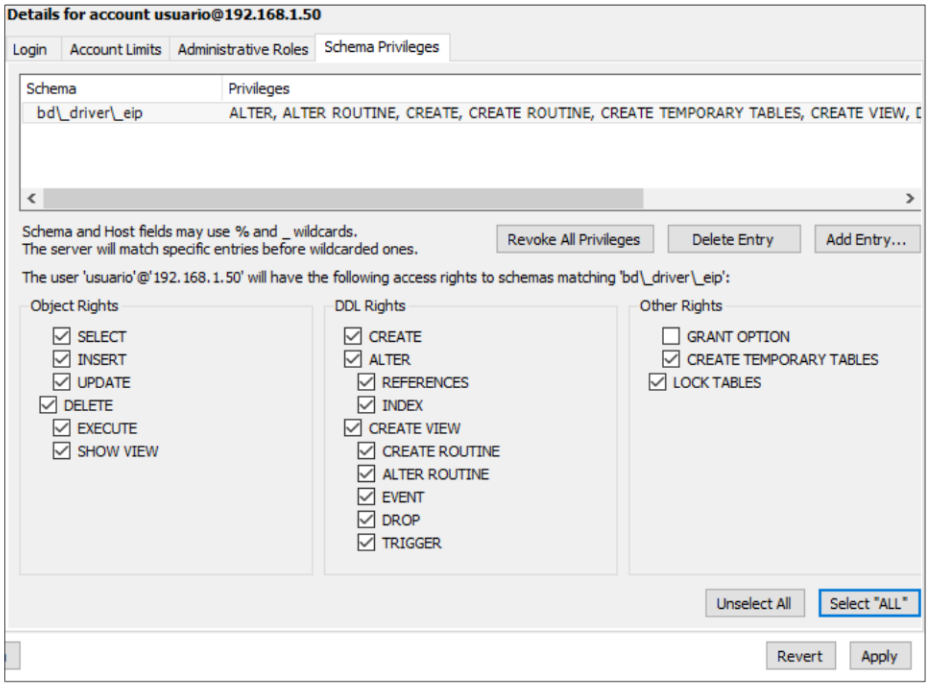


Figura 1.10. Selección de privilegios permitidos.

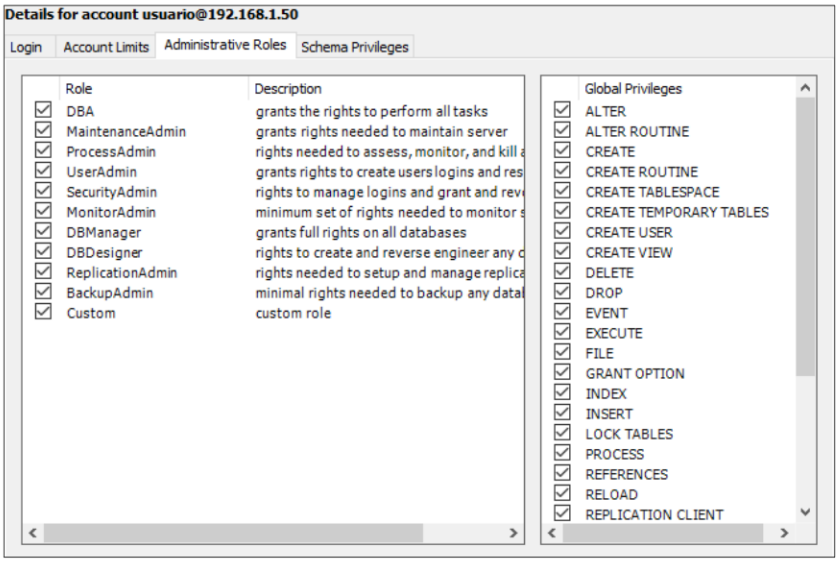
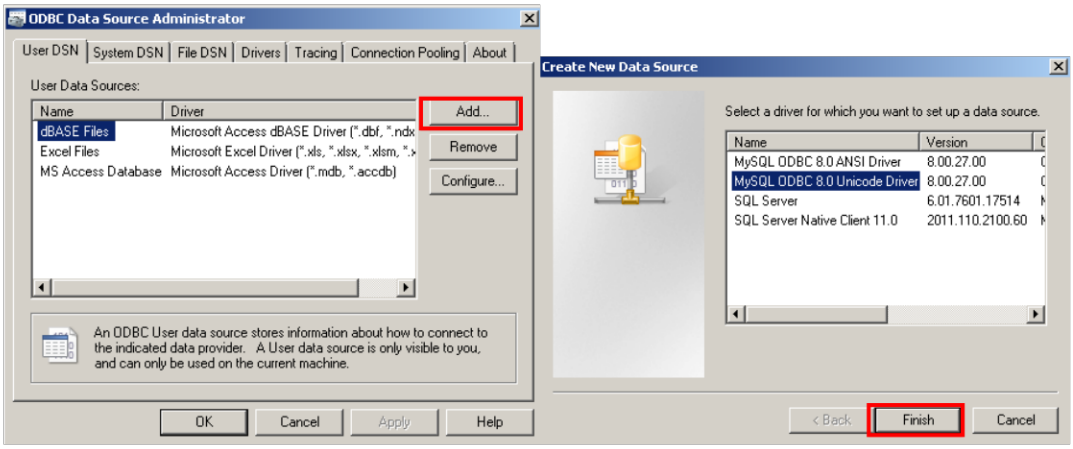


Figura 1.11. Configurar rol de administrador de base de datos.

2. CONECTOR ODBC

Considerando el uso de la máquina virtual de Intouch se requiere realizar las siguientes acciones para comunicar la interfaz con la base de datos. Cabe recalcar que la instalación y configuración se debe realizar en la PC que tenga el software HMI que requiere obtener la información de la base de datos.

1. Descargar e instalar MySQL Connector ODBC 8.0.27 X64 en la PC que contiene Intouch.
2. Abrir ODBC Data Source Administrator, se lo puede encontrar con el buscador de Windows.
3. Dar clic en el botón “Add” de la Figura 2.1 a) y aparecerá la pantalla de la Figura 2.1 b) en la cual se debe seleccionar MySQL ODBC 8.0 Unicode Driver y dar clic en “Finish”.



a) b)

Figura 2.1. Creación de controlador de comunicación para base de datos.

4. Configurar el conector con la información de la Figura 2.2, considerando el usuario y contraseña creados en la sección 1.2 y la dirección IP de la computadora donde se encuentra la base de datos.

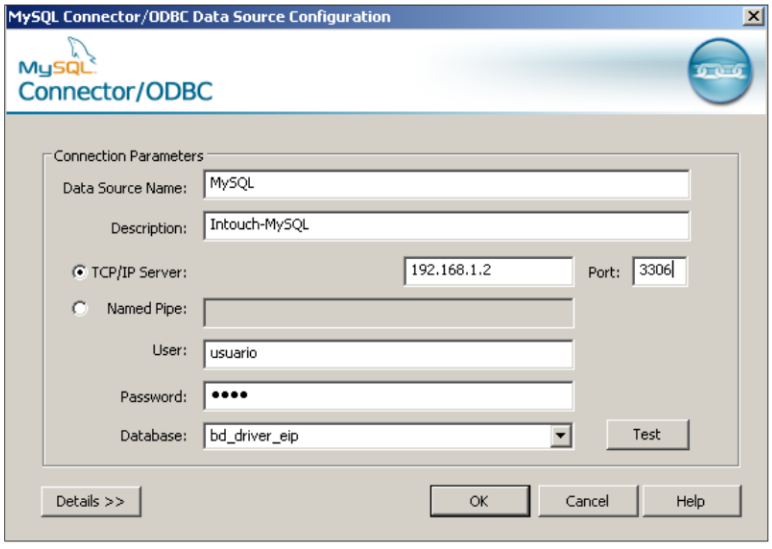


Figura 2.2. Conector ODBC para base de datos MySQL.

3. CONFIGURACIÓN Y FUNCIONAMIENTO DE SERVIDOR EIP

Al abrir el programa por primera vez aparecerá un mensaje de error de conexión a la base de datos por lo cual se debe acceder a la configuración de la base de datos de la Figura 3.1 dando clic en el botón “Base Datos”. En esta ventana se ingresa la dirección IP de la máquina en la que se encuentra la base de datos, el puerto, nombre de usuario y contraseña que fueron configurados al instalar MySQL y a su vez se utilizaron al crear la conexión, por último, se da clic en el botón aceptar para guardar los datos ingresados.

Por otro lado, el botón “Elegir esquema” abre la ventana de la Figura 3.2 que permite seleccionar el archivo de texto “BD_DRIVER.sql” que contiene el esquema con las tablas requeridas y la relación entre ellas. En MySQL existe únicamente el esquema sin tablas, por tanto, si no se carga el archivo con las instrucciones, el software no podrá almacenar la información de configuración ni activar las funciones de lectura ni escritura en el PLC.

Figura 3.1. Configuración de base de datos.

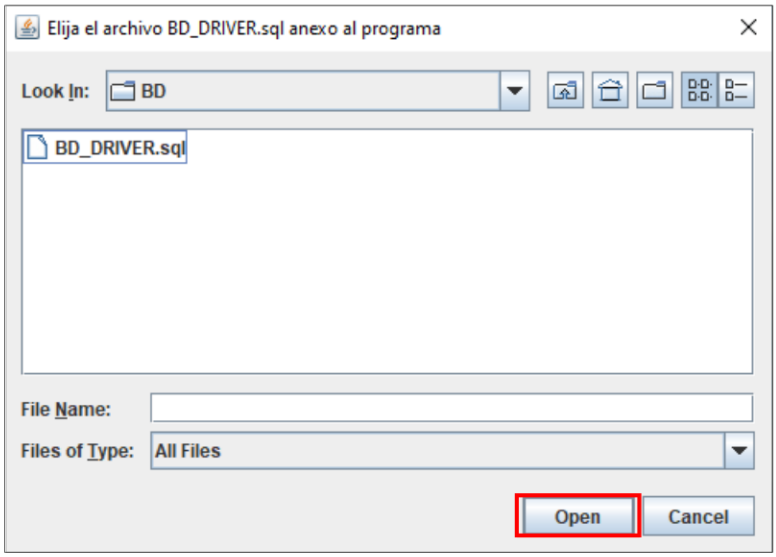


Figura 3.2. Selección de archivo de texto para creación de tablas.

Después de ingresar los datos de configuración o seleccionar el esquema se requiere reiniciar el software para que se efectúen los cambios realizados como se observa en el mensaje de alerta de la Figura 3.3.

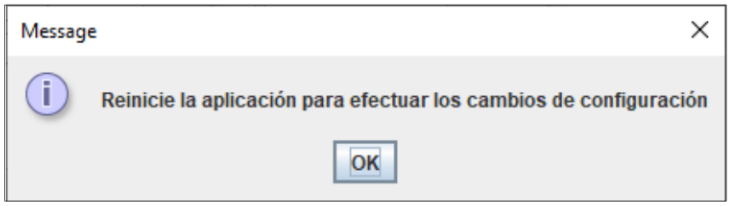


Figura 3.3. Mensaje de reinicio del software.

En caso de no existir el esquema y no haber sido seleccionado en la ventana de la Figura 3.2 se observa el mensaje de la Figura 3.4. En este caso se debe ingresar a la pantalla de configuración de la base de datos y cargar el archivo BD_DRIVER.sql.

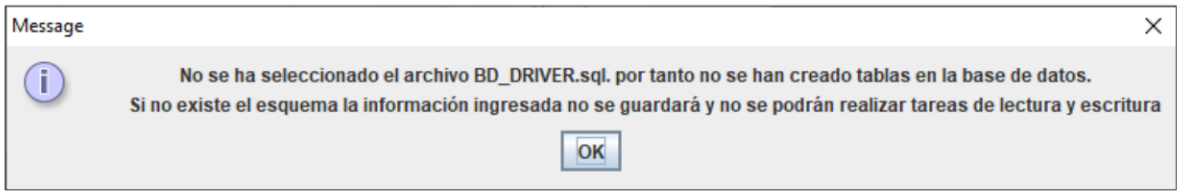


Figura 3.4. Mensaje de advertencia de tablas no creadas.

Una vez se ha configurado la base de datos se procede con el funcionamiento del servidor, el cual requiere inicialmente una configuración de los dispositivos que se administran y los tags que pertenecen a cada PLC.

Primero se administran los dispositivos que pertenecen a la red dando clic en el botón “PLCS” que se encuentra en la sección de ventanas, permitiendo el acceso a la pantalla de administración de PLCs.

Figura 3.5. Ventana de configuración.

En la ventana de administración de PLCs se pueden realizar tres funciones: agregar, modificar y eliminar. A continuación, se detalla cómo realizar cada una de ellas.

3.1. Agregar PLC

Para agregar un PLC se ingresan los datos que se observan en la Figura 3.6 y se presiona el botón “Añadir”:

- Nombre: es un campo único que indica el nombre del dispositivo lógico programable.
- Dirección: en este campo se especifica la dirección IP del PLC, la cual es única para cada dispositivo.
- Device Group: este campo es único y caracteriza al PLC o proceso que controla el dispositivo.
- Tiempo: el tiempo de muestreo indica cada cuanto tiempo se realiza la lectura de tags en el PLC.

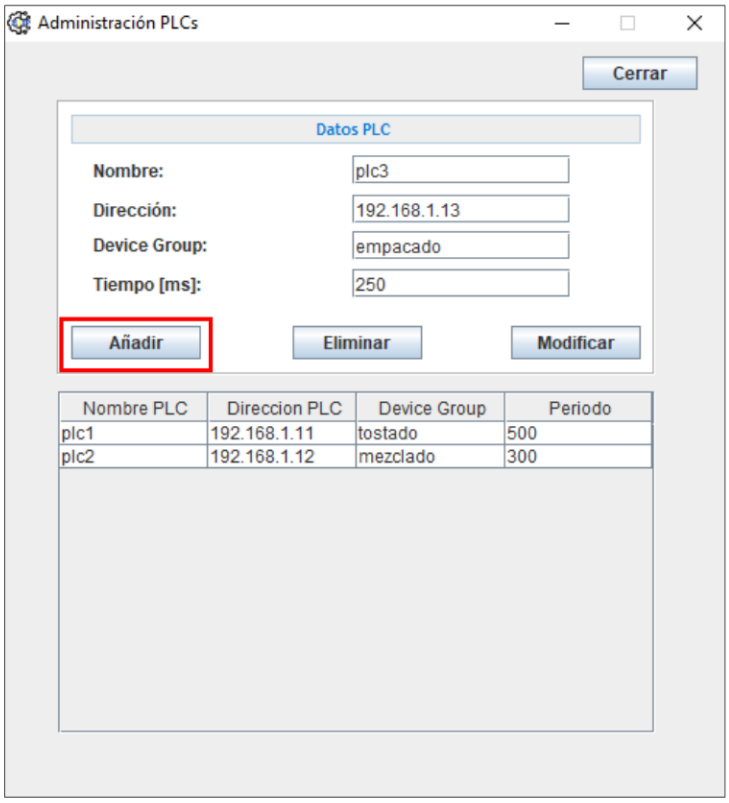


Figura 3.6. Creación de PLC.

Al presionar el botón “Añadir” si existiese un error con los datos ingresados, puede aparecer cualquiera de los mensajes que se muestran en la Figura 3.7, Figura 3.8, Figura 3.9 y Figura 3.10.

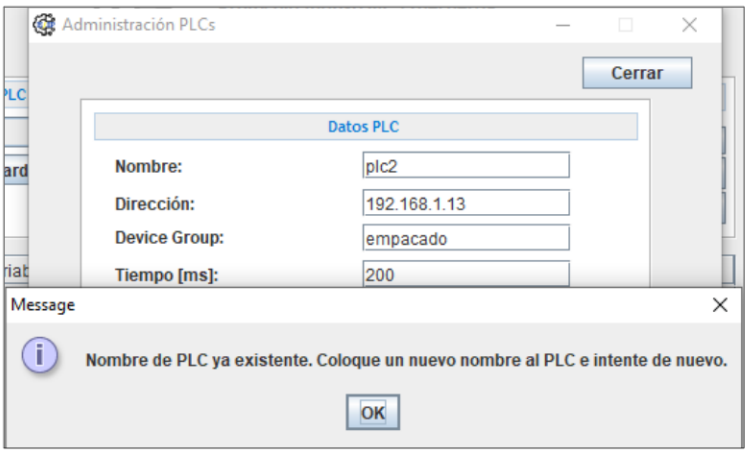


Figura 3.7. Error al ingresar nombre del PLC.

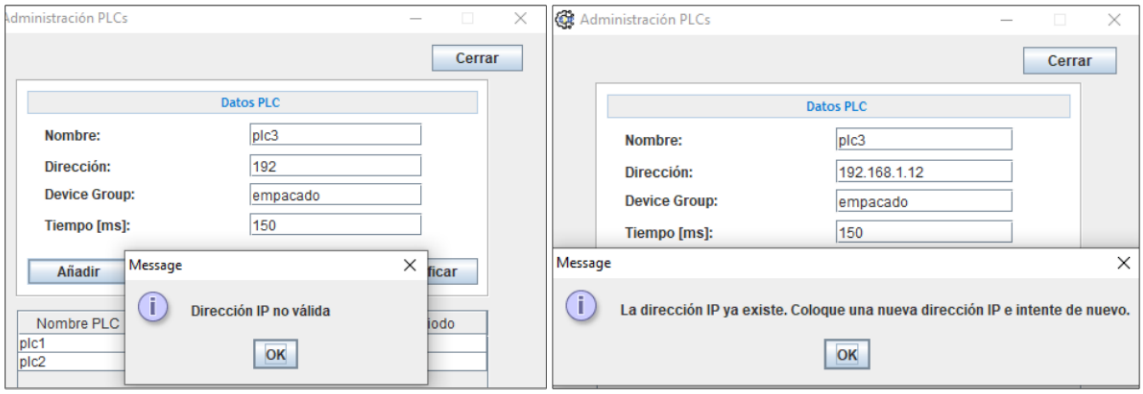


Figura 3.8. Errores al ingresar la dirección IP.

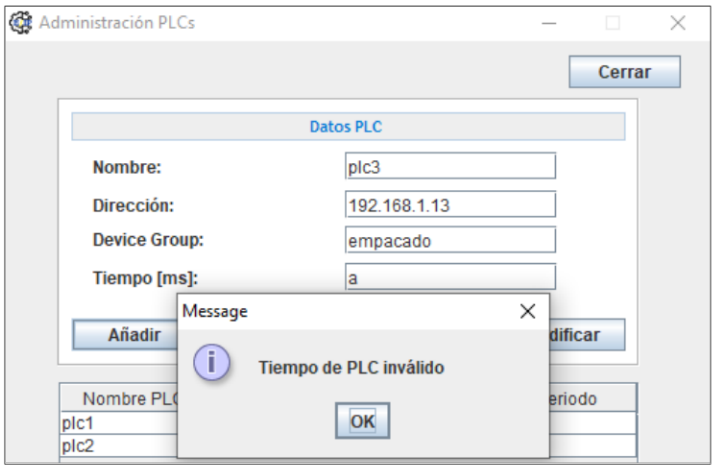


Figura 3.9. Error al ingresar tiempo de PLC.

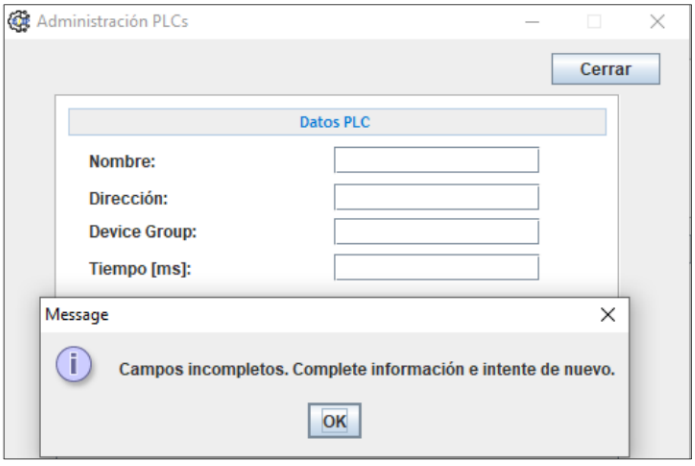


Figura 3.10. Error al registrar campos vacíos.

3.2. Eliminar PLC

Para eliminar un PLC se detallan a continuación los pasos a seguir.

1. Seleccionar PLC que se requiere descartar.

2. Presionar botón “Eliminar”. Figura 3.11.

3. Confirmar eliminación. Figura 3.12.

Con esta acción se borra toda la información relacionada con el PLC seleccionado.

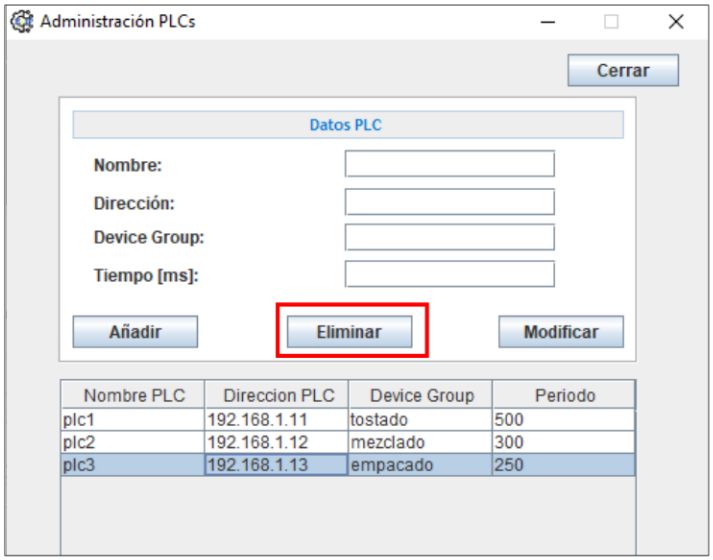


Figura 3.11. Eliminación de PLC.

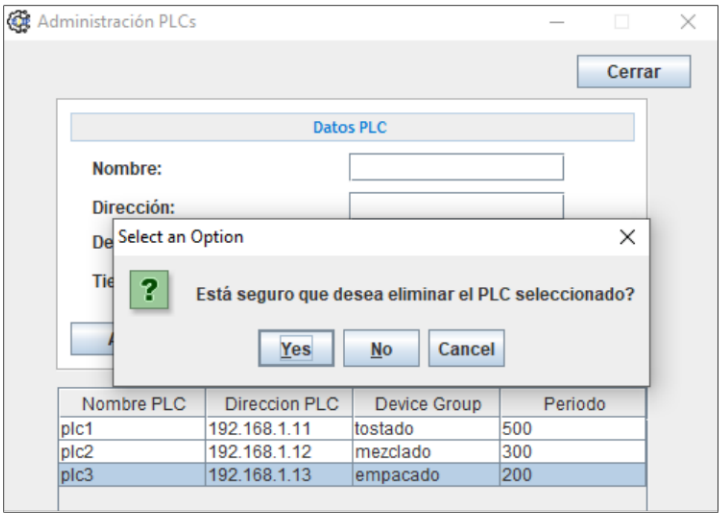


Figura 3.12. Confirmación de eliminación PLC.

3.3. Modificar PLC

Para modificar un parámetro del PLC se debe:

1. Modificar parámetro en la tabla.

2. Presionar enter para registrar el cambio.

3. Dar clic en el botón Modificar.

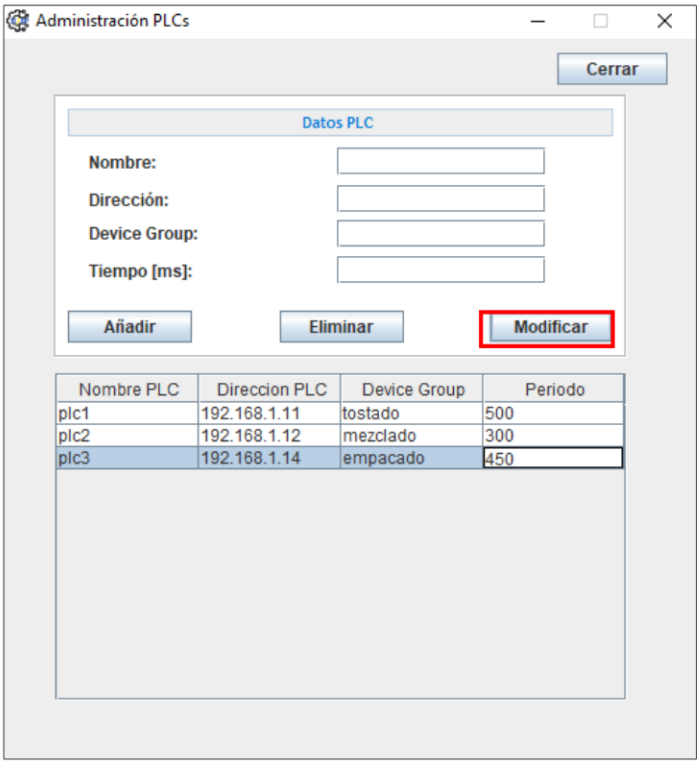


Figura 3.13. Modificación de parámetro de PLC.

Se validan los datos modificados cumpliendo las mismas condiciones que en la creación del PLC, como se observa en la Figura 3.14.

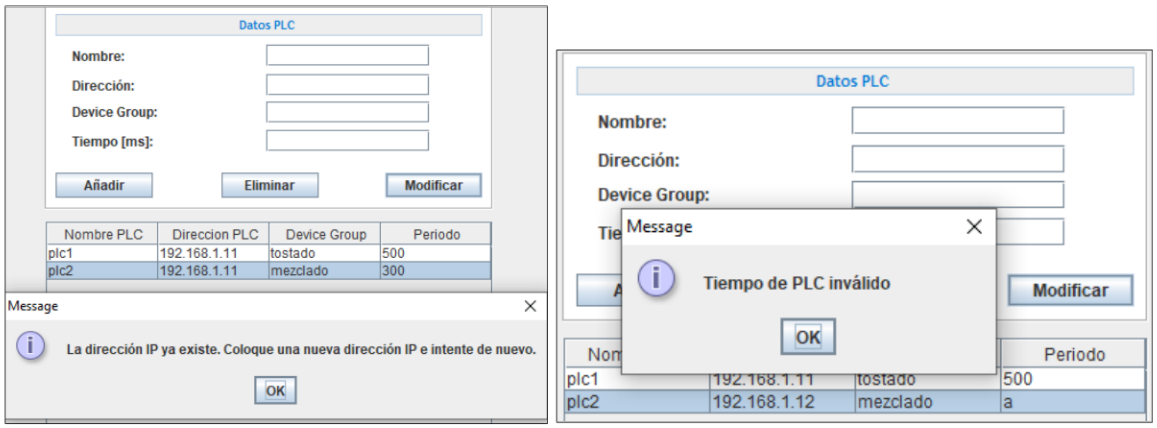


Figura 3.14. Errores al modificar parámetros del PLC.

3.4. Agregar tags a cada PLC

Para añadir variables a cada PLC se requiere:

1. Seleccionar el PLC al cual se requiere añadir tags en la sección PLC de la ventana de configuración.
2. Ingresar el nombre del tag único para cada variable, la dirección que en caso de ser booleano tiene el formato B3:0/0, entero N7:0 y flotante F8:0, el tipo de dato, la acción de lectura o escritura y si requiere o no registro histórico.
3. Dar clic en el botón añadir y se agrega el tag a la lista del PLC seleccionado como se observa en la Figura 3.15 y Figura 3.16.
4. Presionar el botón “Guardar” y esperar a que aparezca el mensaje de confirmación de guardado.

Figura 3.15. Ingreso de información para agregar tag.

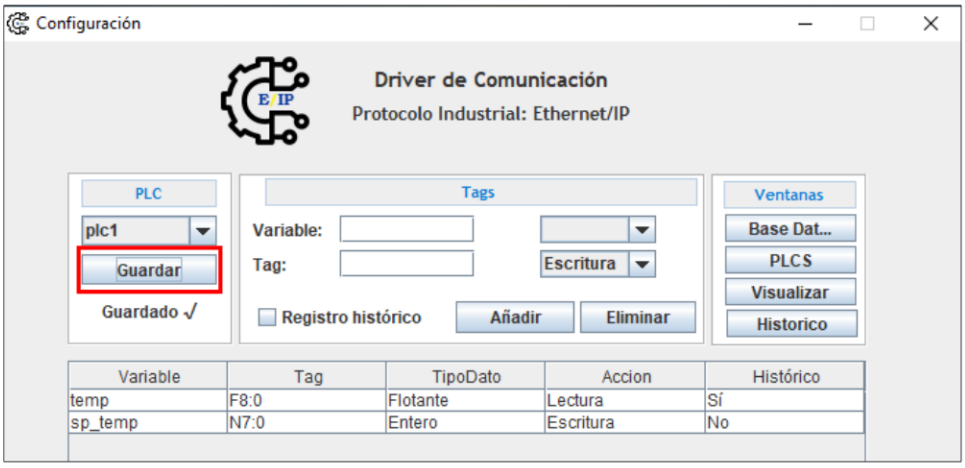


Figura 3.16. Tag agregado al PLC.

Si existiese algún error en el ingreso de un tag se visualizan los mensajes de la Figura 3.17, Figura 3.18 y Figura 3.19.

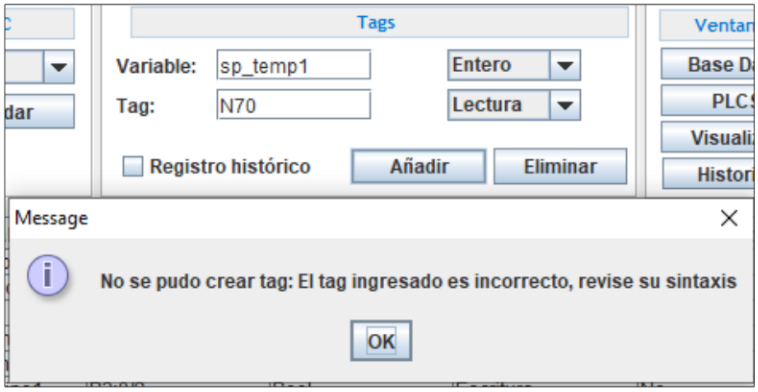


Figura 3.17. Error con la dirección del tag ingresado.

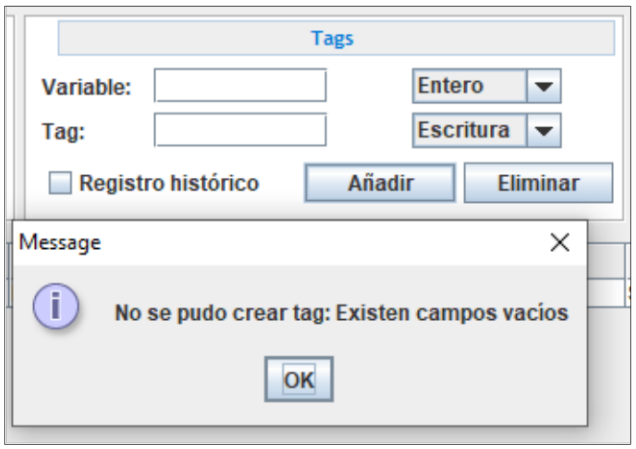


Figura 3.18. Error al existir campos vacíos del tag.

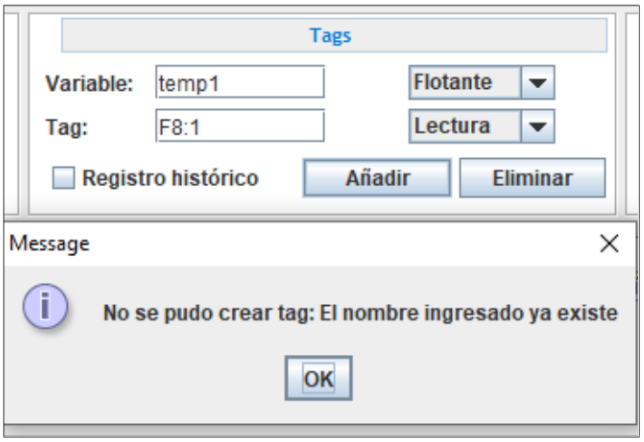


Figura 3.19. Error al ingresar nombre de una variable ya existente.

3.5. Eliminar tags

Para eliminar tags se requiere:

1. Seleccionar la variable que se requiere descartar.
2. Presionar el botón “Eliminar”
3. Confirmar eliminación del tag.
4. Presionar el botón “Guardar” y esperar el mensaje de confirmación que ha sido la información guardada correctamente.

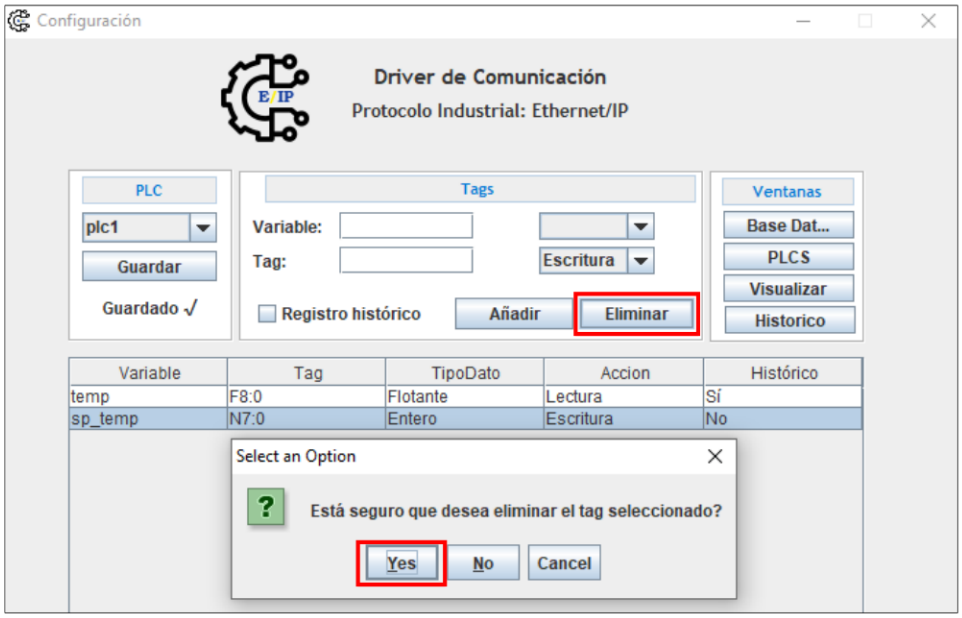


Figura 3.20. Eliminación de tag y confirmación.

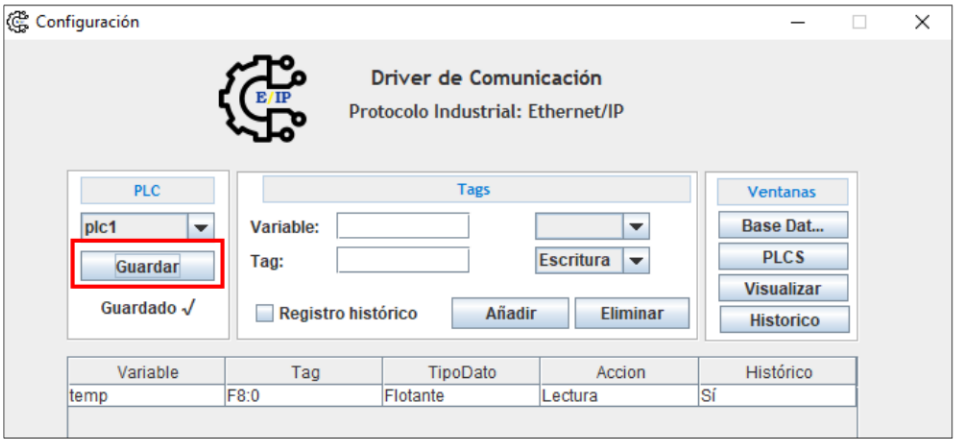


Figura 3.21. Variable eliminada de la lista de tags.

3.6. Activación de modo automático

1. En la ventana de configuración, sección ventanas (Figura 3.5) presionar el botón “Visualizar” para acceder a la pantalla de visualización.
2. Presionar el botón “Activar”.
3. Se muestra un mensaje de espera hasta que se establezca la conexión con el PLC, como se observa la Figura 3.22.

Nota: Si se muestra el símbolo  y la tabla tiene el cuadriculado en verde la conexión ha sido exitosa (Figura 3.23), caso contrario el símbolo  y el mensaje “No se pudo conectar con el PLC” indican que no se ha logrado la conexión con el dispositivo. (Figura 3.24)

Figura 3.22. Mensaje de espera de conexión al PLC.

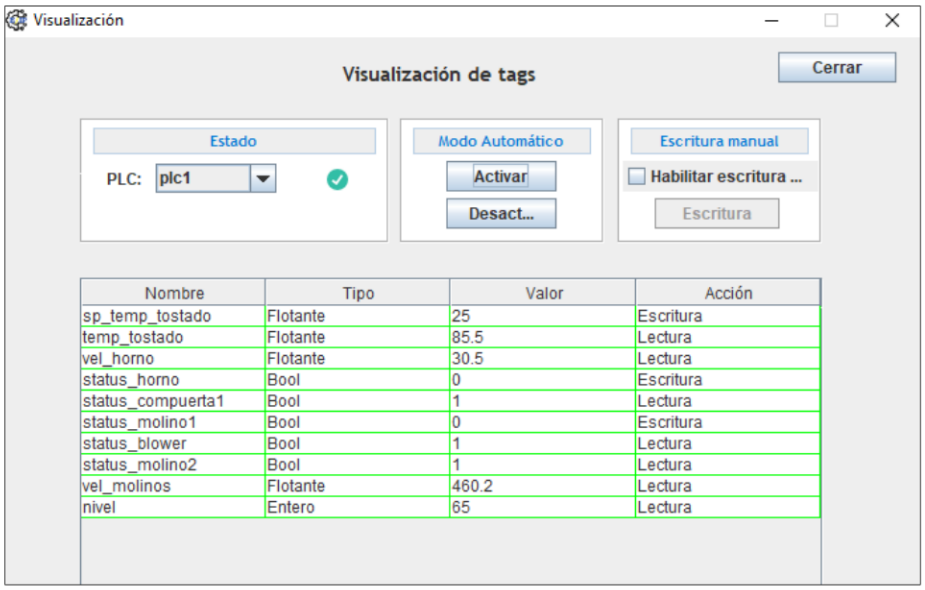


Figura 3.23. Funcionamiento automático de servidor de datos.

Figura 3.24. Error al conectar con el PLC.

3.7. Desactivación de modo automático

Para desactivar el modo automático se presiona el botón “Desactivar”, entonces se visualiza el símbolo  y el cuadriculado de la tabla cambia de verde a gris.

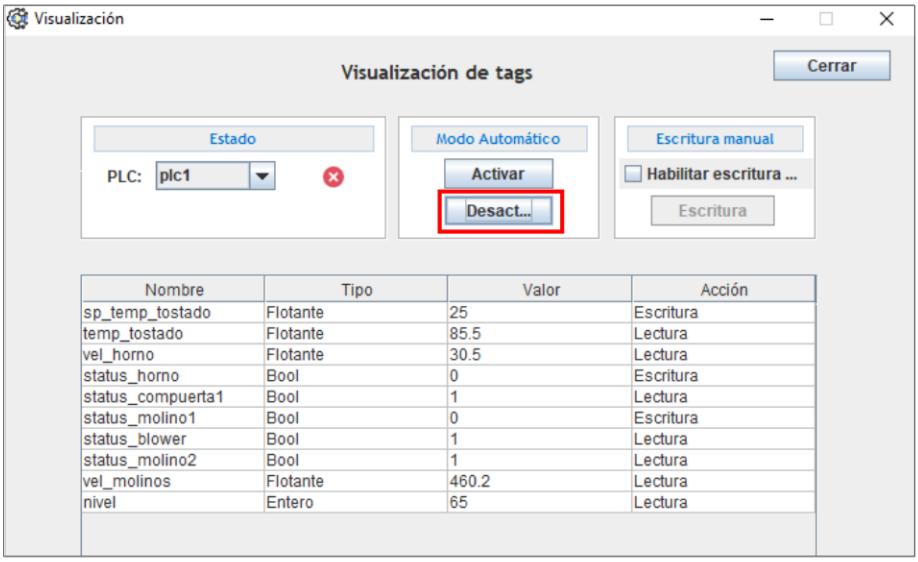


Figura 3.25. Modo automático desactivado.

3.8. Habilitar modo manual

Para habilitar el modo manual se requiere:

1. Habilitar la escritura manual a través de la casilla de verificación en la sección escritura manual.

2. Se muestra un mensaje de advertencia de cambio de modo de funcionamiento y en caso de estar funcionando el modo automático se desactiva inmediatamente. Dar clic en OK.

3. Presionar el botón “Escritura”, el cual habilita una lectura automática destinada para el modo manual y permite visualizar la ventana de escritura desde la aplicación

Figura 3.26. Activación de modo de escritura manual.

4. Seleccionar el PLC en el cual se realizarán las modificaciones.

5. Ingresar los valores de escritura en la tabla y dar enter.

6. Presionar el botón “OK”.

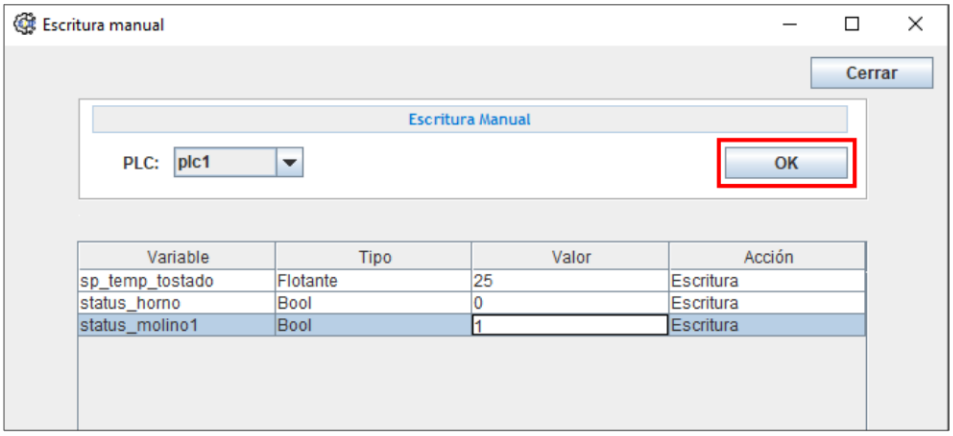


Figura 3.27. Escritura de valores desde ventana de escritura.

Nota: En caso de existir algún problema de conexión se visualiza un mensaje de error en la ventana de Visualización.

Al presionar el botón OK se podría obtener el mensaje de error de ingreso de datos de la Figura 3.28 o el mensaje de error en la conexión de la Figura 3.29.

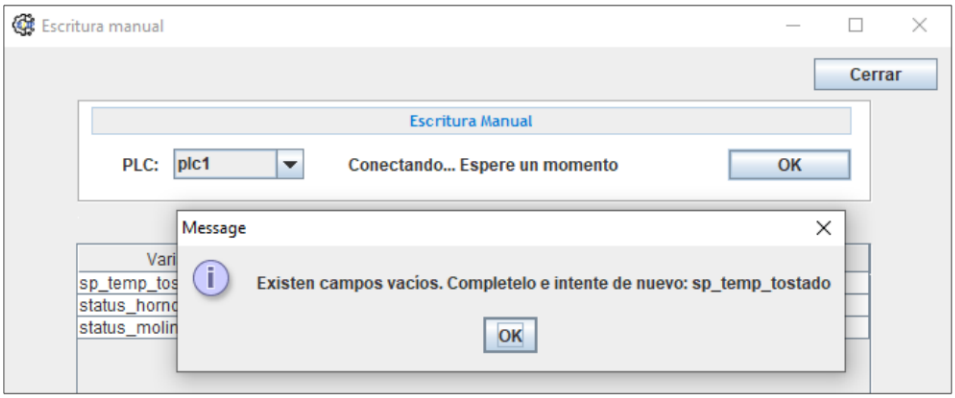


Figura 3.28. Error en el ingreso de valores destinados a escritura.

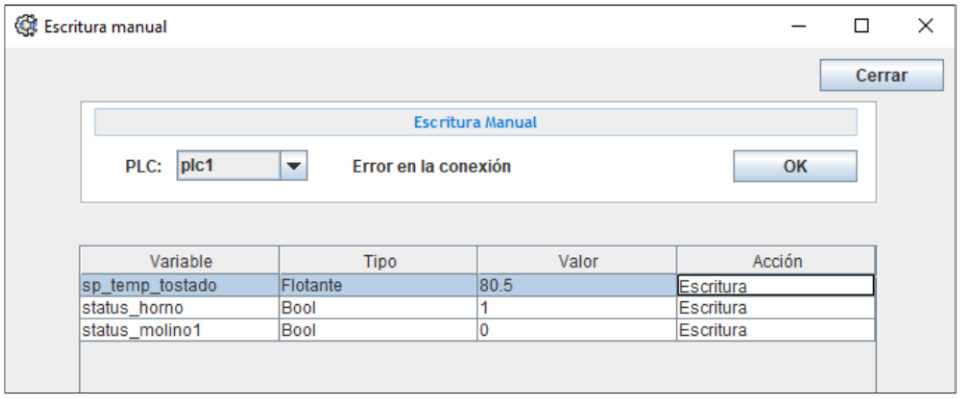


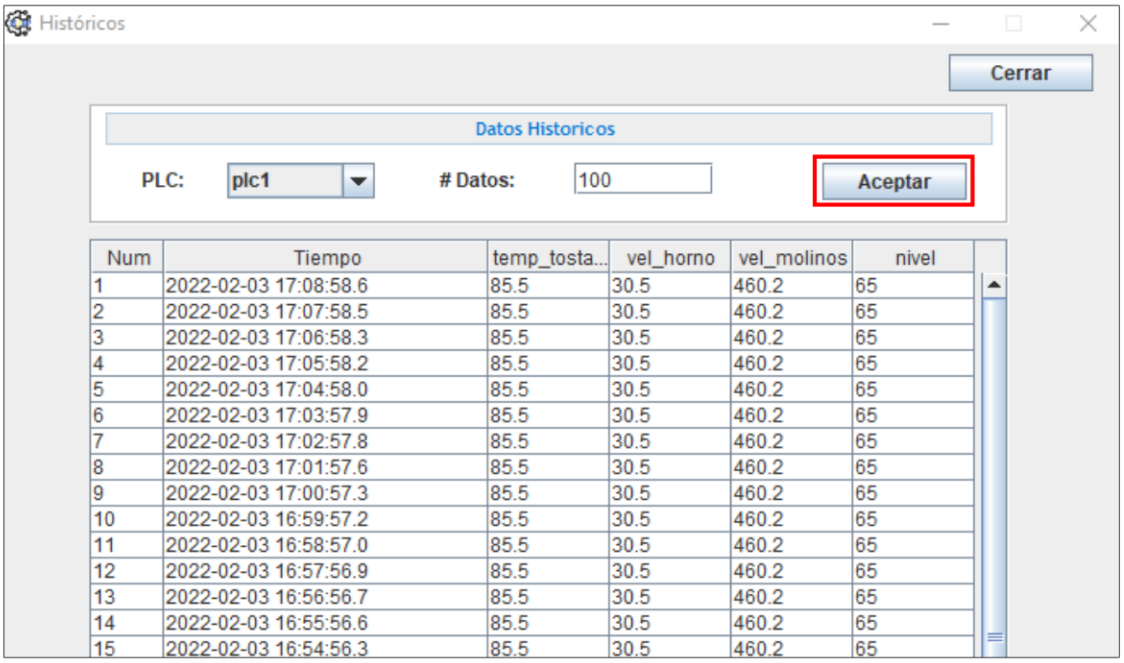
Figura 3.29. Error en la conexión y escritura.

3.9. Registro histórico

Desde la sección ventanas en la pantalla de configuración de la Figura 3.5 se accede al registro histórico, en esta ventana se puede visualizar el valor de las variables cada minuto durante las últimas 24 horas. Cabe mencionar que el registro histórico inicia cuando se activa el servidor en modo automático y se detiene cuando se desactiva. Para visualizar se requiere:

Seleccionar el PLC del cual se necesita el registro histórico.

1. Ingresar el número de datos que se requieren.
2. Dar clic en “Aceptar”.



Num	Tiempo	temp_tosta...	vel_horno	vel_molinos	nivel
1	2022-02-03 17:08:58.6	85.5	30.5	460.2	65
2	2022-02-03 17:07:58.5	85.5	30.5	460.2	65
3	2022-02-03 17:06:58.3	85.5	30.5	460.2	65
4	2022-02-03 17:05:58.2	85.5	30.5	460.2	65
5	2022-02-03 17:04:58.0	85.5	30.5	460.2	65
6	2022-02-03 17:03:57.9	85.5	30.5	460.2	65
7	2022-02-03 17:02:57.8	85.5	30.5	460.2	65
8	2022-02-03 17:01:57.6	85.5	30.5	460.2	65
9	2022-02-03 17:00:57.3	85.5	30.5	460.2	65
10	2022-02-03 16:59:57.2	85.5	30.5	460.2	65
11	2022-02-03 16:58:57.0	85.5	30.5	460.2	65
12	2022-02-03 16:57:56.9	85.5	30.5	460.2	65
13	2022-02-03 16:56:56.7	85.5	30.5	460.2	65
14	2022-02-03 16:55:56.6	85.5	30.5	460.2	65
15	2022-02-03 16:54:56.3	85.5	30.5	460.2	65

Figura 3.30. Registro histórico.

4. CONEXIÓN CON INTOUCH

Para la conexión con Intouch inicialmente se debe abrir la conexión cuando se ejecuta la interfaz mediante scripts de aplicación, como se observa en la Figura 0.1. La sentencia de conexión requiere un tag tipo entero que guarde el identificador de conexión, en este caso el “connectionId” y el Data Source Name configurado en el conector ODBC.

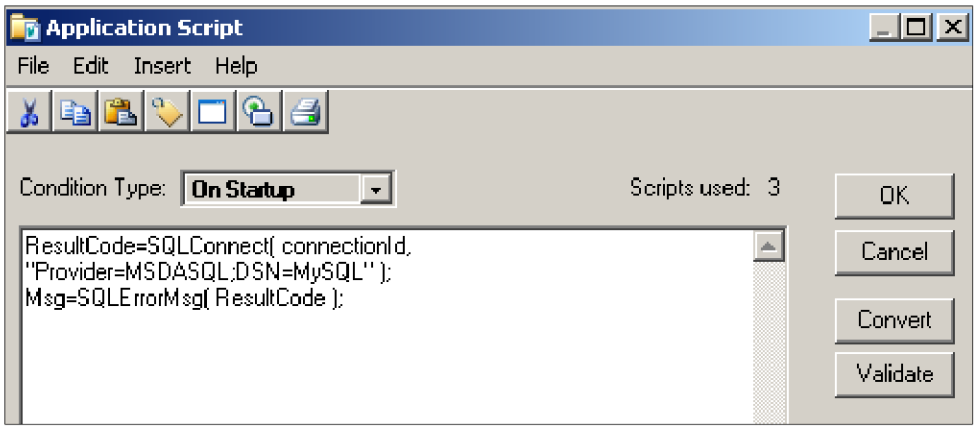


Figura 0.1. Apertura de conexión con la base de datos.

Para establecer la lectura y escritura de información primero se requiere crear un Bind List para cada variable, esta herramienta permite asignar a un tag del HMI el valor de una

columna de la tabla a la cual se requiere acceder. En la Figura 0.2 se observa el Bind List creado con su correspondiente ítem. La columna que contiene la información del tag es “valor”, por tanto, a todos los tags del HMI en cada Bind List se lo debe enlazar con la columna de este nombre.

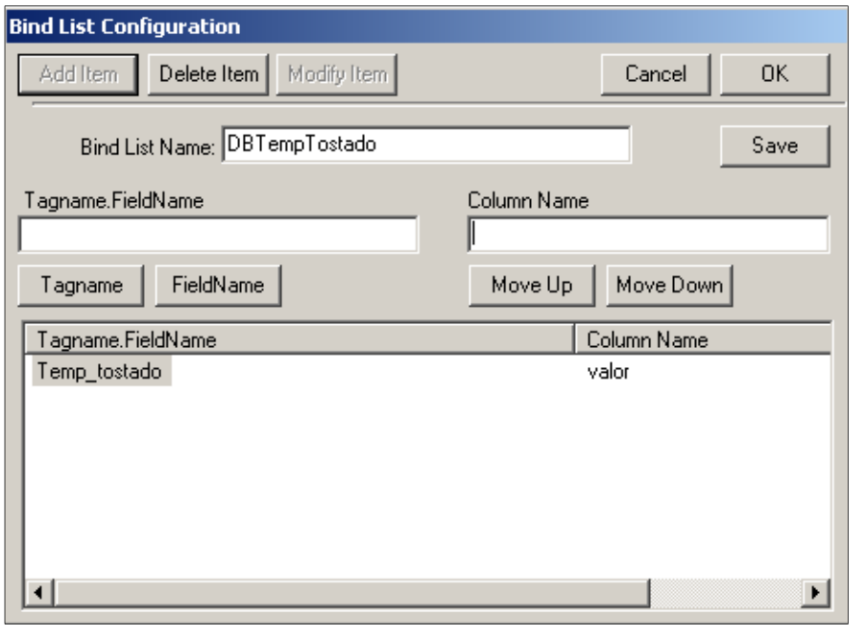


Figura 0.2. Bind List de variable Temp_tostado.

Una vez enlazada cada una de las variables con la columna valor se realiza un script con sentencias SQL para leer y escribir en la base de datos considerando la siguiente estructura:

- Lectura: se utiliza la sentencia SQLSelect que requiere del identificador de conexión, el nombre de la tabla que dependiendo del tipo de dato puede ser boolean, entero y flotante, el nombre del Bind List creado y el nombre de la variable que consta en el driver. La sentencias SQLLast selecciona el último registro de la tabla de resultados después de la ejecución de una sentencia SQLSelect y SQLEnd libera memoria de almacenamiento en la tabla de resultados.

SQLSelect(connectionId, nombreTabla, BindList, "nombreTag=nombreVariable", "");

SQLSelect(connectionId, "flotante", "DBTempTostado", "nombreTag='temp_tostado' ", "");

SQLLast(connectionId);

SQLEnd(connectionId);

- Escritura: se utiliza la sentencia SQLUpdate que requiere del identificador de conexión, el nombre de la tabla, Bind List y el nombre de la variable.

SQLUpdate(connectionId, nombreTabla,BindList, "nombreTag=nombreVariable");

SQLUpdate(connectionId, "flotante","DBSpTostado", "nombreTag='sp_temp_tostado' ");

En la Figura 0.3 se observa un ejemplo de script para escritura y lectura de variables.

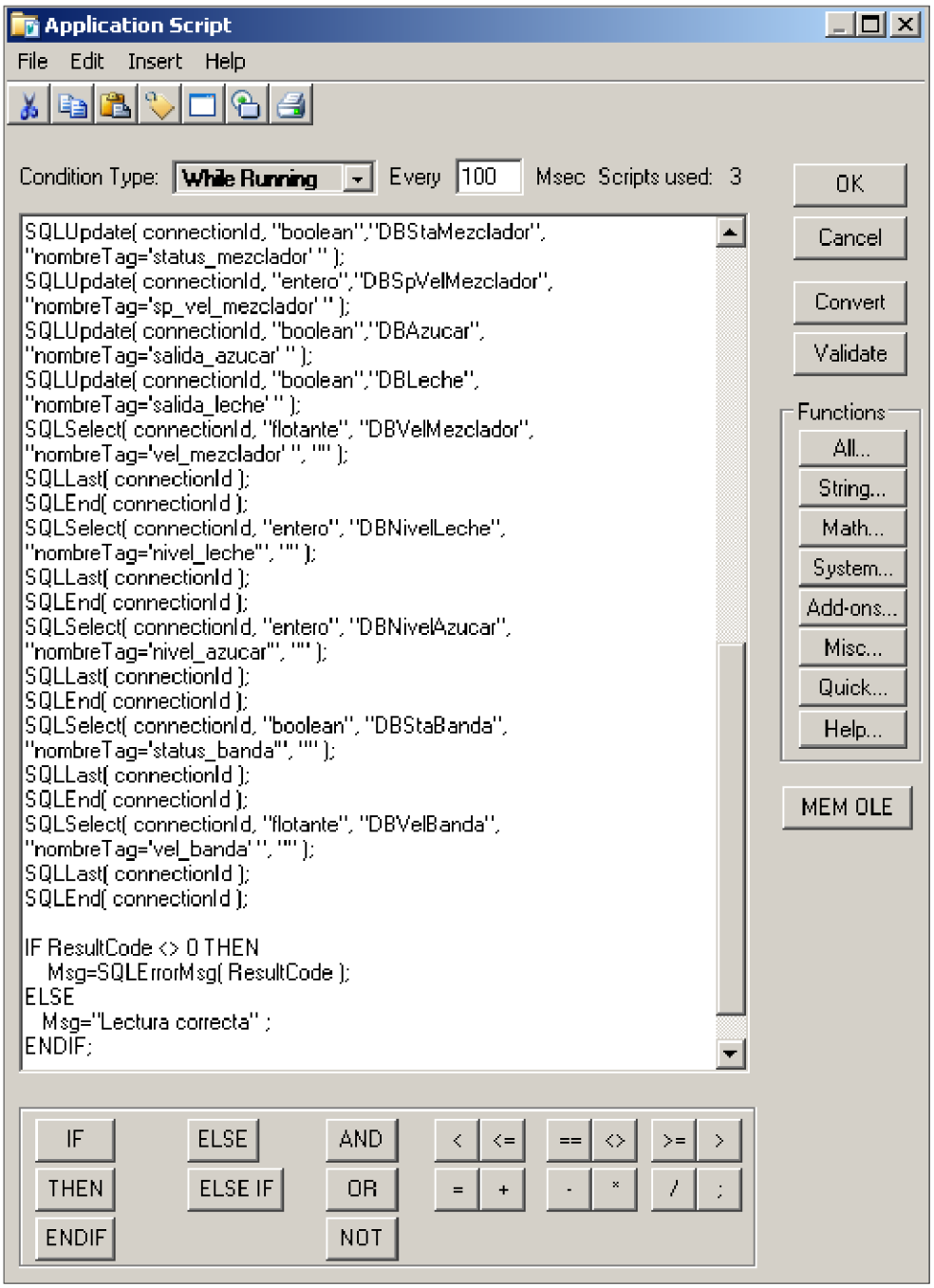


Figura 0.3. Lectura y escritura de la base de datos.

Al cerrar la interfaz se debe cerrar también la conexión con la base de datos mediante la sentencia SQLDisconnect que se realiza en un script de aplicación al cierre como se observa en la Figura 0.4.

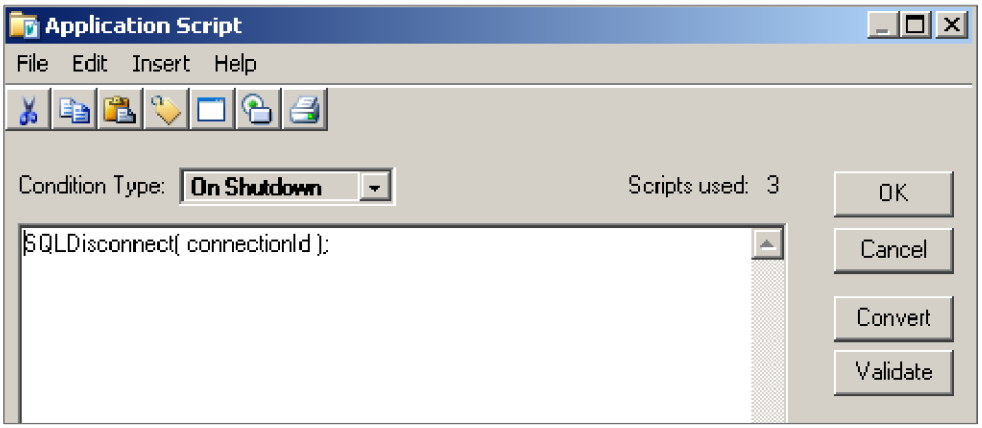


Figura 0.4. Cierre de conexión de base de datos.

ORDEN DE EMPASTADO