



ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

**“DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL
Y MONITOREO DE PARÁMETROS ELÉCTRICOS DENTRO DE
PROCESOS INDUSTRIALES MEDIANTE SOFTWARE LIBRE Y
COMUNICACIÓN MODBUS”**

FREDY DANIEL ROMERO HERRERA

Trabajo de Titulación modalidad: Proyectos de Investigación y Desarrollo, presentado ante el Instituto de Postgrado y Educación Continua de la ESPOCH, como requisito parcial para la obtención del grado de:

MAGISTER EN SISTEMAS DE CONTROL Y AUTOMATIZACIÓN INDUSTRIAL

RIOBAMBA - ECUADOR

SEPTIEMBRE 2022

©2022, Fredy Daniel Romero Herrera.

Se autoriza la reproducción total o parcial, con fines académicos, por cualquier medio o procedimiento, incluyendo la cita bibliográfica del documento, siempre y cuando se reconozca el Derecho de Autor.



ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

CERTIFICACIÓN:

EL TRIBUNAL DEL TRABAJO DE TITULACIÓN CERTIFICA QUE:

El **Trabajo de Titulación Modalidad Proyectos de Investigación y Desarrollo**, denominado: “DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL Y MONITOREO DE PARÁMETROS ELÉCTRICOS DENTRO DE PROCESOS INDUSTRIALES MEDIANTE SOFTWARE LIBRE Y COMUNICACIÓN MODBUS”, de responsabilidad del señor Fredy Daniel Romero Herrera, ha sido minuciosamente revisado y se autoriza su presentación.

ING. LUIS EDUARDO HIDALGO ALMEIDA; Ph. D.
PRESIDENTE

ING. HENRY ERNESTO VALLEJO VIZHUETE Mag.
DIRECTOR

ING. DIEGO FERNANDO VELOZ CHÉRREZ Mag.
MIEMBRO

ING. OSWALDO GEOVANNY MARTÍNEZ
GUASHIMA. M.Sc.
MIEMBRO

Riobamba, septiembre 2022

DERECHOS INTELECTUALES

Yo, Fredy Daniel Romero Herrera soy responsable de la ideas, doctrinas y resultados expuestos en este Trabajo de Titulación y el patrimonio intelectual del mismo pertenece a la Escuela Superior Politécnica de Chimborazo.



FREDY DANIEL ROMERO HERRERA

No. Cédula 0603333642

DECLARACIÓN DE AUTENTICIDAD

Yo, Fredy Daniel Romero Herrera, declaro que el presente proyecto de investigación es de mi autoría y que los resultados de este son auténticos y originales. Los textos constantes en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor, asumo la responsabilidad legal y académica de los contenidos de este Trabajo de Titulación de Maestría.



FREDY DANIEL ROMERO HERRERA

No. Cédula 0603333642

DEDICATORIA

Este trabajo de investigación está dedicado a mis hijos Diego Fernando y Daniel Alejandro.

AGRADECIMIENTO

A Dios y a mi madre Dolorosa por permitirme disfrutar de este momento, a mi esposa Verónica por apoyarme en todo momento, a mis padres Juan y Zoila por brindarme su apoyo incondicional.

TABLA DE CONTENIDO

RESUMEN.....	XIII
SUMMARY	XIV
CAPÍTULO I.....	1
1. INTRODUCCIÓN.....	1
1.1 ANTECEDENTES	1
1.2 FORMULACIÓN DEL PROBLEMA	2
1.3 PREGUNTAS DIRECTRICES	2
1.4 JUSTIFICACIÓN	3
1.5 OBJETIVOS DE LA INVESTIGACIÓN.....	4
1.5.1 <i>Objetivo general</i>	4
1.5.2 <i>Objetivos específicos</i>	4
1.6 HIPÓTESIS	4
CAPÍTULO II	6
2 MARCO TEÓRICO	6
2.1. SISTEMAS DE CONTROL	6
2.1.1. <i>Sistema de control en lazo abierto</i>	7
2.1.2. <i>Sistema de control en lazo cerrado</i>	7
2.2 TIPOS DE CONTROLADORES	8
2.2.1. <i>Controlador ON-OFF</i>	8
2.2.2. <i>Controlador proporcional (P)</i>	9
2.2.3. <i>Controlador integral (I)</i>	9
2.2.4. <i>Controlador proporcional integral (PI)</i>	9
2.2.5. <i>Controlador proporcional integral derivativo (PID)</i>	9
2.3 CONTROLADORE LÓGICOS PROGRAMABLES	10
2.3.1. <i>Definición</i>	10
2.1.1. <i>Funciones</i>	11
2.1.2. <i>Estructura interna</i>	12
2.1.3. <i>Estructura externa</i>	13
2.1.4. <i>Sistema de entrada y salida</i>	14
2.1.4.1. <i>Señales digitales</i>	15

2.1.4.2.	<i>Señales analógicas</i>	15
2.2.	PROTOCOLOS DE COMUNICACIÓN INDUSTRIAL.....	16
2.4.1	<i>El protocolo Hart</i>	17
2.4.2	<i>El protocolo DeviceNet</i>	17
2.4.3	<i>El protocolo Profibus</i>	18
2.5	INTERFAZ HOMBRE MÁQUINA.....	19
CAPÍTULO III.....		20
3.	DESARROLLO DE LA INVESTIGACIÓN.....	20
3.1.	METODOLOGÍA.....	20
3.2.	DISEÑO GENERAL DEL SISTEMA.....	20
3.3.	SELECCIÓN DEL TABLERO DE CONTROL.....	21
3.3.1.	<i>Canaleta Ranurada y Riel Din</i>	22
3.3.2.	<i>Controlador Simatics S7-1200</i>	23
3.3.3.	<i>Sinamics V20</i>	23
3.3.4.	<i>Logo! 8</i>	24
3.3.5.	<i>Módulo de comunicación CM1241</i>	25
3.3.6.	<i>Medidor multifunción Siemens Sentron PAC3100</i>	25
3.3.7.	<i>Bobina de corriente TC</i>	26
3.3.8.	<i>Motor</i>	27
3.4.	RED DE COMUNICACIÓN.....	27
3.4.1.	<i>Comunicación RS 232/485</i>	27
3.4.2.	<i>Comunicación Modbus TCP- IP</i>	27
3.5.	PROGRAMACIÓN EN EL PLC.....	28
3.5.1.	<i>Variables empleadas para la conexión con el HMI</i>	29
3.5.2.	<i>MB_COMM_LOAD</i>	30
3.5.3.	<i>MB_MASTER</i>	31
3.5.4.	<i>Contador ascendente</i>	32
3.5.5.	<i>MOVE</i>	32
3.5.6.	<i>Normalizar</i>	33
CAPÍTULO IV.....		35
4.	ANÁLISIS Y RESULTADOS.....	35
4.1.	HIPÓTESIS.....	35
4.2.	OPERACIONALIZACIÓN DE LAS VARIABLES.....	35
4.2.1.	<i>Operacionalización conceptual</i>	35
4.2.1.1.	<i>VARIABLE INDEPENDIENTE</i>	35

4.2.1.2.	<i>VARIABLE DEPENDIENTE.....</i>	36
4.2.2.	<i>Operacionalización metodológica</i>	36
4.3.	<i>EVALUACIÓN DE VARIABLES.....</i>	37
4.3.1.	<i>Velocidad (rpm).....</i>	37
4.3.1.1.	<i>Prueba de normalidad para la variable velocidad (rpm) con el tacómetro</i>	40
4.3.1.2.	<i>Prueba de normalidad para la variable velocidad (rpm) con la Interfaz.....</i>	41
4.3.1.3.	<i>Prueba de diferencia de medias.....</i>	42
4.3.2.	<i>Corriente (Amperios)</i>	43
4.3.2.1.	<i>Prueba de normalidad para la variable corriente (amperios) con la pinza amperimétrica</i>	46
4.3.2.2.	<i>Prueba de normalidad para la variable corriente (amperios) con la Interfaz .</i>	47
4.3.2.3.	<i>Prueba de Wilcoxon (Prueba de medianas).....</i>	48
4.4.	<i>COMPARACIÓN DE COSTOS</i>	49
4.4.1.	<i>Prueba de normalidad para los costos Scada Siemens.....</i>	50
4.4.2.	<i>Prueba de normalidad para los costos HMI de la investigación.....</i>	51
4.4.3.	<i>Prueba de diferencia de medias</i>	52
4.5.	<i>ANÁLISIS DE DATOS ALMACENADOS</i>	54
	CAPÍTULO V.....	56
5.	<i>ANÁLISIS Y RESULTADOS</i>	56
5.1.	<i>DESARROLLO DE LA INTERFAZ GRÁFICA</i>	56
5.1.1.	<i>Descripción del SNAP 7.....</i>	56
5.1.2.	<i>Botón START.....</i>	57
5.1.3.	<i>Botón Stop</i>	58
5.1.4.	<i>Botones de giro del motor</i>	59
5.1.5.	<i>Velocidad.....</i>	60
5.1.6.	<i>Botón de conexión</i>	60
5.1.7.	<i>Indicador de temperatura.....</i>	61
5.1.8.	<i>Indicador de Velocidad.....</i>	62
5.1.9.	<i>Indicador de corriente.....</i>	63
5.1.10.	<i>Indicador de la frecuencia</i>	64
5.2.	<i>ALMACENAMIENTO DE DATOS</i>	65
	CONCLUSIONES	
	RECOMENDACIONES	
	BIBLIOGRAFÍA	
	ANEXOS	

ÍNDICE DE TABLAS

Tabla 4-1 Operacionalización metodológica de variables Variable Independiente	36
Tabla 4-2 Operacionalización metodológica de variables Variable Dependiente.....	37
Tabla 4-3 Características Tacómetro	38
Tabla 4-4 Mediciones de velocidad	39
Tabla 4-5 Características Pinza amperimétrica.....	44
Tabla 4-6 Mediciones de corriente.....	45
Tabla 4-7 Precios referenciales SCADA.....	50
Tabla 4-8 Precios referenciales HMI DE LA INVESTIGACIÓN.....	50
Tabla 4-9 Almacenamiento de datos.....	54

ÍNDICE DE FIGURAS

Figura 2-1. Componentes básicos de un sistema de control.	6
Figura 2-2 Elementos de un sistema de control a lazo abierto	7
Figura 2-3 Elementos de un sistema de control a lazo cerrado	7
Figura 2-4 Comportamiento del controlador ON-OFF con respecto al tiempo.....	8
Figura 2-5 Estructura interna de un controlador lógico programable.....	13
Figura 2-6 Señal digital a lo largo del tiempo	15
Figura 2-7 Señal analógica a lo largo del tiempo.	16
Figura 2-8 Trama de protocolo de comunicación	17
Figura 2-9 Trama de protocolo de comunicación DeviceNet.....	18
Figura 2-10 Trama de protocolo de comunicación Profibus	18
Figura 2-11 Trama de protocolo de comunicación Modbus TCP IP.....	19
Figura 2-12 Pirámide de automatización	19
Figura 3-1 Diagrama de Bloques	21
Figura 3-2 Tablero eléctrico	22
Figura 3-3 Canaleta Ranurada y Riel Din.....	22
Figura 3-4 Controlador Simatics S7-1200	23
Figura 3-5 Sinamics V20	24
Figura 3-6 Logo ¡8.....	24
Figura 3-7 Módulo de comunicación CM1241.....	25
Figura 3-8 Medidor multifuncional Siemens Sentron PAC3100.....	26

Figura 3-9 Bobina de corriente TC	26
Figura 3-10 Motor.....	27
Figura 3-11 Conector DB9	28
Figura 3-12 Módulo PLC.....	29
Figura 3-13 Declaración de variables	29
Figura 3-14 Tipos de variables	30
Figura 3-15 Variable temperatura.....	30
Figura 3-16 Bloque MB_COMM_LOAD	31
Figura 3-17 Bloque MB_MASTER.....	32
Figura 3-18 CTU – Contador ascendente	32
Figura 3-19 Move	33
Figura 3-20 Normalizar	34
Figura 4-1 Toma de muestra de velocidad.....	38
Figura 4-2 % VELOCIDAS VS VELOCIDAD MOTOR.....	40
Figura 4-3 Toma de muestra de corriente	44
Figura 4-4 % VELOCIDAD VS CORRIENTE	46
Figura 4-5 Gráfica de torque en el tiempo	55
Figura 4-6 Gráfica de corriente en el tiempo	55
Figura 5-1 Snap7.....	57
Figura 5-2 Botón de inicio	58
Figura 5-3 Botón de pare	58

Figura 5-4 Botones de giro del motor	59
Figura 5-5 Velocidad del motor	60
Figura 5-6 Botón de conexión	61
Figura 5-7 Indicador de temperatura	62
Figura 5-8 Indicador de velocidad	63
Figura 5-9 Indicador de corriente	64
Figura 5-10 Indicador de frecuencia	65
Figura 5-11 Diagrama de almacenamiento de datos.....	66
Figura 5-12 Excel de recolección de datos	66

RESUMEN

En este trabajo se diseñó un sistema de control y monitoreo de parámetros eléctricos mediante el protocolo de comunicación Modbus, utilizando el lenguaje de programación Python. Para ello se identificó parámetros eléctricos indispensables en los procesos productivos que nos permitan obtener información necesaria para optimizar tiempos y recursos. La programación en el PLC S7-1200 se la realizó en el TIA PORTAL, mediante la comunicación Modbus se acoplo el variador de frecuencia SIMATICS V20 y el medidor de parámetros eléctricos SENTRON PAC 3100, que permiten obtener los datos que posteriormente se reflejaran en la HMI. Mediante el lenguaje de programación Python utilizando la librería Snap7 se logró desarrollar el código de la interfaz gráfica, esta muestra los datos en tiempo real de los parámetros eléctricos requeridos en el proceso. Mediante comparación de datos se calculó el error relativo permisible, llegando a obtener como resultado que los valores de las variables medidas mediante equipos físicos y mostradas en el HMI, no superan el 1%, es decir son los datos de alta confiabilidad. Además, se comprobó que el diseño abarataría la implementación de un sistema que requiera una licencia de uso, el cual se adapta a los requerimientos de las pequeñas y medianas industrias.

Palabras Clave: <PYTHON>, <HMI>, <SIMATICS V20>, < SENTRON PAC 3100>, <MODBUS>, <SNAP7>, <LENGUAJE DE PROGRAMACIÓN>, <PARÁMETROS ELÉCTRICOS>, <INTERFAZ GRÁFICA>



Firmado electrónicamente por:
**LUIS ALBERTO
CAMINOS
VARGAS**



0081-DBRA-UPT-IPEC-2022

SUMMARY

In this research, a control and monitoring system of electrical parameters was designed through the Modbus communication protocol, using the Python programming language. For this purpose, indispensable electrical parameters were identified in the production processes that allow us to obtain the necessary information to optimize time and resources. The programming in the PLC S7-1200 was done in the TIA PORTAL, through the Modbus communication, the frequency inverter SIMATICS V20 and the electrical parameters meter SENTRON PAC 3100 were coupled, which allow to obtain the data that later will be reflected in the HMI. By means of the Python programming language using the Snap7 library, the code of the graphic interface was developed, which shows the data in real time of the electrical parameters required in the process. By means of data comparison, the permissible relative error was calculated, obtaining as a result that the values of the variables measured through physical equipment and shown in the HMI do not exceed 1%, that is to say, the data are highly reliable. In addition, it was found that the design would make it cheaper to implement a system that requires a license for use, which is adapted to the requirements of small and medium-sized industries.

Keywords: <PYTHON>, <HMI>, <SIMATICS V20>, <SENTRON PAC 3100>, <MODBUS>, <SNAP7>, <PROGRAMMING LANGUAGE>, <ELECTRIC PARAMETERS>, <GRAPHIC INTERFACE>.

CAPÍTULO I

1. INTRODUCCIÓN

1.1 Antecedentes

Dentro de un sistema SCADA (Control Supervisor y Adquisición de Datos, según sus siglas en inglés) se encuentran las herramientas necesarias para el desarrollo de tareas como el monitoreo y control de procesos desde una central computarizada (Rodríguez Penín, n.d.)

Los procesos pueden variar de acuerdo con su aplicación en planta, pero realizan las mismas actividades que son la recolección de información entre dispositivos, el accionamiento de actuadores o inicialización de procesos de acuerdo a algún algoritmo programado,

La filosofía de desarrollo del software libre permite que, gracias al trabajo cooperativo, un proyecto evolucione. El trabajar con software libre garantiza que un proyecto no pueda ser cancelado unilateralmente por las razones que fuesen, es decir, siempre y cuando existan interesados en continuar con el proyecto, éste seguirá desarrollándose (J. M. Hernández. “Software Libre: Técnicamente Viable, Económicamente Sostenible y Socialmente Justo”. 2005).

Según el Instituto Nacional de Ciberseguridad de España S.A. en su página de internet, existen algunos sistemas SCADA de visualización que son libres es decir no requieren de una licencia de pago, entre los que se puede mencionar. ScadaBR es un sistema de control completo que funciona en sistemas operativos Windows (Windows XP y Windows 7), disponible en la licencia de código abierto (software libre) desarrollado a partir de un proyecto de investigación. El software ha sido desarrollado en el marco de la CERTI. IGSS FREE50, Es la versión libre del software SCADA IGSS V11 creado por Schneider Electric. Esta versión libre dispone de todas las funcionalidades de la versión comercial, pero está limitado al uso máximo de 50 objetos. Se pueden crear proyectos con total funcionalidad.

Esta versión libre del SCADA IGSS también incluye proyectos preconfigurados que pueden ser utilizados en el modo Demo. Esta opción permite mostrar las capacidades del software sobre diversos entornos simulados. Funciona bajo entornos Windows.

IndigoSCADA es un software SCADA-DCS para las plataformas Windows y Linux. Está desarrollado principalmente en lenguaje C/C++. Sus características principales son uso de datos para generar informes de gestión ya sean diarios, semanales, mensuales, presentación gráfica de los datos de tiempo real e histórico, notificación de alarma y eventos en tiempo real soporta múltiples ventanas HMI entre otras.

TeslaMultiSCADA es un SCADA para el acceso a los datos de producción y de proceso desarrollado para dispositivos Android, utilizando protocolos industriales como Modbus TCP (UDP), Siemens ISO / TCP y Ethernet / IP para la comunicación.

La empresa proporciona una versión de evaluación que tiene las siguientes restricciones:

No se puede utilizar más de 16 etiquetas en el proyecto.

No se puede importar proyectos.

El período de evaluación es de 60 días.

A nivel Nacional en Ecuador se ha desarrollado sistemas HMI-SCADA con software libre en total de dos documentos es por eso que el desarrollo de este proyecto tiene como finalidad realizar este diseño para que pueda ser asequible a las pequeñas y medianas empresas que deseen expandir su visión la cual tiene por objetivo abaratar los costos, obteniendo una herramienta para el control y monitoreo de sus procesos.

1.2 Formulación del problema

- ¿Cómo diseñar e implementar un sistema de control y monitoreo de parámetros eléctricos dentro de procesos industriales mediante software libre y comunicación modbus?

1.3 Preguntas Directrices

- ¿Cómo identificar los parámetros en tiempo real con su producción que sean esenciales para llevar a PLC mediante el módulo CM 1241 con protocolos de comunicación?

- ¿Qué tipo de software libre nos ayudará a diseñar la interfaz gráfica para la verificación de los requerimientos?
- ¿Qué nivel de adaptabilidad tendrá el diseño para implementar en la interfaz?
- ¿Qué tipos de comunicación permite al sistema la mejor adquisición de datos del proceso industrial?

1.4 Justificación

En la actualidad la mayoría de las industrias cuentan con procesos de producción cada vez más exigentes, por lo que, los fallos durante el funcionamiento producen paros de emergencia y tiempos muertos, causando un impacto financiero negativo.

Con el objetivo de reducir al mínimo este tipo de fallos, es necesaria la supervisión constante de los equipos principales, lo que deriva en un alto consumo de recursos debido a que se debe evaluar cada máquina en el sitio con dispositivos de medición convencionales.

Debido a esto, es una necesidad automatizar este proceso de adquisición de los valores de parámetros como; velocidad, corriente, frecuencia y temperatura, propios del funcionamiento de los equipos inmersos en un proceso productivo, para que nos ayude a conocer características importantes de su desempeño.

La visualización de los parámetros a analizar es muy importante, ya que de eso depende los resultados y las decisiones tomadas en relación con el equipo sometido a pruebas. De acuerdo con esto se debe ver la mejor técnica de adquisición de datos en función a la comodidad y fidelidad al momento de elegir el instrumento de visualización.

Esta sería una opción viable para visualizar y analizar los datos se opta por el uso de la computadora portátil la visualización y análisis de los datos que se va a obtener, pero los altos costos de las licencias de programas dedicados a la elaboración SCADA y OPC, hace que se opte por HMI, aunque posiblemente tengan recursos limitados para su personalización.

El control y monitoreo de datos y parámetros de un dispositivo o de una máquina completa a distancia es primordial en cualquier empresa productiva, pero en las pequeñas y medianas posiblemente no tengan accesibilidad a las licencias o programas por su elevado costo. Dada esta problemática que se presenta existe la necesidad inminente de

automatizar los procesos abaratando costos como se propone en este trabajo de investigación

1.5 Objetivos de la Investigación

1.5.1 Objetivo general

Diseñar e implementar un sistema de control y monitoreo de parámetros eléctricos dentro de procesos industriales mediante software libre y comunicación modbus.

1.5.2 Objetivos específicos

- Identificar los parámetros eléctricos en tiempo real con su producción que sean esenciales para llevar a PLC mediante el módulo CM 1241 con comunicación RS485.
- Obtener de los parámetros eléctricos de voltaje, corriente, frecuencia, factor de potencia energía activa y reactiva dentro del ambiente industrial mediante un medidor de parámetros eléctricos.
- Programar el dispositivo de control mediante el software de ingeniería para la comunicación con el medidor de parámetros eléctricos y posterior almacenamiento de los parámetros obtenidos.
- Diseñar mediante software libre la interfaz gráfica de usuario de acuerdo con los requerimientos y necesidades del proceso industrial.
- Implementación de la interfaz.
- Comunicar el PLC mediante Modbus a los esclavos, TCP IP a la PC y a su vez enviar la información a la HMI para su monitoreo y control.

1.6 Hipótesis

Con el desarrollo de una HMI basada en software libre se contribuye a que pequeñas y medianas empresas tengan acceso a tecnología de calidad a un precio accesible, incrementando así su capacidad de competitividad en el mercado.

El sistema basado en software libre equipara en funcionalidad, rapidez y eficiencia a un sistema con licencias de paga, ejecutando las mismas órdenes con tiempos de respuesta similares, abaratando costos a las pequeñas y medianas industrias.

La obtención de los parámetros eléctricos (corriente, velocidad, frecuencia, temperatura), ayudan de forma rápida y eficiente a comprobar el estado y comportamiento de el o los equipos sometidos a estudio.

Mediante las redes utilizadas aseguramos que los datos obtenidos mediante el supervisor de red lleguen al dispositivo de control en forma fiable y rápida con la mínima pérdida de datos.

CAPÍTULO II

2 MARCO TEÓRICO

2.1. Sistemas de Control

Se conoce al instrumento inteligente diseñado para el uso en entornos industriales encaminados al control de procesos automatizados, dichos módulos industriales se encuentran en un progreso constante tanto en su software como en su hardware, ofreciendo ventajas de instalación por una gran sencillez en su tamaño, modelo y un avance en lo que refiere al software de control de los dispositivos industriales.

Este sistema se define como el conjunto de elementos que funcionan de manera encadenada para dar como resultado una respuesta deseada partiendo de una determinada entrada. Los elementos que los forman son: objetivos de control, componentes del sistema de control y resultados o salidas.

Los objetivos de control componen las entradas del sistema, los componentes del sistema de control son los elementos electrónicos o mecánicos que desarrollan la lógica de control del sistema, y los resultados o salidas representan las respuestas obtenidas del sistema. (Juan & Alberto Perez Ing Analía Perez Hidalgo Bioing Elisa Perez Berenguer, n.d.)

En la figura 2-1 se observa los elementos descritos del sistema de control.

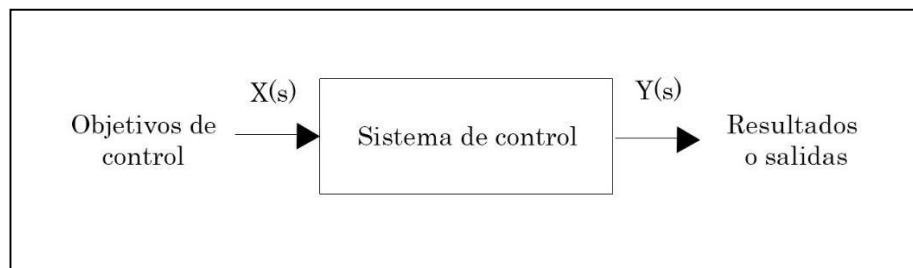


Figura 2-1. Componentes básicos de un sistema de control.

Realizado por: Romero F. 2022

Estos sistemas de control se pueden clasificar en dos tipos, los sistemas de control en lazo abierto y los sistemas de control en lazo cerrado.

2.1.1. Sistema de control en lazo abierto

Son un tipo de sistema en el cual la salida no se ve afectada por la entrada, lo que significa, carecen de una señal de error derivada a partir de la comparación de la salida actual con la entrada del sistema. Los sistemas presentan un tipo de respuesta basada únicamente en los datos iniciales y en el transcurso del tiempo.

(Carrillo Paz, 2011)

En la figura 2-2 se observa el diagrama de elementos de control d lazo abierto.

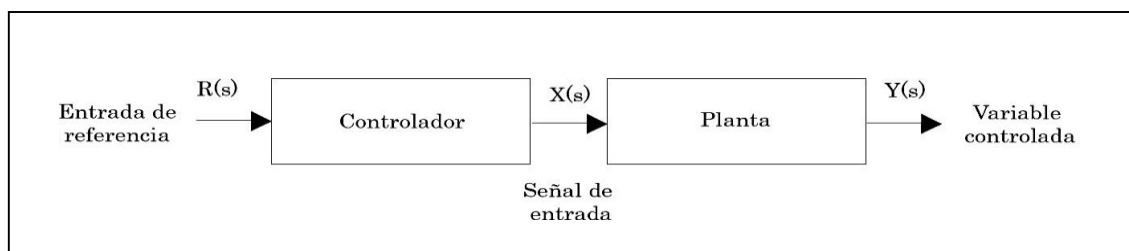


Figura 2-2 Elementos de un sistema de control a lazo abierto

Realizado por: Romero F. 2022

2.1.2 Sistema de control en lazo cerrado

Este tipo de sistema que presenta un lazo de retroalimentación, el mismo que toma una señal de salida y la compara con la señal de referencia para obtener una señal de error. Esta señal ingresa en el controlador, el cual tomará una decisión basada en la magnitud de ese error. Debido a esta lógica de control estos sistemas muestran insensibilidad ante posibles perturbaciones que puedan darse dentro de un sistema. (Carrillo Paz, 2011)

En la figura 2-3 se presenta el diagrama correspondiente a un sistema de control en lazo cerrado.

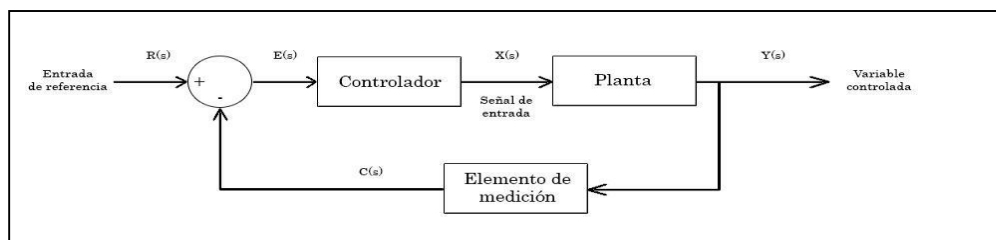


Figura 2-3 Elementos de un sistema de control a lazo cerrado

Realizado por: Romero F. 2022

2.2 Tipos de controladores

Los controladores automáticos son aquellos que se encargan de comparar el valor real de la salida con el valor de referencia del sistema para obtener la desviación o error, la misma que emite una señal de control esta busca reducir el error calculado. La forma en la que el controlador causa esta señal de control se denomina acción de control, la cual se puede clasificar en diferentes tipos, los cuales se detallan a continuación. ((99+) *Ingenieria-de-Control-Moderna-Ogata-5ed* / Jefferson Maciel Leite - Academia.Edu, n.d.)

2.2.1. Controlador ON-OFF

Este sistema de control presenta dos posiciones fijas que generalmente son encendido y apagado. El control es relativamente simple y de bajo coste, y es implementada en una serie de procesos a nivel industrial y doméstico. ((99+) *Ingenieria-de-Control-Moderna-Ogata-5ed* / Jefferson Maciel Leite - Academia.Edu, n.d.)

El comportamiento con respecto al tiempo se muestra en la figura 2-4.

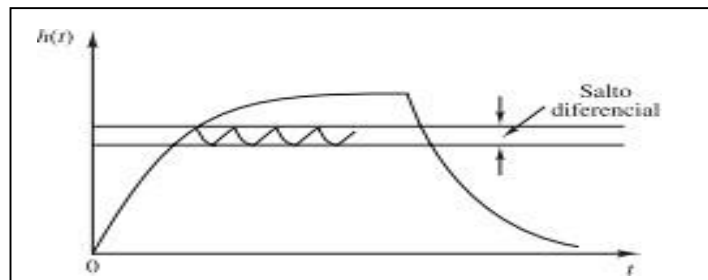


Figura 2-4 Comportamiento del controlador ON-OFF con respecto al tiempo

Fuente: ((99+) *Ingenieria-de-Control-Moderna-Ogata-5ed* / Jefferson Maciel Leite - Academia.Edu, n.d.)

2.2.2 Controlador proporcional (P)

Este controlador es un amplificador con ganancia ajustable. Su comportamiento a lo largo del tiempo, donde k_p es la ganancia proporcional del controlador. ((99+) *Ingenieria-de-Control-Moderna-Ogata-5ed* / Jefferson Maciel Leite - Academia.Edu, n.d.)

2.2.3 Controlador integral (I)

Este se dispone de una razón proporcional a la señal de error, dando lugar a k_i , la cual es una constante ajustable. ((99+) *Ingenieria-de-Control-Moderna-Ogata-5ed* / Jefferson Maciel Leite - Academia.Edu, n.d.)

2.2.4 Controlador proporcional integral (PI)

Este controlador combina los efectos de la acción proporcional e integral, brindando de esta manera una lógica de control capaz de reconocer a la desviación de la salida y el tiempo en que se mantiene esta desviación. ((99+) *Ingenieria-de-Control-Moderna-Ogata-5ed* / Jefferson Maciel Leite - Academia.Edu, n.d.)

2.2.5 Controlador proporcional integral derivativo (PID)

Este tipo de controlador combina las tres acciones de control descritas en los otros controladores, brindando una lógica de control capaz de responder a variaciones de la señal de entrada, desviaciones de la señal de salida y además producir una ganancia proporcional al error, suele ser utilizado en procesos de tipo industrial que requieran grados de exactitud y una rápida estabilización. ((99+) *Ingenieria-de-Control-Moderna-Ogata-5ed* / Jefferson Maciel Leite - Academia.Edu, n.d.)

2.3 Controladores Lógicos Programables

2.3.1 Definición

La norma internacional IEC 61131 indica que un controlador lógico programable se define como “una máquina electrónica programable capaz de ejecutar un conjunto de instrucciones organizadas de una manera adecuada, con el propósito de resolver algún problema dado. Estos son programables de manera que están diseñados para trabajar de manera industrial donde sea víctima de un contexto hostil. Las instrucciones de este dispositivo son capaces de controlar procesos, porque tienen funciones lógicas, operaciones aritméticas, de conteo y eventos, de temporización. Además, el PLC es diseñado para que sea capaz de cumplir su tarea de control de una forma rápida y sencilla, con una serie de entradas y salidas de tipo digital y analógico”. (*Entornos Integrados de Automatización*, 6113)

Estos dispositivos se usan en instalaciones que necesiten procesos de maniobra, control y señalización, pueden definir como campo aplicativo de estos autómatas los sitios donde se lleve a cabo procesos de transformaciones de tipo industrial, o instalaciones donde se necesite un control de sus procesos. (*CONTROLADORES LÓGICOS PROGRAMABLES (PLC) - Micro Automación*, n.d.)

Algunos ejemplos de su campo aplicativo pueden ser:

- Maniobra de embalajes
- Maquinaria industrial de plásticos
- Maniobra de maquinaria
- Maniobra de instalaciones de calefacción
- Instalaciones de seguridad
- Señalización y control de plantas industriales

Este tipo de controlador lógico programable puede solventar necesidades tales como:

- Procesos de producción periódicamente cambiantes
- Procesos secuenciales

- Instalaciones de procesos extensos y complejos
- Maquinaria de procesos variables

2.1.1. Funciones

Los controladores lógico-programables (PLC) establecen los autómatas programables que se utilizan en la industria, estos brindan una serie de funciones, las cuales son clave para realizar un buen control en la industria, estas funciones son las siguientes:

Detección: Estos son capaces de leer diferentes tipos de señales emitidas por los captadores o sensores distribuidos por toda la planta.

Mando: Elabora y envía las instrucciones que se deben realizar al sistema, mediante accionadores y preaccionadores.

Programación: Da la posibilidad de que se introduzca, se elabore o incluso se cambie el programa del autómata, incluso mientras éste se encuentre llevando a cabo un proceso.

Redes de comunicación: Usan una serie de protocolos de comunicación, que permiten intercambiar información con otros autómatas de la línea de producción en cuestión de milisegundos, brindando la posibilidad de controlar varias partes de la planta sin necesidad de movilizarse por todas las estaciones de trabajo.

Sistemas de supervisión: Permiten la comunicación con computadores que dispongan de algún software de supervisión industrial.

Control de procesos continuos: Tienen un sistema de control bastante preciso de procesos de tipo continuo, con una serie de módulos de entradas y salidas de tipo analógico, lo que posibilita ejecutar controladores de tipo PID, que pueden estar programados dentro del autómata.

Buses de campo: Permiten mediante un solo cable de comunicación se puedan conectar al bus otros dispositivos adyacentes al proceso. (Mandado Pérez et al., 2018).

2.1.2. Estructura interna

Los autómatas tienen una estructura interna con ciertas partes esenciales, incluso los autómatas más simples deben presentar este tipo de estructura, para que su control sea eficaz dentro del proceso donde han sido instalados. Esta estructura esencial es la siguiente:

- Sección de entradas: Contiene una serie de líneas de entrada que se encargarán de recibir las señales provenientes de los sensores presentes en el proceso, éstas pueden ser de tipo digital o analógico.
- Sección de salidas: Presenta una línea completa para salidas, estas al igual que las entradas pueden ser de tipo digital o analógico, y se encargan de enviar las señales de control o alerta desde el PLC al resto del proceso.
- Unidad central de proceso: Llamado CPU, se encarga de procesar el programa o el conjunto de instrucciones que el proceso requiera, usando su banco de memoria, instrucciones de programa y registros. (*CONTROLADORES LÓGICOS PROGRAMABLES (PLC) - Micro Automación*, n.d.)

Los autómatas de estructura más compleja pueden presentar otro tipo de elementos, los cuales permiten aligerar el proceso de interacción entre humano y máquina, que sirve para optimizar el proceso en sí, contando en algunos casos con sistemas de control incorporados que permiten la toma de decisiones mucho más complejas, un ejemplo de esto es la presencia de un controlador PID.

Podemos encontrar algunos elementos dentro del PLC:

- Unidad de alimentación: Esta unidad se encarga de suministrar las tensiones necesarias para el funcionamiento de los diferentes circuitos que estén presentes en el sistema.
- Unidad de programación: Esta sección permite modificar de una manera mucho más clara

y sencilla el programa de usuario.

- Dispositivos periféricos: Son dispositivos que dan al controlador lógico de una mayor cantidad de unidades de entrada, salida y capacidad de memoria.
- Interfaces: Son elementos, que presentan la configuración del controlador lógico en un entorno gráfico, puede llevar a cabo una comunicación fluida entre el operario y el PLC. (Patricia & Martinez, n.d.)

La figura 2-5, indica un diagrama que representa los elementos internos de un controlador lógico programable complejo.

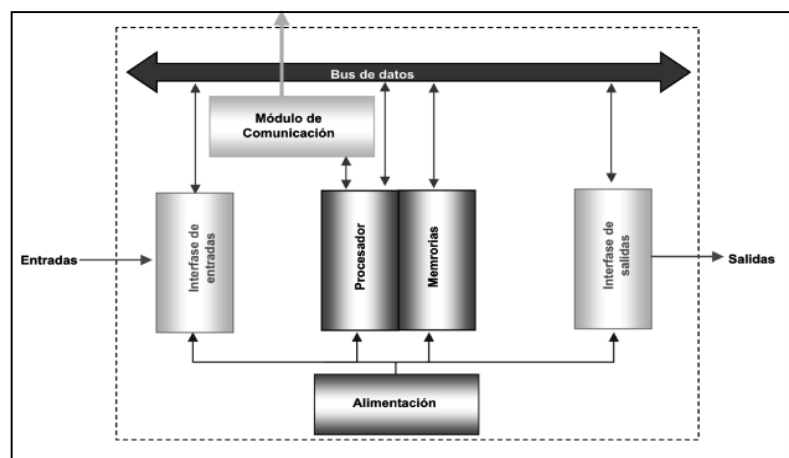


Figura 2-5 Estructura interna de un controlador lógico programable

Fuente: (CONTROLADORES LÓGICOS PROGRAMABLES (PLC) - Micro Automación, n.d.)

2.1.3. Estructura externa

La estructura es una construcción similar a un cubo, en la cual se disponen las conexiones necesarias para las entradas y salidas del PLC, así como otros elementos que se puedan presentar en PLC's más complejos que dispongan de ciertas funciones especiales, como pueden ser módulos wifi o módulos de comunicación.

Dependiendo del tipo de distribución que se necesite, un controlador lógico programable puede dividirse principalmente en 4 tipos, el PLC compacto, modular, con estructura americana y con estructura europea, los cuales se detallan a continuación:

- Compacto: Corresponde a un tipo de estructura en la cual todos los elementos se disponen en un solo bloque.
- Modular: Esta estructura presenta una serie de ranuras de expansión, en las cuales es posible conectar ciertos módulos que permiten aumentar el número de entradas y salidas disponibles en el PLC.
- Estructura americana: Es un tipo disposición en la cual las entradas y salidas del controlador lógico no están en el bloque principal, y mediante conexiones son separadas del mismo.
- Estructura europea: Este tipo de estructura presenta una serie de módulos, los cuales cada uno de ellos dispondría de una función diferente, como pueden ser el CPU, la unidad de alimentación, la unidad de comunicación, las entradas y salidas, entre otros. *(Repositorio Digital - EPN: Diseño y Construcción de Un Controlador Autómata Programable En Lenguaje FBD Con Software de Simulación, n.d.)*

2.1.4. Sistema de entrada y salida

Todo controlador lógico programable necesita un medio de comunicación con su entorno. Esto lo consigue a través del sistema de entradas y salidas, los cuales corresponden a un conjunto de interfaces E/S que posibilitan la conexión entre la CPU y la planta. Cada una de estas entradas y salidas están identificadas mediante una tabla de direcciones las cuales deben ser consideradas al momento de realizar el diseño de la parte de mando.

Dependiendo el tipo de PLC el número de entradas y salidas disponibles pueden ser ampliados para cumplir con las necesidades del sistema. *(Técnicas de Automatización Avanzadas En Procesos Industriales - Dialnet, n.d.)*

Los sistemas de entradas y salidas presentan dos configuraciones principales las cuales son los sistemas de E/S centralizados y los distribuidos. Los sistemas centralizados son aquellos donde las interfaces de E/S se comunican directamente con el controlador lógico mediante su bus interno de datos, sin la necesidad de utilizar procesadores de comunicación. Mientras que los sistemas distribuidos son aquellos que utilizan procesadores de enlace de E/S conectados al bus interno de datos del PLC. Esta conexión posibilita la comunicación entre los módulos de E/S y la CPU del autómata.

2.1.4.1. Señales digitales

Este tipo de señales se pueden encontrar en sensores de infrarrojos, inductivos, capacitivos entre otros, utilizados para detectar la presencia de cierta sustancia o cuerpo. A continuación, se presenta una representación temporal de una señal digital. (Miyara & Nacional De Rosario, n.d.)

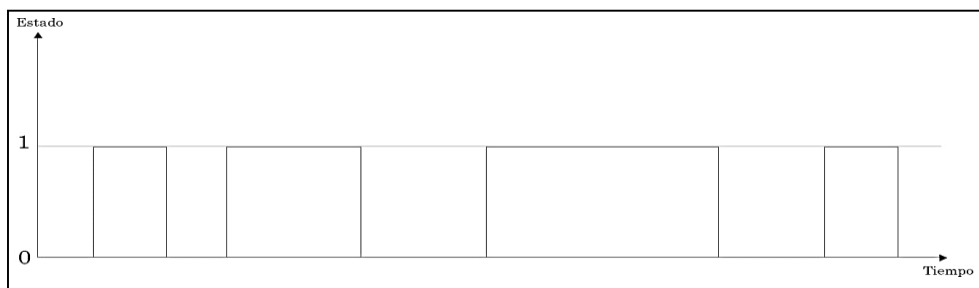


Figura 2-6 Señal digital a lo largo del tiempo

2.1.4.2. Señales analógicas

15

Este tipo de señales se presentan al medir magnitudes físicas tales como temperatura, donde se pueden encontrar sensores que emitan señales analógicas tanto de voltaje como de resistencia eléctrica. También se pueden encontrar al medir humedad, presión, sonido, distancia, entre otras.

Las señales de este tipo normalmente son normalizadas, el cual es un término que hace referencia a buscar un estado de la señal donde los límites inferior y superior de la magnitud coincidan con los límites del dispositivo que se utilice. Esta normalización busca el mejor aprovechamiento de la relación señal/ruido del dispositivo de medida. (Miyara & Nacional De Rosario, n.d.)

A continuación, se presenta una representación a lo largo del tiempo de una señal analógica.

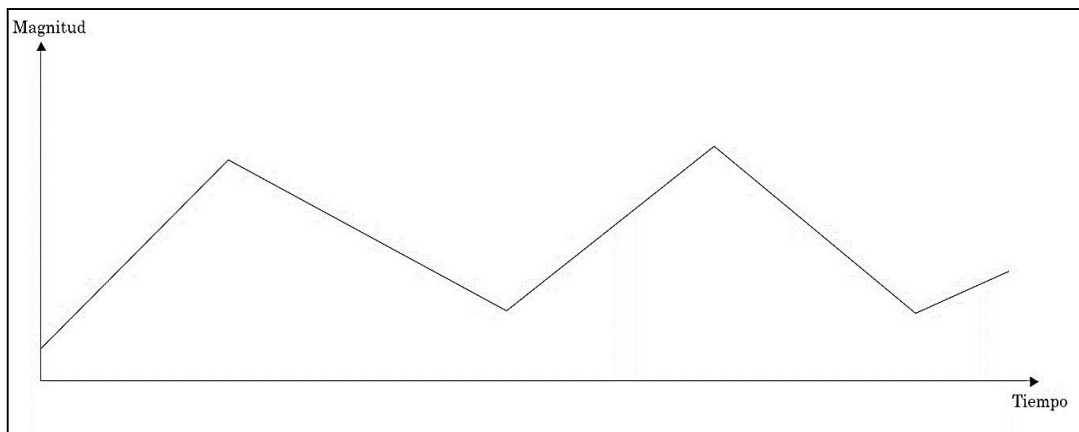


Figura 2-7 Señal analógica a lo largo del tiempo.

Realizado por: Romero F. 2022

2.2. Protocolos de Comunicación Industrial

Los protocolos de comunicación constituyen un conjunto de normas que se deben cumplir por dos o más dispositivos que deseen entablar un intercambio de información entre sí. Estos protocolos disponen de una semántica, una sintaxis, una temporización y una serie de reglas que aseguran que la comunicación entre dispositivos sea satisfactoria.

Los protocolos de comunicación que se pueden encontrar a nivel industrial son: protocolo Hart, protocolo Modbus, protocolo Profibus, protocolo CCM, protocolo DeviceNet y protocolo SNP. Estos protocolos se detallan a continuación. (de Comunicación, n.d.)

- 2.4.1 *El protocolo Hart* es uno de los estándares líderes a nivel industrial, dispone de una serie de instrumentos inteligentes de campo, y su manera de enviar información es mediante el envío de señales a diferentes frecuencias. Para enviar un valor lógico de 1 este protocolo envía una señal de 1200 Hz superpuesta a la señal del lazo, y cuando se desea enviar un valor lógico de 0 se utiliza una señal a 2200 Hz. (de Comunicación, n.d.)

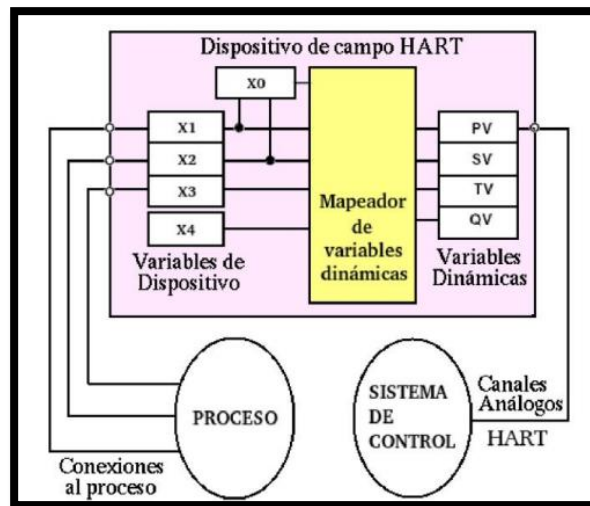


Figura 2-8 Trama de protocolo de comunicación

Fuente: (Barandica López, 2010)

- 2.4.2 *El protocolo DeviceNet* utiliza una definición fundamentada en la orientación a objetos para modelar los servicios de comunicación y el comportamiento externo que presente en los nodos. Este protocolo puede trabajar con conexiones de maestro y esclavo, interrogación cíclica, mensajes espontáneos de cambio de estado, comunicación uno a uno, carga y descarga de bloques de datos. (de Comunicación, n.d.)

1 bit	11 bits	1 bit	6 bits	0-8 bytes	15 bits	1 bit	1 bit	1 bit	7 bits	3 bits
Início da Frame	Identificador	RTR bit	Campo de Controle	Campo de DADOS (Variável de 0 a 8 bytes)	Sequenciador CRC	Delimitador de CRC	Bit de Ack	Delimitador de Ack	Final da Frame	Espaço entre Frames

Figura 2-9 Trama de protocolo de comunicación DeviceNet

Fuente: (DEVICENET | Facultad-Ingenieria, n.d.)

2.4.3 *El protocolo Profibus es un estándar cuyo origen data en la década de los 80, es utilizado principalmente en aplicaciones de transmisión de datos a alta velocidad.*

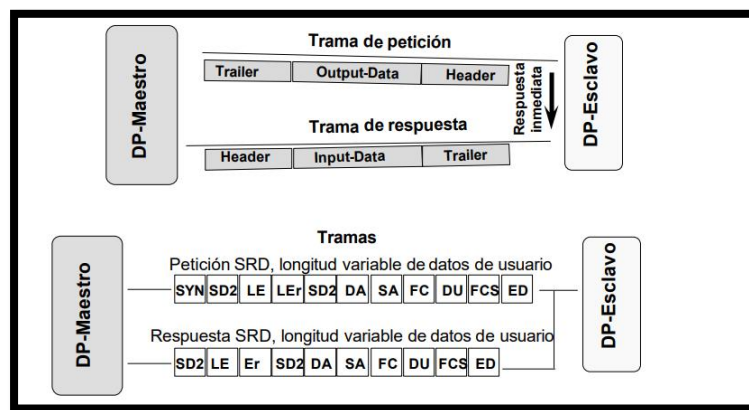


Figura 2-10 Trama de protocolo de comunicación Profibus

Fuente: (Sistemas Industriales Distribuidos Profibus Profibus, n.d.)

2.4.4 Protocolo Modbus Tcp Ip

Modbus es un protocolo de comunicaciones publicado por Modicon en 1979, diseñado para trabajar con equipos industriales tales como Controladores Lógicos Programables (PLC's) computadoras, motores, sensores y otros tipos de dispositivos electrónicos.

Simple y robusto, se ha convertido en un protocolo estándar de comunicación, siendo uno de los más comunes en industria para la comunicación de dispositivos electrónicos.

Es un protocolo tan extendido debido a que es público, su implementación es sencilla, requiere tiempos de desarrollo reducidos y maneja bloques de datos sin suponer restricciones

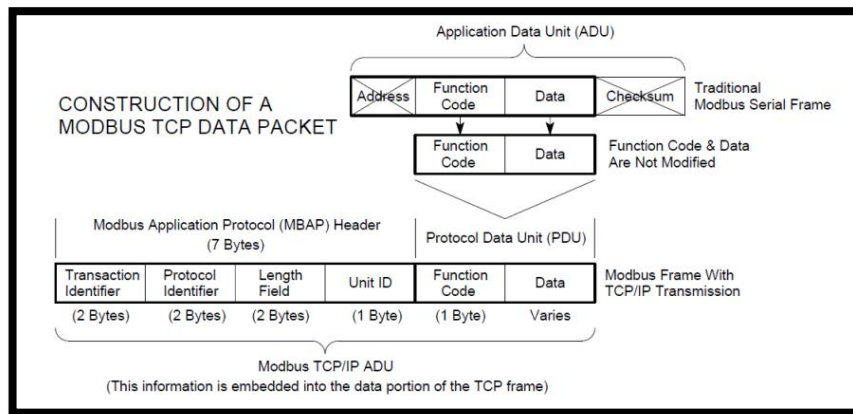


Figura 2-11 Trama de protocolo de comunicación Modbus TCP IP

Fuente: (Interface de Comunicaciones CEM M-ETH, n.d.)

2.5 Interfaz Hombre Máquina

La interfaz hombre máquina (HMI), forma parte del programa informático que se comunica con el usuario. En ISO 9241-110, el término interfaz de usuario se define como "todas las partes de un sistema interactivo (software o hardware) que proporcionan la información y el control necesarios para que el usuario lleve a cabo una tarea con el sistema interactivo".

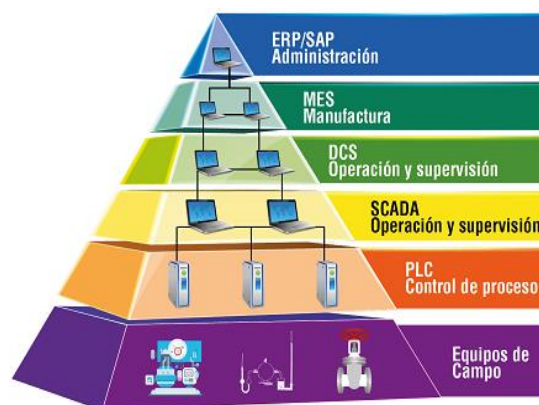


Figura 2-12 Pirámide de automatización

Fuente: (Pirámide de Automatización, n.d.)

CAPÍTULO III

3. DESARROLLO DE LA INVESTIGACIÓN

3.1. Metodología

En este capítulo se describe cada uno de los pasos a seguir para el desarrollo de la investigación, en el cual se implementa el tablero de control, se diseña la red de comunicaciones y se desarrolla la interfaz gráfica programada en python, con la finalidad de cumplir los objetivos planteados y demostrar la hipótesis.

3.2. Diseño general del sistema

La idea fundamental de todo sistema de control es aumentar la eficiencia en todo proceso en el que intervenga la automatización, para con esto tratar de reducir riesgos en toda la producción y evitar pérdidas económicas.

Con esta implementación enlazaremos el PLC S7 – 1200 junto con el variador de frecuencia Sinamics V20 concatenados con el resto de los elementos que complementan el sistema de control.

Para ello se identificó parámetros eléctricos indispensables en los procesos productivos que nos permitan obtener información necesaria para optimizar tiempos y recursos.

La programación en el PLC S7-1200 se la realizó en el TIA PORTAL, mediante la comunicación Modbus se acoplo el variador de frecuencia SIMATICS V20 y el medidor de parámetros eléctricos SENTRON PAC 3100, que permiten obtener los datos que posteriormente se reflejaran en la HMI, que se puede observar en el diagrama de bloques de la figura 3-1.

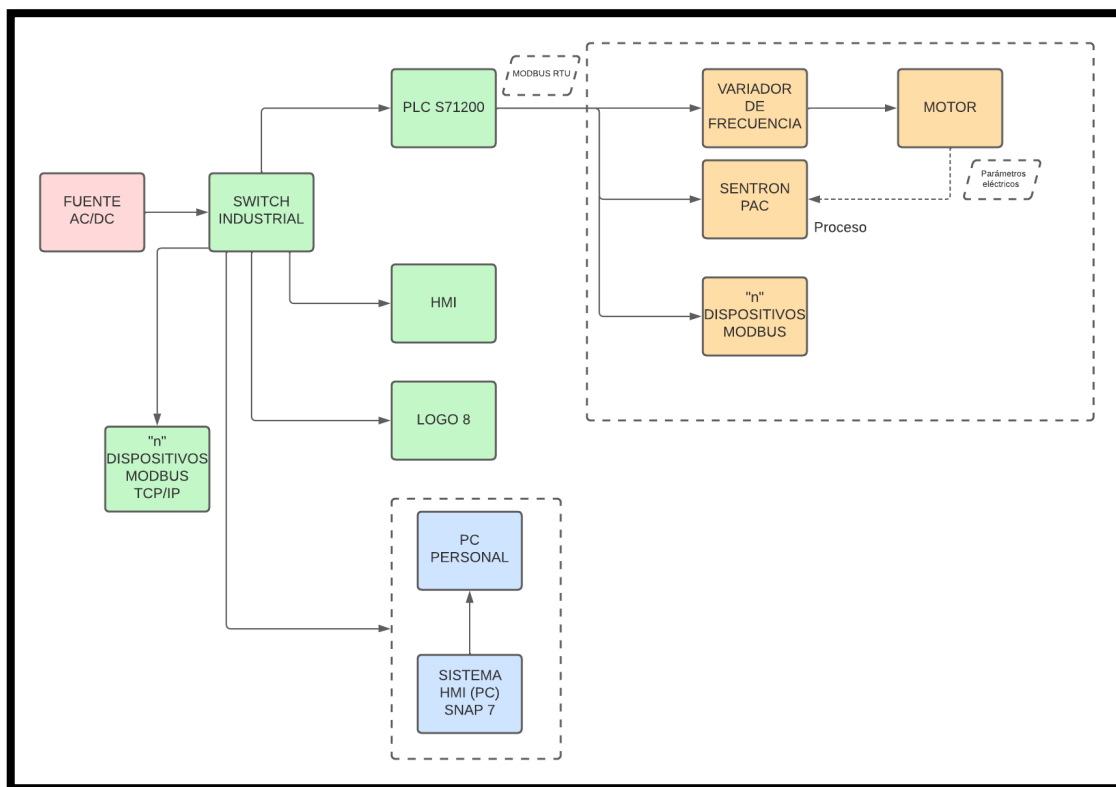


Figura 3-1 Diagrama de Bloques

Realizado por: Fredy Romero, 2022

3.3. Selección del tablero de control

El sistema de control y monitoreo de parámetros eléctricos está implementado en un tablero S – Box, gabinete de metal de dimensiones 60x40x20 cm, aquí se monta todo el sistema de control. Este tablero fue seleccionado de acuerdo con la norma IEC 61439, con las especificaciones de los conjuntos de aparata de baja tensión.



Figura 3-2 Tablero eléctrico

Realizado por: Fredy Romero, 2022

3.3.1. Canaleta Ranurada y Riel Din

En el tablero, los rieles se ubican de manera horizontal en donde montamos los elementos del sistema de control, para luego pasar el cableado por las canaletas realizando todo el ajuste mediante tornillos de acero. Por cada una de las canaletas pasará todo el cableado respectivo de los elementos que interconectan el sistema.



**Figura 3-3 Canaleta Ranurada
y Riel Din**

Realizado por: Fredy Romero, 2022

3.3.2. Controlador Simatics S7-1200

Montado sobre los rieles del tablero el controlador tiene las siguientes especificaciones: SIMATIC S7-1200, CPU 1214C, CPU compacta, AC/DC/Relé, E/S integradas: 14 ED 24VDC; 10 SD relés 2A; 2 AI 0 - 10V DC, Alimentación: AC 85 - 264 V AC BEI 47 - 63 HZ, Memoria de programa/datos 100 KB.

La función de recibir la información para realizar el control y automatizar con altas prestaciones de memoria y velocidad de información, tiene múltiples entradas y salidas analógicas y digitales a las cuales se le puede adaptar módulos que nos permite expandir su uso dependiendo el proceso en el cual se desee utilizar.

En este caso nos permite controlar el motor y monitorear mediante el módulo de comunicación RS 232/485 los parámetros eléctricos, con el protocolo de comunicación modbus, el variador, el cual recibe la información del controlador da una respuesta de salida hacia el motor.



Figura 3-4 Controlador Simatics S7-1200

Realizado por: Fredy Romero, 2022

3.3.3. Sinamics V20

Montado sobre los rieles del módulo permite dar una rápida respuesta al motor ya que es monofásico, en esta implementación permite variar la velocidad del motor a la requerida indicando en la interfaz la frecuencia a la cual se desea que este desarrolle su trabajo.



Figura 3-5 Sinamics V20

Realizado por: Fredy Romero, 2022

3.3.4. Logo! 8

Montado sobre los rieles del tablero, es un relé programable que en este caso me permite controlar la temperatura ya sea en el ambiente, pero en este caso controlamos temperatura del motor por lo que es necesario realizar una pequeña programación.



Figura 3-6 Logo ;8

Realizado por: Fredy Romero, 2022

3.3.5. Módulo de comunicación CM1241

Montado en los rieles del módulo en conexión con el controlador S7 - 1200, en este proceso me permite establecer la comunicación MODBUS con el variador de frecuencia SINAMICS V20. Esta interfaz realiza una conexión analógica mediante el protocolo de transferencia de datos diseñados para controlar la comunicación serie, organiza estos datos que fluyen desde un equipo terminal de datos a un receptor u otro equipo de comunicación de datos en este caso el controlador lógico programable.



Figura 3-7 Módulo de comunicación CM1241

Realizado por: Fredy Romero, 2022

3.3.6. Medidor multifunción Siemens Sentron PAC3100

Este es un instrumento compacto indicado para la medición y visualización de diferentes parámetros de red con toma de corriente que puede conectarse en redes monofásicas o trifásicas.

El mismo detecta los valores energéticos de consumidores individuales o de derivaciones eléctricas. Además de tensión y corriente, permite medir la potencia activa, reactiva y aparente, este medidor lleva integrado una interfaz RS485 que le permite una integración plena en las infraestructuras existentes utilizando el protocolo Modbus RTU.



**Figura 3-8 Medidor multifuncional
Siemens Sentron PAC3100**

Realizado por: Fredy Romero, 2022

3.3.7. Bobina de corriente TC

Los transformadores de corriente forman parte de los transformadores de instrumentos, que reducen señales de la corriente y tensión, pueden ser conectados a las entradas de los instrumentos de medida y a los relés de protección.

Estos aíslan y protegen los equipos de medida y protección de los altos niveles de tensión. En este caso nuestro TC mide la corriente del sistema de potencia y la corriente que pasa al motor. Tiene una capacidad de carga: 2.5VA a 15VA, una tensión máxima de 600V, y una frecuencia de 50/60Hz.



Figura 3-9 Bobina de corriente TC

Realizado por: Fredy Romero, 2022

3.3.8.Motor

El motor seleccionado para la implementación de este sistema de control es de marca Siemens; con una potencia de 0,5 hp, motor trifásico de 1200 rpm (6 polos). Con una eficiencia de 64,3 %, conectado a 220V y 2,2 amperios.



Figura 3-10 Motor

Realizado por: Fredy Romero, 2022

3.4. Red de Comunicación

La comunicación juega un papel importante dentro del sistema, por lo que se debe asegurar la fiabilidad en el envío y recepción de información de las variables de entrada y salida tanto para el control como para la visualización de resultados.

Los tipos de comunicación utilizados se describen a continuación.

3.4.1.Comunicación RS 232/485

Este protocolo se utiliza para establecer la comunicación entre el variador de frecuencia y el PLC de modo que se puedan ajustar los parámetros de control del motor, además de obtener las medidas de las variables eléctricas presentes en el funcionamiento del motor.

3.4.2.Comunicación Modbus TCP- IP

Este protocolo de comunicación permite interconectar los siguientes dispositivos:

PLC y el HMI de modo que el PIC recibe los parámetros de control enviados desde el HMI y envía las mediciones eléctricas del Motor al HMI



Figura 3-11 Conector DB9

Realizado por: Fredy Romero, 2022

3.5. Programación en el PLC

La programación se realizó en base a los requerimientos de los procesos a controlar, mediante el software TIA-PORTAL, se detalla a continuación:

En la configuración del dispositivo para el proyecto se ha seleccionado el CPU 1214 C AC/DC/RLY, bajo las necesidades del proyecto.

Es importante tomar en cuenta el número de RACK, así como el SLOT en el que se define el CPU puesto que, estos parámetros serán utilizados en la programación de la interfaz.

Se define también el módulo de comunicación a utilizar, para este proyecto es el CM1241 RS 422/485 y el módulo de dos salidas analógicas SM1232 AQ.

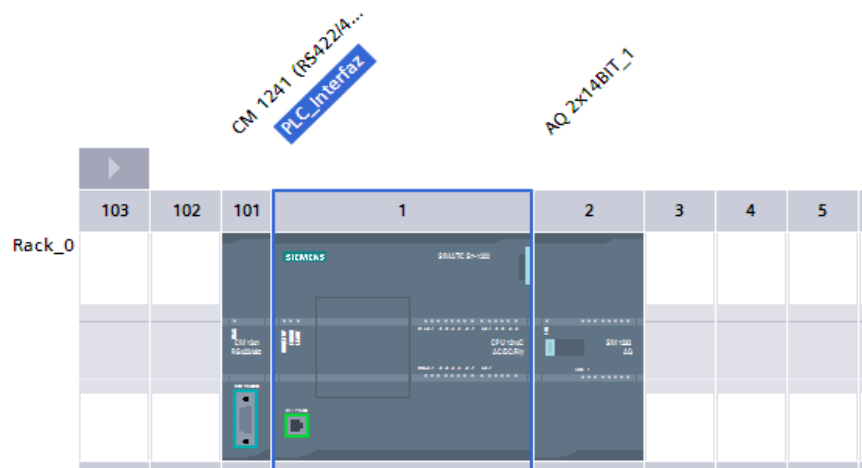


Figura 3-12 Módulo PLC

Realizado por: Fredy Romero, 2022

3.5.1. Variables empleadas para la conexión con el HMI.

Se tienen variables de tipo booleano utilizado para los botones de inicio, paro y la inversión de giro y variables de tipo entero para la lectura de los parámetros a ser medidos, tales como Frecuencia, velocidad, amperaje y temperatura.

	Nombre	Tipo de datos	Offset	Valor de arranq...	Remanen...	Accesible d...	Escrib...	Visible en ..	Valor de a...	Comentario
1	Static									
2	Inicio	Bool	0.0	false		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
3	Paro	Bool	0.1	false		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
4	Inversión	Bool	0.2	false		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
5	VelocidadSP	Int	2.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
6	VelocidadET	Int	4.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
7	Voltaje	Int	6.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
8	Corriente	Int	8.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
9	Frecuencia	Int	10.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
10	Velocidad	Int	12.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		

Figura 3-13 Declaración de variables

Realizado por: Fredy Romero, 2022

Estas variables de parámetros disponibles desde el variador de los cuales se ha tomado los datos de corriente, velocidad y frecuencia.

Nombre	Tipo de datos	Offset	Valor de arranq...	Remanen...	Accesible d...	Escrib...	Visible en ..	Valor de a..	Com
Static									
FreqOutppput	Int	0.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Speed	Int	2.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Current	Int	4.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Torque	Int	6.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
ActualPWR	Int	8.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
TotalKWH	Int	10.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
DCBusVolts	Int	12.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Reference	Int	14.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
RatedPWR	Int	16.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
OutputVolts	Int	18.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
FWDREV	Int	20.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
STOPRUN	Int	22.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
ATMaxFrec	Int	24.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
ControlMode	Int	26.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Enabled	Int	28.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
ReadyToRun	Int	30.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		

Figura 3-14 Tipos de variables

Realizado por: Fredy Romero, 2022

El dato de temperatura es una variable de tipo entero definido en un solo bloque.

Nombre	Tipo de datos	Offset	Valor de arranq...	Remanen...	Accesible d...	Escrib...	Visible en ..	Valor de a..
Static								
Temperatura	Int	0.0	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Figura 3-15 Variable temperatura

Realizado por: Fredy Romero, 2022

3.5.2.MB_COMM_LOAD

El bloque MB_COMM_LOAD se usa para seleccionar el módulo de comunicación, ajustar los parámetros de la comunicación y parametrizar el enlace con los parámetros del variador esclavo. El bloque MB_COMM_LOAD se debe llamar en el primer ciclo de programa (mediante la activación de la marca de sistema M1.0 en los ajustes de hardware o mediante una llamada en el OB 100 de arranque).

Después de insertar el módulo de comunicación en la configuración de hardware se puede seleccionar el nombre simbólico del módulo de comunicación en el parámetro PORT. Los parámetros de comunicación BAUD (velocidad de transmisión) y PARITY (paridad) deben ser idénticos para todos los nodos.

La configuración del puerto de la interfaz RS485 en la vista de equipo del STEP 7 (TIA Portal) es irrelevante en este caso, por consiguiente, define el módulo de comunicación (parámetro PORT) MODBUS.

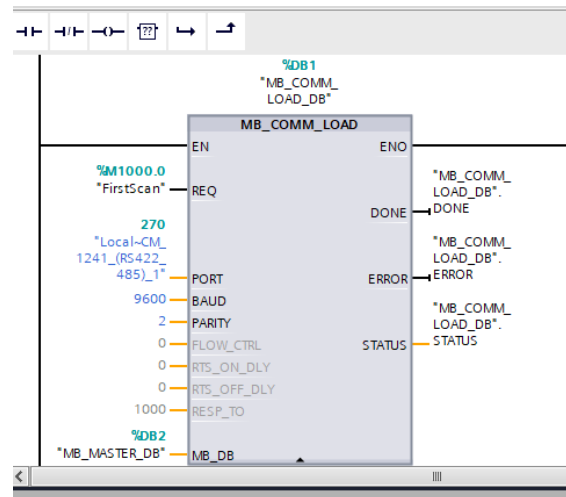


Figura 3-16 Bloque MB_COMM_LOAD

Realizado por: Fredy Romero, 2022

3.5.3.MB_MASTER

Se usa el bloque MB_MASTER para definir el módulo de comunicación seleccionado con el bloque de configuración MB_COMM_LOAD como maestro MODBUS.

El bloque MB_MASTER se usa para seleccionar el esclavo MODBUS a direccionar, seleccionar el código de función y definir la zona de almacenamiento de datos local. La tabla dada más abajo explica los parámetros.

El "MB_MASTER" se tiene que llamar en el primer ciclo de programa

- Mediante la activación de la marca de sistema M1.0 en los ajustes de hardware
- Mediante una llamada en el OB 100 de arranque.

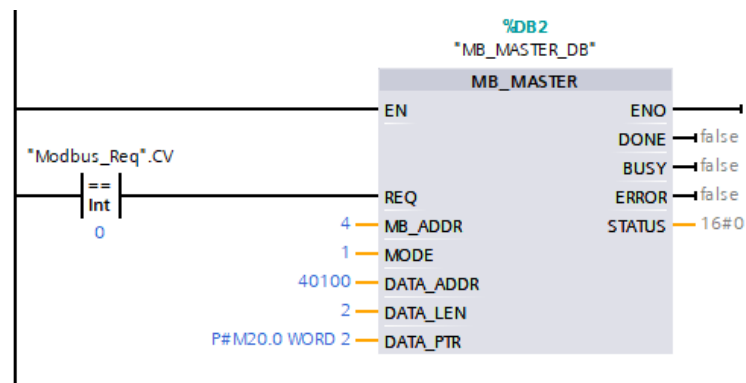


Figura 3-17 Bloque MB_MASTER

Realizado por: Fredy Romero, 2022

3.5.4. Contador ascendente

Cuando se activa el estado lógico del operando, se ejecuta la instrucción "Contador ascendente" y el actual valor se incrementa a uno.

El valor del parámetro PV se utiliza como límite para determinar la salida que devuelve el estado lógico uno, mientras el valor de conteo actual sea mayor o igual que el valor del operando. En todos los demás casos, la salida devuelve el estado lógico "0".

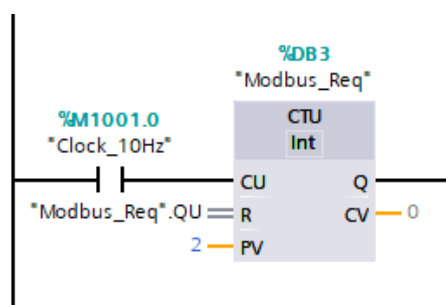


Figura 3-18 CTU – Contador ascendente

Realizado por: Fredy Romero, 2022

3.5.5. MOVE

La instrucción "Copiar valor" transfiere el contenido del operando de la entrada IN al operando de la salida OUT1. La transferencia se efectúa siempre por orden ascendente de direcciones.

La salida de habilitación ENO devuelve el estado lógico "0" cuando se cumple una de las condiciones siguientes:

La entrada de habilitación EN devuelve el estado lógico "0".

El tipo de datos del parámetro IN no puede convertirse al tipo de datos indicado en el parámetro OUT1.

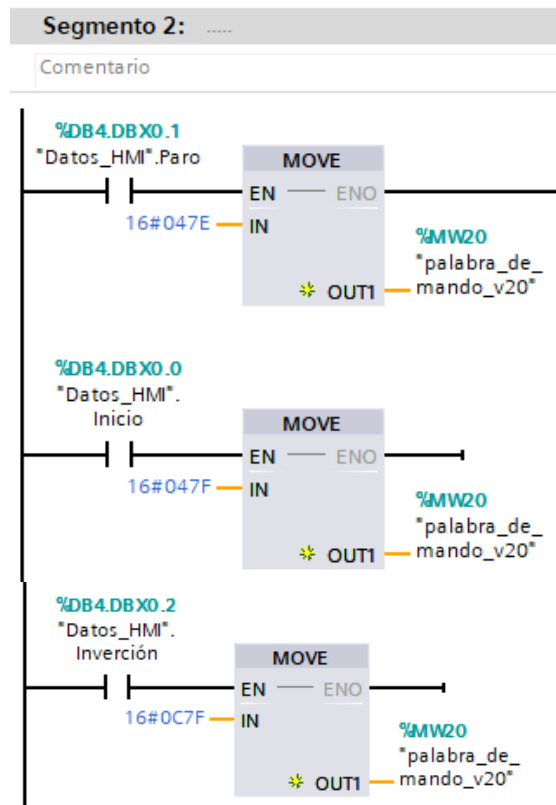


Figura 3-19 Move

Realizado por: Fredy Romero, 2022

3.5.6. Normalizar

La instrucción "Normalizar" normaliza el valor de la variable de la entrada VALUE representándolo en una escala lineal. Los parámetros MIN y MAX sirven para definir los límites de un rango de valores que se refleja en la escala. En función de la posición del valor que se debe normalizar en este rango de valores, se calcula el resultado y se deposita como número en coma flotante en la salida OUT.

Si el valor que se debe normalizar es igual al valor de la entrada MIN, la salida OUT devuelve el valor "0.0". Si el valor que se debe normalizar es igual al valor de la entrada MAX, la salida OUT devuelve el valor "1.0".

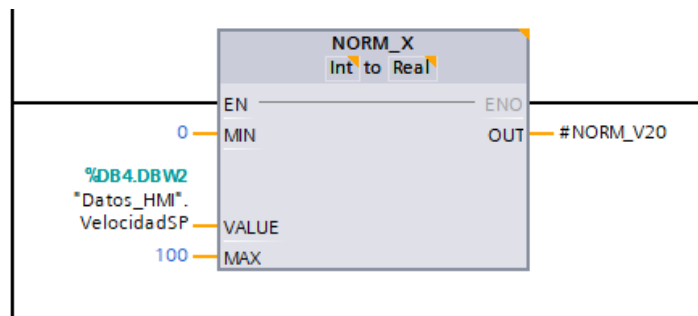


Figura 3-20 Normalizar

Realizado por: Fredy Romero, 2022

CAPÍTULO IV

4. ANÁLISIS Y RESULTADOS

4.1. Hipótesis

Con el desarrollo de una HMI basada en software libre se contribuye a que pequeñas y medianas empresas tengan acceso a tecnología de calidad a un precio accesible, incrementando así su capacidad de competitividad en el mercado.

El sistema basado en software libre equipara en funcionalidad, rapidez y eficiencia a un sistema con licencias de paga, ejecutando las mismas órdenes con tiempos de respuesta similares, abaratando costos a las pequeñas y medianas industrias.

La obtención de los parámetros eléctricos (corriente, velocidad, frecuencia, temperatura), ayudan de forma rápida y eficiente a comprobar el estado y comportamiento de el o los equipos sometidos a estudio.

Mediante las redes utilizadas aseguramos que los datos obtenidos mediante el supervisor de red lleguen al dispositivo de control en forma fiable y rápida con la mínima pérdida de datos.

4.2. Operacionalización de las variables

4.2.1. Operacionalización conceptual

4.2.1.1. VARIABLE INDEPENDIENTE

Parámetros eléctricos obtenidos de la producción en tiempo real.

En la presente investigación los parámetros eléctricos (velocidad, corriente,) son entre los parámetros eléctricos más utilizados en procesos industriales, pero se debe recalcar que no son los únicos puestos el sistema implementado es abierto a cualquier parámetro.

4.2.1.2. VARIABLE DEPENDIENTE.

Interfaz HMI

Parámetros eléctricos (corriente, velocidad) tomados del desarrollo de una HMI basada en software libre.

4.2.2. Operacionalización metodológica

Tabla 4-1 Operacionalización metodológica de variables Variable Independiente

VARIABLES	DIMENSIONES	INDICADORES.
VARIABLE INDEPENDIENTE. Parámetros eléctricos obtenidos de la producción en tiempo real.	<ul style="list-style-type: none">• Corriente• Velocidad	<ul style="list-style-type: none">• Corriente• Velocidad

Realizado por: Fredy Romero, 2022

Tabla 4-2 Operacionalización metodológica de variables Variable Dependiente

VARIABLES	DIMENSIONES	INDICADORES.
VARIABLE DEPENDIENTE. Interfaz HMI	<ul style="list-style-type: none"> • Corriente • Velocidad 	<ul style="list-style-type: none"> • Corriente • Velocidad

Realizado por: Fredy Romero, 2022

4.3. Evaluación de Variables

Para la comprobación de la hipótesis vamos a realizar una comparación entre las variables dependientes e independientes, tomando en cuenta que lo que queremos comprobar es que los datos tomados en línea de producción sean semejantes a los generados en la Interfaz HMI, con el software libre.

A continuación, se muestra un estudio de las variables, Velocidad, Corriente, para comprobar una parte de nuestra hipótesis.

4.3.1. Velocidad (rpm)

La evaluación de la velocidad medida en rpm, utilizado en el presente trabajo, se realizó a partir de una comparación entre las velocidades tomadas al mismo tiempo con un tacómetro y el resultado de la programación en la interfaz, que recibe los datos del variador de frecuencia.

El tacómetro tiene las siguientes características:

Tabla 4-3 Características Tacómetro

Marca	TACHOMETER
Descripción	PCE-T236
Rango	0,5 ... 999,9 = 0,1 rpm < 99.999 = 1,0 rpm

Realizado por: Fredy Romero, 2022



Figura 4-1 Toma de muestra de velocidad

Realizado por: Fredy Romero, 2022

En la tabla 4-4, se muestra 20 medidas de velocidad tomadas cada 5 % de velocidad. Dichas medidas fueron tomadas tanto con el tacómetro, como con el resultado de la programación en la interfaz, que recibe los datos del variador de frecuencia.

Tabla 4-4 Mediciones de velocidad

N° ensayo	% Velocidad Máxima	Tacómetro (rpm)	Programación en la Interfaz. (rpm)
1	5	49,4	55
2	10	115	119
3	15	176	175
4	20	238,6	240
5	25	298,4	300
6	30	355,2	359
7	35	414,5	416
8	40	481	480
9	45	540,5	543
10	50	600	600
11	55	662	663
12	60	719	719
13	65	779,4	781
14	70	840	840
15	75	900,5	900
16	80	961	959
17	85	1019,3	1021
18	90	1082	1080
19	95	1140,2	1139
20	100	1199	1200

Realizado por: Fredy Romero, 2022

Realizando la curva del porcentaje de velocidad colocado en el sistema versus las medidas tomadas tanto con el instrumento como con el sistema nos damos cuenta de que es linealmente creciente, a más porcentaje de velocidad mayor revoluciones del motor.

Además, se puede notar la mínima desviación estándar de los datos, en ambas tomas.

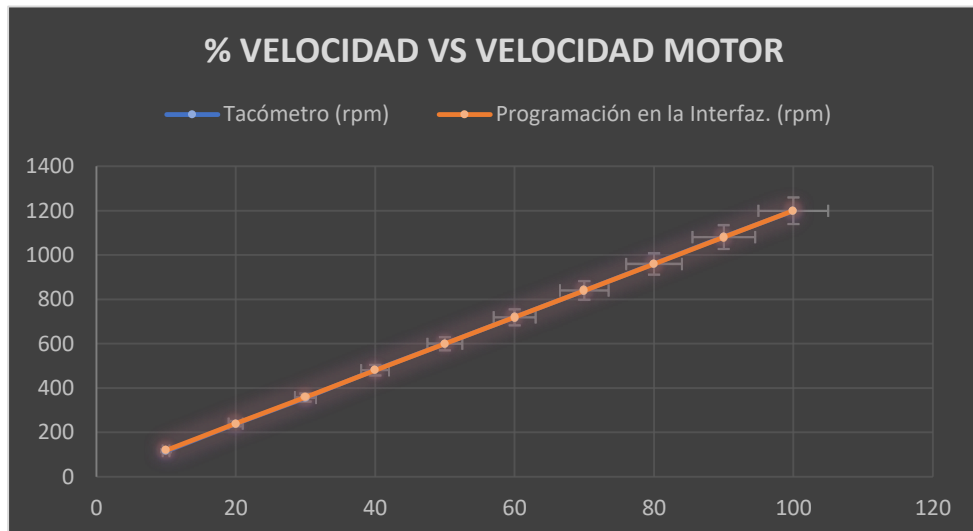


Figura 4-2 % VELOCIDAS VS VELOCIDAD MOTOR

Realizado por: Fredy Romero, 2022

4.3.1.1. Prueba de normalidad para la variable *velocidad (rpm)* con el *tacómetro*

Se desea determinar si los datos nacen de una distribución normal, para identificar después si existe alguna variación con la variable velocidad (rpm) con la Interfaz, para lo cual seguimos los siguientes pasos:

1. Planteamos hipótesis.

H0: Los datos se ajustan a una distribución normal

H1: Los datos NO se ajustan a una distribución normal

2. Nivel de significancia $\alpha = 0.05$

3. Utilizando el SOFTWARE R encontramos:

```
> shapiro.test(TACOMETRO)

      Shapiro-Wilk normality test

data:  TACOMETRO
W = 0.96089, p-value = 0.5618
```

4. $W_c = 0.905$

5. Tomamos una decisión:

Como $W > W_c$, no rechazamos la hipótesis nula y concluimos que no existe evidencia suficiente para decir bajo el nivel de significancia de 0.05 que los datos no se ajustan a una distribución normal.

Por lo tanto, los datos de la variable *velocidad (rpm) con el tacómetro* provienen de una distribución normal.

4.3.1.2. Prueba de normalidad para la variable velocidad (rpm) con la Interfaz

Se desea determinar si los datos nacen de una distribución normal, para identificar después si existe alguna variación con la variable velocidad (rpm) con el Tacómetro, para lo cual seguimos los siguientes pasos:

1. Planteamos hipótesis.

H0: Los datos se ajustan a una distribución normal

H1: Los datos NO se ajustan a una distribución normal

2. Nivel de significancia $\alpha = 0.05$

3. Utilizando el SOFTWARE R encontramos:

```
> shapiro.test(INFERFAZ)

      Shapiro-Wilk normality test

data:  INFERFAZ
W = 0.96069, p-value = 0.5578
```

4. $W_c = 0.905$

5. Tomamos una decisión:

Como $W > W_c$, no rechazamos la hipótesis nula y concluimos que no existe evidencia suficiente para decir bajo el nivel de significancia de 0.05 que los datos no se ajustan a una distribución normal.

Por lo tanto, los datos de la variable *velocidad (rpm) con la Interfaz* provienen de una distribución normal.

4.3.1.3. Prueba de diferencia de medias

Para el análisis estadístico se aplicará el contraste de medias con la prueba t-student y se utilizará la distribución t-student porque los datos a analizar son menores de 30.

1.

$H_0: \mu_1 - \mu_2 = 0$ (Medias Iguales) El parámetro velocidad medido en el Sistema de control y monitoreo realizado a través del software libre es igual en funcionalidad, rapidez y eficiencia a el parámetro velocidad medido por un instrumento utilizado en las pequeñas y medianas industrias.

$H_1: \mu_1 - \mu_2 \neq 0$ (Medias distintas) El parámetro velocidad medido en el Sistema de control y monitoreo realizado a través del software libre no es igual en funcionalidad, rapidez y eficiencia a el parámetro velocidad medido por un instrumento utilizado en las pequeñas y medianas industrias.

2. $\alpha = 0.05$

3.

Región Crítica: Para definir la región crítica, calculamos los grados de libertad:

Grados de libertad = $(n_1 + n_2 - 1) = (20 + 20 - 1) = 39$

n_1 = Número de datos del primer grupo

n_2 = Número de datos del segundo grupo

$$t_{(0,05; 39)} >= 2,02$$

4. Utilizando el SOFTWARE R encontramos

```

> t.test(TACOMETRO, INFERFAZ)

Welch Two Sample t-test

data: TACOMETRO and INFERFAZ
t = -0.0079837, df = 38, p-value = 0.9937
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -229.1091 227.3091
sample estimates:
mean of x mean of y
 628.55 629.45

```

5. $T_c = 1.64$
 $-T_c = -1.64$
6. Como $p = 0.9937$ es mayor que el nivel de significancia $\alpha = 0.05$ (o también $-T_c < T < T_c$) aceptamos la hipótesis nula y concluimos que no existe evidencia suficiente para decir bajo el nivel de significancia planteado que las medias poblacionales son distintas.

Es decir, El parámetro velocidad medido en el Sistema de control y monitoreo realizado a través del software libre es igual en funcionalidad, rapidez y eficiencia a el parámetro velocidad medido por un instrumento utilizado en las pequeñas y medianas industrias, ayudando de forma rápida y eficiente a comprobar el estado y comportamiento de el o los equipos de cualquier proceso, con la mínima pérdida de datos en forma fiable y rápida.

4.3.2. Corriente (Amperios)

La evaluación de la corriente medida en amperios, utilizado en el presente trabajo, se realizó a partir de una comparación entre las corrientes tomadas al mismo tiempo con una Pinza Amperimétrica y el resultado de la programación en la interfaz, que recibe los datos del variador de frecuencia.

La pinza amperimétrica tiene las siguientes características:

Tabla 4-5 Características Pinza amperimétrica

Marca	FLUKE323
Descripción	Pinza amperimétrica de 400A AC con un diámetro máximo de 30mm; RMS AC con precisión del 2%, medida de voltaje máximo 600V AC
Rango	Medida de corriente CA de 400 A Medida de tensión de CA y CC de 600 V

Realizado por: Fredy Romero, 2022



Figura 4-3 Toma de muestra de corriente

Realizado por: Fredy Romero, 2022

En la tabla 4-6 se muestra 20 medidas de corriente tomadas cada 5 % de velocidad. Dichas medidas fueron tomadas tanto con la pinza amperimétrica, como con el resultado de la programación en la interfaz, que recibe los datos del variador de frecuencia.

Tabla 4-6 Mediciones de corriente

Nº ensayo	% Velocidad Máxima	Pinza amperimétrica (amperios)	Programación en la Interfaz. (rpm)
1	5	1,6	1,7
2	10	1,7	1,6
3	15	2,1	1,8
4	20	2,1	2
5	25	2,2	2
6	30	2,1	2
7	35	2,1	2,1
8	40	2,2	2
9	45	2	2,1
10	50	2,1	2
11	55	2,2	2,1
12	60	2,1	2
13	65	2,1	2
14	70	2,1	2
15	75	2,1	2
16	80	2	2,1
17	85	2	2,2
18	90	2	2,2
19	95	2,1	2,1
20	100	2	2,1

Realizado por: Fredy Romero, 2022

Realizando la curva del porcentaje de velocidad colocado en el sistema versus las medidas tomadas tanto con el instrumento como con el sistema nos damos cuenta de que es la corriente se mantiene constante durante todo el proceso.

Además, se puede notar la mínima desviación estándar de los datos, en ambas tomas.

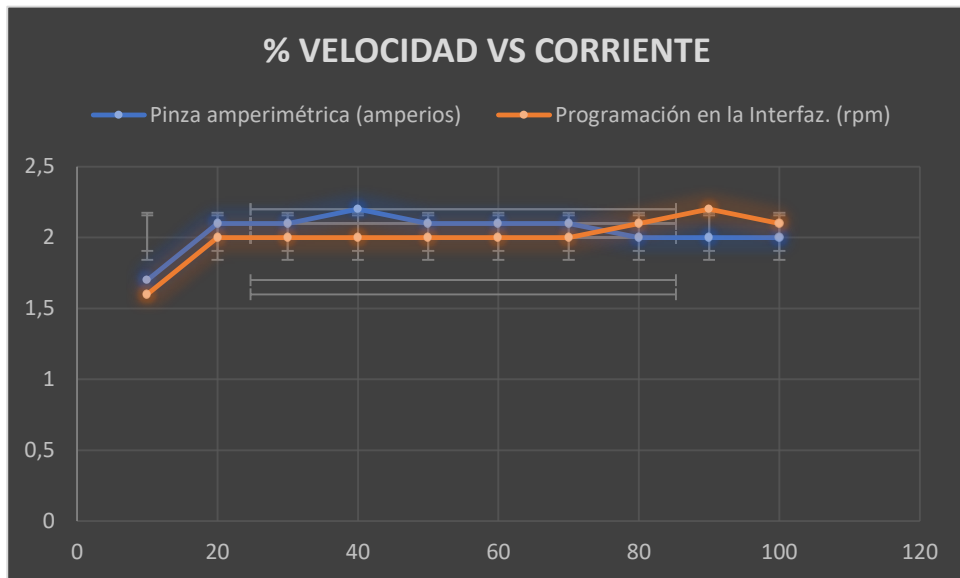


Figura 4-4 % VELOCIDAD VS CORRIENTE

Realizado por: Fredy Romero, 2022

4.3.2.1. Prueba de normalidad para la variable corriente (amperios) con la pinza amperimétrica

Se desea determinar si los datos nacen de una distribución normal, para identificar después si existe alguna variación con la variable corriente (amperios) con la Interfaz, para lo cual seguimos los siguientes pasos:

1. Planteamos hipótesis.
 H0: Los datos se ajustan a una distribución normal
 H1: Los datos NO se ajustan a una distribución normal
2. Nivel de significancia $\alpha = 0.05$
3. Utilizando el SOFTWARE R encontramos:

```
> shapiro.test (AMPERIMETRO)

      Shapiro-Wilk normality test

data:  AMPERIMETRO
W = 0.72293, p-value = 0.00007524
```

4. $W_c = 0.905$

5. Tomamos una decisión:

Como $W < W_c$, rechazamos la hipótesis nula y concluimos que existe evidencia suficiente para decir bajo el nivel de significancia de 0.05 que los datos no se ajustan a una distribución normal.

Por lo tanto, los datos de la variable **corriente (amperios) con la pinza amperimétrica** no provienen de una distribución normal.

4.3.2.2. Prueba de normalidad para la variable corriente (amperios) con la Interfaz

Se desea determinar si los datos nacen de una distribución normal, para identificar después si existe alguna variación con la variable corriente (amperios) con la pinza amperimétrica, para lo cual seguimos los siguientes pasos:

1. Planteamos hipótesis.

H0: Los datos se ajustan a una distribución normal

H1: Los datos NO se ajustan a una distribución normal

2. Nivel de significancia $\alpha = 0.05$

3. Utilizando el SOFTWARE R encontramos:

```
> shapiro.test (INTRFAZAMPER)

      Shapiro-Wilk normality test

data:  INTRFAZAMPER
W = 0.81026, p-value = 0.001234
```

4. $W_c = 0.905$

5. Tomamos una decisión:

Como $W < W_c$, rechazamos la hipótesis nula y concluimos que existe evidencia suficiente para decir bajo el nivel de significancia de 0.05 que los datos no se ajustan a una distribución normal.

Por lo tanto, los datos de la variable *corriente (amperios) con la Interfaz no* provienen de una distribución normal.

4.3.2.3. Prueba de Wilcoxon (Prueba de medianas)

Al no venir estos datos de una distribución normal, no podemos utilizar una prueba paramétrica, por lo que utilizaremos una prueba no paramétrica para probar la igualdad de medianas.

1.

$H_0: \tilde{\mu}_1 = \tilde{\mu}_2$ (Medianas Iguales) El parámetro corriente medido en el sistema de control y monitoreo realizado a través del software libre es igual en funcionalidad, rapidez y eficiencia a el parámetro corriente medido por un instrumento utilizado en las pequeñas y medianas industrias.

$H_1: \tilde{\mu}_1 > \tilde{\mu}_2$ (Medianas distintas) El parámetro corriente medido en el sistema de control y monitoreo realizado a través del software libre no es igual en funcionalidad, rapidez y eficiencia a el parámetro corriente medido por un instrumento utilizado en las pequeñas y medianas industrias.

2. $n_1 = 20$

$n_2 = 20$

3. Ordenamos de menor a mayor las observaciones de las 2 muestras mezcladas.

4. Asignamos un rango de 1, 2,... a cada observación ordenada. Si la observación se repite, el rango asignado será la media de los rangos que debieron corresponder a cada uno.

5.

Utilizando el SOFTWARE R encontramos


```
> wilcox.test(x = AMPERIMETRO, y = INTRFAZAMPER, alternative = "greater")  
  
Wilcoxon rank sum test with continuity correction  
  
data: AMPERIMETRO and INTRFAZAMPER  
W = 246.5, p-value = 0.09377  
alternative hypothesis: true location shift is greater than 0
```

6. El valor crítico

$u_c = 261$

7. Finalmente tomamos una decisión:

Para una prueba a cola derecha, como $u_B = 246.6$ es menor que el valor crítico $u_c = 261$ (o también $p > \alpha$), entonces apruebo H_0 , y concluyo que existe evidencia suficiente para decir que las medianas son iguales.

Es decir, El parámetro corriente medido en el Sistema de control y monitoreo realizado a través del software libre es igual en funcionalidad, rapidez y eficiencia a el parámetro corriente medido por un instrumento utilizado en las pequeñas y medianas industrias, ayudando de forma rápida y eficiente a comprobar el estado y comportamiento de el o los equipos de cualquier proceso, con la mínima pérdida de datos en forma fiable y rápida.

4.4. Comparación de costos

El desarrollo del sistema realizado en software libre, para el control y monitoreo de variables, se lo realizo para abaratar costos pensando en las necesidades pequeñas y medianas industrias, para las cuales comprar un software que efectúe los mismos procesos, puede llegar a ser costoso, para lo cual se ha realizado una tabla determinando precios referenciales del Sistema SCADA de la marca SIEMENS, dependiendo del número de Power Tags, y la propuesta desarrollada en esta investigación.

Tabla 4-7 Precios referenciales SCADA

DESCRIPCIÓN	N° POWER TAGS	SIEMENS ECUADOR
TIA PORTAL WinCC V17 Profesional Runtime (versión descargable). Software SCADA para alta funcionalidad. Incluye licencia Microsoft SQL Server para la gestión de datos. Licencia para 128 Power Tags. Incluye licencia WinCC WebUx para monitorear el proceso desde un dispositivo Smart (celular, tablets, smart tv, etc.).	128	5.134,00
TIA PORTAL WinCC V17 Profesional Runtime. Software SCADA para alta funcionalidad. Incluye licencia Microsoft SQL Server para la gestión de datos. Licencia para 128 Power Tags. Incluye licencia WinCC WebUx para monitorear el proceso desde un dispositivo Smart (celular, tablets, smart tv, etc.).	128	5.937,00
TIA PORTAL WinCC V17 Profesional (versión descargable). Software de ingeniería para la configuración Sistemas SCADA de alta funcionalidad. Licencia para 51 Power Tags. Para aplicaciones con mayor número de Power Tags favor consulta	51	4.823,00
Software de ingeniería para la configuración Sistemas SCADA de alta funcionalidad. Licencia para < 51 Power Tags.	< 51	4.823,00

Realizado por: Fredy Romero, 2022

Tabla 4-8 Precios referenciales HMI DE LA INVESTIGACIÓN

DESCRIPCIÓN	N° POWER TAGS	SIEMENS ECUADOR
VERSIÓN PROGRAMADA DE 128 TAGS	128	1.000,00
VERSIÓN PROGRAMADA DE 120 TAGS	120	1.000,00
VERSIÓN PROGRAMADA DE 51 TAGS	51	800,00
VERSIÓN PROGRAMADA < 51 TAGS	< 51	500,00

Realizado por: Fredy Romero, 2022

4.4.1. Prueba de normalidad para los costos Scada Siemens

Se desea determinar si los datos nacen de una distribución normal, para identificar después si existe alguna variación con los costos HMI de la investigación, para lo cual seguimos los siguientes pasos:

1. Planteamos hipótesis.

H0: Los datos se ajustan a una distribución normal

H1: Los datos NO se ajustan a una distribución normal

2. Nivel de significancia $\alpha = 0.05$

3. Utilizando el SOFTWARE R encontramos:

```
> SCADA= c(5.134, 5.937, 4.823, 4.823)
> shapiro.test(SCADA)

      Shapiro-Wilk normality test

data:  SCADA
W = 0.80482, p-value = 0.1111
```

4. $W_c = 0.748$

5. Tomamos una decisión:

Como $W > W_c$, no rechazamos la hipótesis nula y concluimos que no existe evidencia suficiente para decir bajo el nivel de significancia de 0.05 que los datos no se ajustan a una distribución normal.

Por lo tanto, los datos de *costos Scada Siemens* provienen de una distribución normal.

4.4.2. Prueba de normalidad para los costos HMI de la investigación

Se desea determinar si los datos nacen de una distribución normal, para identificar después si existe alguna variación con *los costos Scada Siemens*, para lo cual seguimos los siguientes pasos:

1. Planteamos hipótesis.

H0: Los datos se ajustan a una distribución normal

H1: Los datos NO se ajustan a una distribución normal

2. Nivel de significancia $\alpha = 0.05$
3. Utilizando el SOFTWARE R encontramos:

```
> HMI= c(1000, 1000, 800, 500)
> shapiro.test(HMI)

      Shapiro-Wilk normality test

data:  HMI
W = 0.84808, p-value = 0.22
```

4. $W_c = 0.748$
5. Tomamos una decisión:

Como $W > W_c$, no rechazamos la hipótesis nula y concluimos que no existe evidencia suficiente para decir bajo el nivel de significancia de 0.05 que los datos no se ajustan a una distribución normal.

Por lo tanto, los datos de la variable costos HMI de la investigación provienen de una distribución normal.

4.4.3. Prueba de diferencia de medias

Para el análisis estadístico se aplicará el contraste de medias con la prueba t-student y se utilizara la distribución t-student porque los datos a analizar son menores de 30.

- 1.

$H_0: \mu_1 - \mu_2 = 0$ (Medias Iguales) El costo del desarrollo e implementación de un HMI basado en software libre es igual al costo de un software adquirido mediante una licencia de paga.

$H_1: \mu_1 - \mu_2 \neq 0$ (Medias distintas) El costo del desarrollo e implementación de un HMI basado en software libre no es igual al costo de un software adquirido mediante una licencia de paga.

2. $\alpha = 0.05$

3.

Región Crítica: Para definir la región crítica, calculamos los grados de libertad:

Grados de libertad = $(n_1 + n_2 - 1) = (4 + 4 - 1) = 7$

n_1 = Número de datos del primer grupo

n_2 = Número de datos del segundo grupo

$$t_{(0.05; 7)} \geq 1,89$$

4. Utilizando el SOFTWARE R encontramos

```
> t.test(SCADA,HMI)

Welch Two Sample t-test

data: SCADA and HMI
t = -6.9391, df = 3, p-value = 0.006138
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1195.8109 -443.8306
sample estimates:
mean of x mean of y
 5.17925 825.00000
```

5. Como $p = 0.006138$ es menor que el nivel de significancia $\alpha = 0.05$ rechazamos la hipótesis nula y concluimos que existe evidencia suficiente para decir bajo el nivel de significancia planteado que las **medias poblacionales son distintas**.

Es decir, El costo del desarrollo e implementación de un HMI basado en software libre no es igual al costo de un software adquirido mediante una licencia de paga, siendo notable que el costo de esta propuesta de investigación es menor a la de marcas reconocidas en el ámbito de la automatización, contribuyendo a que pequeñas y medianas empresas tengan acceso a tecnología de calidad a un precio accesible, incrementando así su capacidad de competitividad en el mercado, equiparando en funcionalidad, rapidez y eficiencia.

4.5. Análisis de datos almacenados

Los datos almacenados e importados en el Excel nos dan una serie de ventajas en cuestión al funcionamiento de los procesos, los cuales mediante gráficas dan paso al análisis del comportamiento de cada parámetro, y posibles conclusiones de los procesos.

A continuación, se mostrará un ejemplo de tabla obtenida de los diferentes parámetros en un determinado tiempo donde observaremos las gráficas de funcionamiento.

Tabla 4-9 Almacenamiento de datos

Time	PA.	PAC.	P.R	F.P	VOL.	CO.	VEL.	TOR.
31/3/2022 10:48	6	0	3	1	23	184	4	1,84
31/3/2022 10:48	15	-3	3	22	29	209	4	2,09
31/3/2022 10:48	51	-4	5	7	44	255	-4	2,55
31/3/2022 10:48	92	19	2	21	62	239	-39	2,39
31/3/2022 10:48	157	58	-11	37	84	216	-16	2,16
31/3/2022 10:48	223	84	-8	38	104	209	-9	2,09
31/3/2022 10:48	360	122	65	34	124	204	-5	2,04
31/3/2022 10:48	456	138	164	30	136	200	-4	2
31/3/2022 10:48	488	145	261	30	136	200	-4	2
31/3/2022 10:48	487	144	284	30	136	200	-4	2
31/3/2022 10:48	488	145	283	30	136	200	-4	2
31/3/2022 10:48	487	145	283	30	136	200	-4	2
31/3/2022 10:48	488	145	283	30	136	200	-4	2
31/3/2022 10:48	491	146	287	30	136	200	-4	2
31/3/2022 10:48	488	144	287	30	136	200	-4	2
31/3/2022 10:48	488	144	284	30	136	200	-4	2
31/3/2022 10:48	493	146	289	30	136	200	-4	2

Realizado por: Fredy Romero, 2022

Con estos datos se puede realizar gráficas de comportamiento de los diferentes parámetros en el proceso, los cuales con su respectivo análisis ayudaran al usuario a verificar el comportamiento de su equipo.

En la gráfica podemos observar el comportamiento del torque en un instante de tiempo, cabe recalcar que los datos están obtenidos en un instante de tiempo dentro de un segundo.

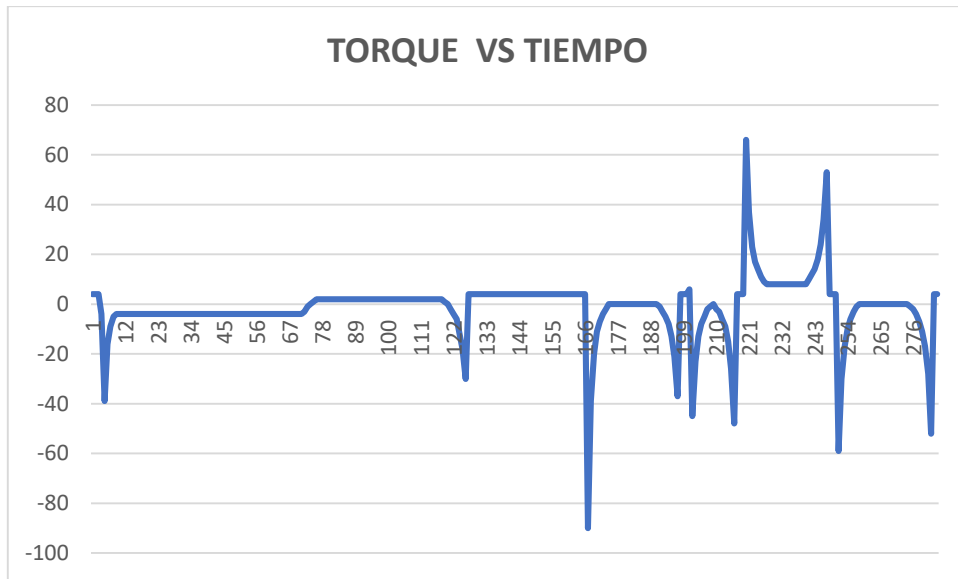


Figura 4-5 Gráfica de torque en el tiempo

Realizado por: Fredy Romero, 2022

En la gráfica podemos observar el comportamiento de la corriente en un instante de tiempo, cabe recalcar que los datos están obtenidos en un instante de tiempo dentro de un segundo.

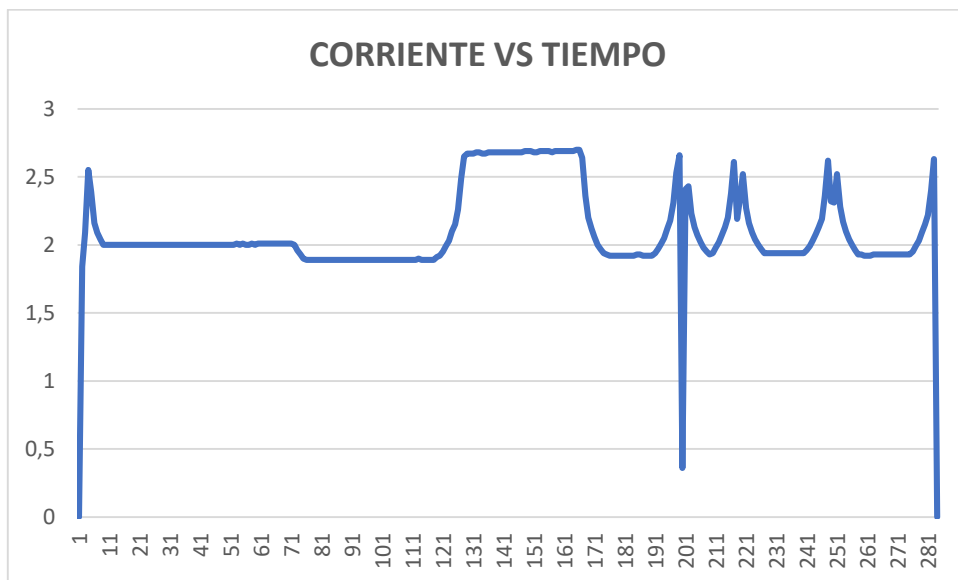


Figura 4-6 Gráfica de corriente en el tiempo

Realizado por: Fredy Romero, 2022

CAPÍTULO V

5. ANÁLISIS Y RESULTADOS

5.1. Desarrollo de la interfaz gráfica

Para el desarrollo de esta interfaz se utilizó el lenguaje de programación Python, que, junto con las librerías existentes, nos ayuda a desarrollar aplicaciones, programas, códigos y demás instrumentos que se utilizan en el ámbito empresarial y educativo, este se enlaza con muchos otros softwares que permiten aplicar en muchos más procesos dependiendo en donde se lo esté utilizando.

Para establecer la comunicación se utilizó las funciones propias del software Snap7.

5.1.1.Descripción del SNAP 7

Este software nos permite comunicarnos con el PLC, empleando tecnología ethernet y protocolo S7, ya que su arquitectura diseñada para 32 y 64 bits permite que los sistemas operativos funcionen óptimamente en las computadoras.

Basado en el protocolo S7, de la marca Siemens, soporta considerablemente el protocolo TCP, ya que este permite el envío de datos en la automatización de procesos y es indispensable orientar esta información.

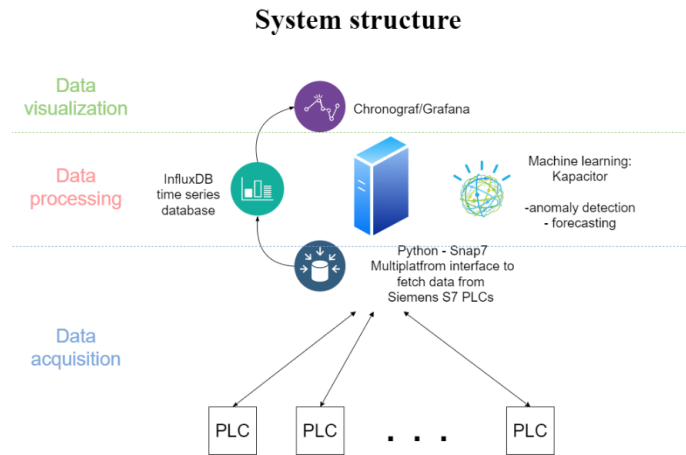


Figura 5-1 Snap7

Fuente: (GitHub - ARMAGEDONgtx/IoT-System-PLC-Data-to-InfluxDB; El Objetivo de Este Proyecto Es Proporcionar Software Gratuito Para Obtener Datos de Plcs (Siemens S7-300/400/1200/1500) y Almacenarlos. La Pila Usada Es Completamente de Código Abierto. Usé InfluxDB Como Almacenamiento de Datos, Por Lo Que El Principio de La Aplicación Sigue El Paradigma Big Data., n.d.)

5.1.2. Botón START

Para el desarrollo de los objetos gráficos propios de la interfaz se utilizó el módulo tkinter el cual me permite la puesta en marcha del motor ya sea en sentido horario o antihorario según la selección que se haya dispuesto, este botón está implementado bajo el siguiente código.

```

start_btn = Button(root, text="START", height=2, width=10, bg = bgColorButton,
fg=fgColorColor, command=lambda: btnClicked(DB_NUMBER,0,1,0,0,plc))
start_btn.place( x = 150, y = 80, width = 130, height = 55)
start_btn.configure(font=font_port_bd)
  
```

En la variable start_btn se almacena el objeto Button que tiene por parámetros: root este parámetro hace referencia a la pertenencia del objeto, text = "START" mostrará el texto en el botón, height indica la altura del botón, bg configura el color de fondo del botón, fg configura el color del texto que aparece en el botón, command es la acción que realiza el botón cuando es presionado.

El código descrito genera el siguiente objeto en la interfaz gráfica.

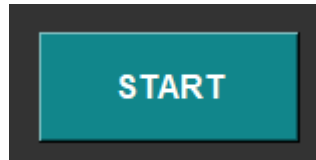


Figura 5-2 Botón de inicio

Realizado por: Fredy Romero, 2022

5.1.3. Botón Stop

Este botón me permite realizar el paro del motor en cualquier instante del proceso industrial, este botón está implementado bajo el siguiente código.

```
stop_btn = Button(root, text="STOP", height=2, width=10, bg = bgColorButton,
fg=fgColorColor, command= lambda: btnClicked(DB_NUMBER,0,1,0,1,plc))
stop_btn.place(
x = 150, y = 155, width = 130, height = 55)
stop_btn.configure(font=font_port_bd)
```

En la variable `start_btn` se almacena el objeto `Button` que tiene por parámetros: `root` este parámetro hace referencia a la pertenencia del objeto, `text = "STOP"` mostrará el texto en el botón, `height` indica la altura del botón, `bg` configura el color de fondo del botón, `fg` configura el color del texto que aparece en el botón, `command` es la acción que realiza el botón cuando es presionado.

El código descrito genera el siguiente objeto en la interfaz gráfica.

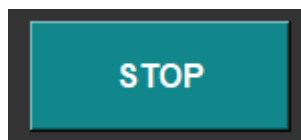


Figura 5-3 Botón de pare

Realizado por: Fredy Romero, 2022

5.1.4. Botones de giro del motor

Estos botones me permiten dar el giro necesario al motor dependiendo del proceso en el que esté funcionando, bajando la velocidad a cero para luego iniciar el giro hasta llegar a la frecuencia determinada con un arranque lento del motor, implementados bajo el siguiente código.

```
inv_h_btn = Button(root, text="HORARIO", height=2, width=10, bg = bgColorButton,
fg=fgColorColor, command=lambda: btnClicked(DB_NUMBER,0,1,0,0,plc))
```

```
inv_h_btn.place( x = 70, y = 230, width = 130, height = 55)
```

```
inv_h_btn.configure(font=font_port_bd)
```

```
inv_a_btn = Button(root, text="ANTIHORARIO", height=2, width=10, bg =
bgColorButton, fg=fgColorColor, command=lambda:
btnClicked(DB_NUMBER,0,1,0,2,plc))
```

```
inv_a_btn.place( x = 225, y = 230, width = 130, height = 55)
```

```
inv_a_btn.configure(font=font_port_bd)
```

En la variable `inv_h_btn` en ambos casos se almacena el objeto `Button` que tiene por parámetros: `root` este parámetro hace referencia a la pertenencia del objeto, `text = "HORARIO"` y `"ANTIHORARIO"` respectivamente, que mostrará el texto en el botón, `height` indica la altura del botón, `bg` configura el color de fondo del botón, `fg` configura el color del texto que aparece en el botón, `command` es la acción que realiza el botón cuando es presionado.

El código descrito genera el siguiente objeto en la interfaz gráfica.

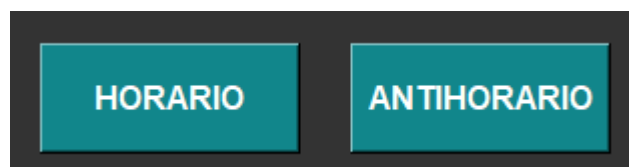


Figura 5-4 Botones de giro del motor

Realizado por: Fredy Romero, 2022

5.1.5.Velocidad

El botón frecuencia me permite cambiar la velocidad del motor en cualquier instante del proceso que va en un rango del cero al cien por ciento, la información requerida la colocamos dentro de un label, interpretados bajo el siguiente código.

```
freq_lable = Label(root, text="VELOCIDAD(0-100 %): ", bg=bgColorScreen,
fg=fgColorColor)
freq_lable.place( x = 20, y = 300, width = 280, height = 55)
freq_lable.configure(font=font_port_lb)
editFrecuencia = Entry(root,justify = 'center', bd =5)
editFrecuencia.place( x = 70, y = 375, width = 130, height = 53)
editFrecuencia.configure(font=font_port_bd)
enviar_btn = Button(root, text="FRECUENCIA", height=2, width=10, bg =
bgColorButton, fg=fgColorColor, command=freq_changed)
enviar_btn.place( x = 225, y = 375, width = 130, height = 55)
enviar_btn.configure(font=font_port_bd)
```

Dentro de la variable freq_lable se demuestra el porcentaje con el que podemos controlar el motor y su ubicación dentro de la interfaz, seguido de editFrecuencia como un comando que luego nos permitirá cambiar el dato ingresado. El código descrito genera el siguiente ojeo en la interfaz gráfica.



Figura 5-5 Velocidad del motor

Realizado por: Fredy Romero, 2022

5.1.6.Botón de conexión

Este botón me permite establecer la comunicación del sistema, cambiando el color de fondo el momento que existe algún inconveniente al establecer la conexión dado por el siguiente código.

```
connect_btn = Button(root, text="CONNECT", height=2, width=10, state="disabled", bg
= bgColorButton, fg=fgColorColor, command=connection_select)
connect_btn.place(x = 150, y = 480, width = 130, height = 55)
connect_btn.configure(font=font_port_bd)
```

El código descrito con connect_btn almacena la información de comunicación existente seguido de su ubicación en la interfaz generando el siguiente objeto.



Figura 5-6 Botón de conexión

Realizado por: Fredy Romero, 2022

5.1.7. Indicador de temperatura

El siguiente objeto de la interfaz gráfica permite visualizar el valor de la temperatura como valor nominal, así como también de forma gráfica, viene dado por el siguiente código utilizando la librería tkinter.

```
gaugeVolt_lable = Label(root, text="TEMPERATURA", bg=bgColorScreen,
fg=fgColorColor)
gaugeVolt_lable.place( x = 625, y = 60, width = 200, height = 55)
gaugeVolt_lable.configure(font=font_port_lb)
graph.canvas = Canvas(root, width=200, height=200, bg=bgColorScreen,
highlightthickness=0)
graph.canvas.place( x = 600, y = 100, width = 200, height = 200,
graph.outer = graph.canvas.create_arc(
10, 10, 190, 190, start=90, extent=100, outline= bgColorGaugeV,
fill=bgColorGaugeV, width=2)
graph.canvas.create_oval(
75, 75, 125, 125, outline= bgColorGaugeV, fill="white", width=2)
graph.text = graph.canvas.create_text(
95, 100, anchor=E, font=("Helvetica", "12"), text="---")
```

```
graph.canvas.create_text(
    110, 100, anchor=CENTER, font=("Helvetica", "12"), text="°C")
graph1 = Graphics()a librería tkinter,
```

En el código descrito tenemos la variable `gaugeVolt_lable` donde se almacena información como el nombre, la ubicación del objeto en la interfaz y las etiquetas donde se indica la información requerida del proceso en el que se esté trabajando, el objeto se muestra de la siguiente manera en la interfaz gráfica.

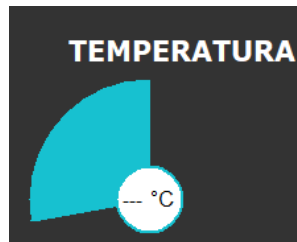


Figura 5-7 Indicador de temperatura

Realizado por: Fredy Romero, 2022

5.1.8. Indicador de Velocidad

El siguiente objeto de la interfaz gráfica permite visualizar el valor de la velocidad como valor nominal, así como también de forma gráfica, viene dado por el siguiente código utilizando la librería tkinter.

```
gaugeVel_lable = Label(root, text="VELOCIDAD", bg=bgColorScreen,
fg=fgColorColor)
gaugeVel_lable.place( x = 625, y = 360, width = 150, height = 55)
gaugeVel_lable.configure(font=font_port_lb)
graph2.canvas = Canvas(root, width=200, height=200, bg=bgColorScreen,
highlightthickness=0)
graph2.canvas.place( x = 600, y = 400, width = 200, height = 200)
graph2.outer = graph2.canvas.create_arc(
10, 10, 190, 190, start=90, extent=100, outline=bgColorGaugeS,
fill=bgColorGaugeS, width=2)
graph2.canvas.create_oval(
75, 75, 125, 125, outline=bgColorGaugeS, fill="white", width=2)
graph2.text = graph2.canvas.create_text(
```

```

95, 100, anchor=E, font=("Helvetica", "10"), text="---")
graph2.canvas.create_text( 110, 100, anchor=CENTER, font=("Helvetica", "10"),
text=" RPM")

```

En el código descrito tenemos la variable `gaugeVel_lable` donde se almacena información como el nombre, la ubicación del objeto en la interfaz y las etiquetas donde se indica la información requerida del proceso en el que se esté trabajando, el objeto se muestra de la siguiente manera en la interfaz gráfica.

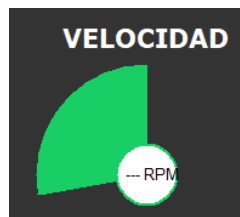


Figura 5-8 Indicador de velocidad

Realizado por: Fredy Romero, 2022

5.1.9. Indicador de corriente

El siguiente objeto de la interfaz gráfica permite visualizar el valor de la corriente como valor nominal, así como también de forma gráfica, viene dado por el siguiente código utilizando la librería tkinter.

```

gaugeCurr_lable = Label(root, text="CORRIENTE", bg=bgColorScreen,
fg=fgColorColor)
gaugeCurr_lable.place(x = 925, y = 60, width = 150, height = 55)
gaugeCurr_lable.configure(font=font_port_lb)
graph1.canvas = Canvas(root, width=200, height=200, bg=bgColorScreen,
highlightthickness=0)
graph1.canvas.place( x = 900, y = 100, width = 200, height = 200)
graph1.outer = graph1.canvas.create_arc(
10, 10, 190, 190, start=90, extent=100, outline=bgColorGaugeI,
fill=bgColorGaugeI, width=2)
graph1.canvas.create_oval(
75, 75, 125, 125, outline=bgColorGaugeI, fill="white", width=2)
graph1.text = graph1.canvas.create_text(

```

```

95, 100, anchor=E, font=("Helvetica", "12"), text="---")
graph1.canvas.create_text(
110, 100, anchor=CENTER, font=("Helvetica", "12"), text="A")

```

En el código descrito tenemos la variable `gaugeCurr_lable` donde se almacena información como el nombre, la ubicación del objeto en la interfaz y las etiquetas donde se indica la información requerida del proceso en el que se esté trabajando, el objeto se muestra de la siguiente manera en la interfaz gráfica.

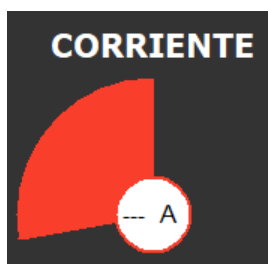


Figura 5-9 Indicador de corriente

Realizado por: Fredy Romero, 2022

5.1.10. *Indicador de la frecuencia*

El siguiente objeto de la interfaz gráfica permite visualizar el valor de la frecuencia como valor nominal, así como también de forma gráfica, viene dado por el siguiente código utilizando la librería `tkinter`.

```

gaugeFeq_lable = Label(root, text="FRECUENCIA", bg=bgColorScreen,
fg=fgColorColor)
gaugeFeq_lable.place(x = 925, y = 360,width = 150,height = 55)
gaugeFeq_lable.configure(font=font_port_lb)
graph3.canvas = Canvas(root, width=200, height=200, bg=bgColorScreen,
highlightthickness=0)
graph3.canvas.place( x = 900, y = 400, width = 200, height = 200)
graph3.outer = graph3.canvas.create_arc(
10, 10, 190, 190, start=90, extent=100, outline=bgColorGaugeF,
fill=bgColorGaugeF, width=2)

```



```

graph3.canvas.create_oval(
    75, 75, 125, 125, outline=bgColorGaugeF, fill="white", width=2)
graph3.text = graph3.canvas.create_text(
    95, 100, anchor=E, font=("Helvetica", "12"), text="---")
graph3.canvas.create_text(
    110, 100, anchor=CENTER, font=("Helvetica", "12"), text="Hz")

```

En el código descrito tenemos la variable `gaugeFeq_lable` donde se almacena información como el nombre, la ubicación del objeto en la interfaz y las etiquetas donde se indica la información requerida del proceso en el que se esté trabajando, el objeto se muestra de la siguiente manera en la interfaz gráfica.

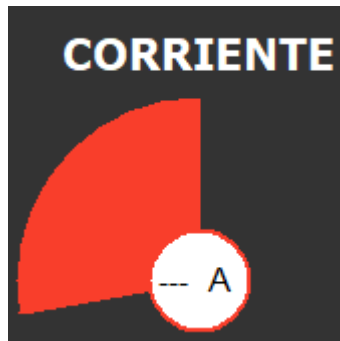


Figura 5-10 Indicador de frecuencia

Realizado por: Fredy Romero, 2022

5.2. Almacenamiento de datos

Es indispensable tener un registro de almacenamiento de datos en tiempo real, de los diferentes parámetros eléctricos, para lo cual se ha establecido un diagrama con las diferentes entradas de los parámetros a almacenar, el dispositivo lee según los tiempos establecidos por el usuario y va almacenando cada uno de los datos indicando la fecha y hora de adquisición de este.

EL proceso se lo realiza desde la puesta en marcha hasta el momento en que se da stop es decir se detiene el proceso.

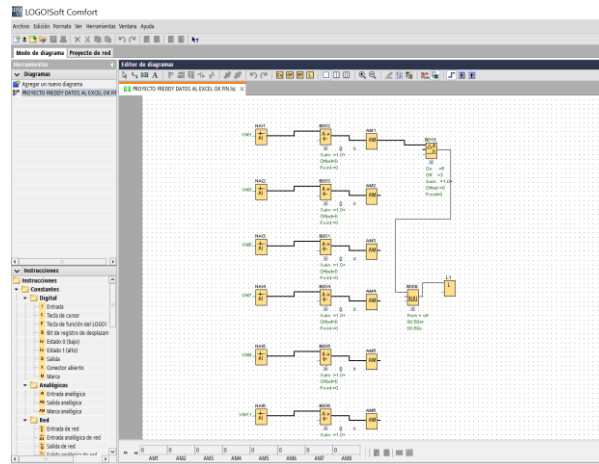


Figura 5-11 Diagrama de almacenamiento de datos

Realizado por: Fredy Romero, 2022

Los datos obtenidos se importan a un archivo de Excel, en el que podemos procesar la información de cada uno de los parámetros eléctricos realizados en esta investigación.

Estos datos son fundamentales para que el usuario pueda mantener un registro necesario para programar mantenimientos, o visualizar en funcionamiento adecuado de sus procesos.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Time	AM1	AM2	AM3	AM4	AM5	AM6	AM7	AM8					
2	25/3/2020 8:00	7	-1	6	11	27	49	0	4					
3	25/3/2020 8:00	48	-6	7	13	43	157	223	4					
4	25/3/2020 8:00	88	17	4	19	59	280	242	-45					
5	25/3/2020 8:00	146	55	-12	37	80	403	221	-21					
6	25/3/2020 8:00	202	78	-14	39	101	526	214	-13					
7	25/3/2020 8:00	318	115	29	36	121	648	208	-8					
8	25/3/2020 8:00	458	143	153	31	141	756	203	-5					
9	25/3/2020 8:00	565	162	320	29	164	894	199	-2					
10	25/3/2020 8:00	621	168	423	27	184	1002	196	-1					
11	25/3/2020 8:00	696	173	587	25	204	1124	193	0					
12	25/3/2020 8:00	763	176	707	23	219	1200	191	1					
13	25/3/2020 8:01	778	174	753	22	219	1200	191	1					
14	25/3/2020 8:01	779	171	754	22	219	1200	191	1					
15	25/3/2020 8:01	777	178	745	23	219	1200	191	1					
16	25/3/2020 8:01	779	177	742	23	219	1200	191	1					
17	25/3/2020 8:01	778	172	744	22	219	1200	191	1					
18	25/3/2020 8:01	778	179	743	23	219	1200	191	1					
19	25/3/2020 8:01	779	173	744	22	219	1200	191	1					
20	25/3/2020 8:01	775	175	743	23	219	1200	191	1					
21	25/3/2020 8:01	779	179	743	23	219	1200	191	1					

Figura 5-12 Excel de recolección de datos

Realizado por: Fredy Romero, 2022

CONCLUSIONES

- Se identificó los parámetros eléctricos de monitoreo más frecuentes utilizados en procesos productivos, en los cuales existen mayores riesgos de posibles fallas o baja eficiencia, llegando a determinar los siguiente: frecuencia, corriente, velocidad, los cuales se muestran en la interfaz gráfica, además se utilizó el lenguaje de programación Phyton, siendo un software libre nos permite utilizar librerías para diseñar la interfaz gráfica, basada en los parámetros determinados y dando respuesta a la necesidad del proceso.
- Se estableció la comunicación Modbus de manera correcta, sin necesidad de obtener una licencia de paga, por su amplio soporte dentro del diseño de las HMI, permitiendo acoplar varios equipos con un bajo costo de desarrollo.
- El sistema implementado nos permite el control y monitoreo de los parámetros definidos que se muestran en la HMI, es de bajo costo y benefician al desarrollo de nuevas aplicaciones, en las cuales se podrán implementar parámetros diferentes de acuerdo con los procesos productivos y la identificación de variables esenciales para los mismos.
- Las mediciones de los diferentes parámetros eléctricos, utilizando el sistema implementado, y comparándolo con las mediciones tomadas con instrumentos físicos no superan el límite de error permisible, estando dentro del rango de aceptación, considerando que las mediciones con instrumentos físicos desencadenan una serie de errores humanos o de precisión, el sistema de control implementado nos da valores en tiempo real.
- El diseño contribuye a que pequeñas y medianas empresas tengan acceso a tecnología de calidad a un bajo costo, incrementando así su capacidad de competitividad en el mercado.

RECOMENDACIONES

- Verificar antes de implementar el sistema, que los esclavos sean compatibles con el maestro, respetando así la arquitectura de este.
- Identificar correctamente los parámetros eléctricos del proceso en el que se va a implementar la interfaz, y modificar el código de programación de acuerdo con cada proceso.

BIBLIOGRAFÍA

(99+) *Ingeniería-de-Control-Moderna-Ogata-5ed* / Jefferson Maciel Leite - Academia.edu. (n.d.). Retrieved March 14, 2022, from https://www.academia.edu/9814191/Ingenieria_de_Control_Moderna_Ogata_5ed

Barandica López, A. (2010). Análisis del desempeño del protocolo HART sobre par trenzado en diferentes configuraciones. Performance analysis of the HART protocol over twisted pair in different configurations. *Revista Energía y Computación*, 18.

Carrillo Paz, A. (2011). Sistemas automáticos de control fundamentos básicos de análisis y modelado. *Sistemas Automáticos de Control Fundamentos Básicos de Análisis y Modelado*, 255.

CONTROLADORES LÓGICOS PROGRAMABLES (PLC) - Micro Automación. (n.d.). Retrieved March 15, 2022, from <https://ar.microautomacion.com/es/catalog/controladores-logicos-programables-3/>

de Comunicación, R. (n.d.). *PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR FACULTAD DE INGENIERÍA MAESTRÍA EN REDES DE COMUNICACIÓN INVESTIGACIÓN PREVIA LA OBTENCIÓN DEL TÍTULO DE MAESTRÍA “DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE CONTROL Y COMUNICACIÓN POR INTERNET PARA REPORTE DE PROCESOS INDUSTRIALES PARA TOMA DE DECISIONES A NIVEL GERENCIAL” FAUSTO ERNESTO OROZCO IGUASNIA.*

DEVICENET / facultad-ingenieria. (n.d.). Retrieved July 11, 2022, from <https://instrumentacionuc.wixsite.com/facultad-ingenieria/copia-de-controlnet>

Entornos Integrados de Automatización. (6113).

GitHub - ARMAGEDONgtx/IoT-system-PLC-data-to-InfluxDB: el objetivo de este proyecto es proporcionar software gratuito para obtener datos de plcs (Siemens S7-300/400/1200/1500) y almacenarlos. La pila usada es completamente de código abierto. Usé InfluxDB como almacenamiento de datos, por lo que el principio de la aplicación sigue el paradigma Big Data. (n.d.). Retrieved March 24, 2022, from <https://github.com/ARMAGEDONgtx/IoT-system-PLC-data-to-InfluxDB>

Interface de comunicaciones CEM M-ETH. (n.d.). Retrieved July 11, 2022, from www.circutor.com

Juan, S., & Alberto Perez Ing Analía Perez Hidalgo Bioing Elisa Perez Berenguer, M. (n.d.). *Universidad Nacional de "INTRODUCCION A LOS SISTEMAS DE CONTROL Y MODELO MATEMÁTICO PARA SISTEMAS LINEALES INVARIANTES EN EL TIEMPO."*

Mandado Pérez, E., Marcos Acevedo, J., Fernández Silva, C., Armesto Quiroga, I., Luis Rivas López, J., & María Núñez Ortuño, J. (2018). *SISTEMAS DE AUTOMATIZACIÓN Y AUTÓMATAS PROGRAMABLES*. www.marcombo.com

Miyara, F., & Nacional De Rosario, U. (n.d.). *CONVERSORES D/A Y A/D / D*. Retrieved March 20, 2022, from <http://www.fceia.unr.edu.ar/enica3>

Patricia, I., & Martinez, A. (n.d.). *FACULTAD DE INGENIERIA MECANICA Y ELECTRICAZVUTSRPONMLJIGFED DIVISION DE ESTUDIOS DE POST-GRADO zywxvutsrqponmljihgfedcbaYV POR: utrponmljigfedcbaZVUTSRPONMLKJIHGFEDCBA*.

Pirámide de automatización. (n.d.). Retrieved March 20, 2022, from <https://www.smctraining.com/es/webpage/indexpage/311>

Repositorio Digital - EPN: Diseño y construcción de un controlador autómatas programable en lenguaje FBD con software de simulación. (n.d.). Retrieved March 20, 2022, from <https://bibdigital.epn.edu.ec/handle/15000/1360?locale=en>

Rodríguez Penin, A. (n.d.). *Comunicaciones industriales : guía práctica*. 286.

Sistemas Industriales Distribuidos Profibus Profibus. (n.d.).

Técnicas de automatización avanzadas en procesos industriales - Dialnet. (n.d.). Retrieved March 20, 2022, from <https://dialnet.unirioja.es/servlet/tesis?codigo=60>

ANEXOS

Anexo A

Código de la Interfaz Gráfica

```
from tkinter import *
import tkinter.font as tkFont
import tkinter
from tkinter import ttk
import threading
import signal
import time
from tokenize import Double
import snap7
import sys

global plc

parametrosVariador = {'Frequency': 0,
                       'Speed': 0,
                       'Current': 0,
                       'Torque': 0,
                       'ActualPWR': 0,
                       'TotalKWH': 0,
                       'DCBusVolts': 0,
                       'Reference': 0,
                       'RatedPWR': 0,
                       'OutputVolts': 0,
                       'FWDREV': 0,
                       'STOPRUN': 0,
                       'ATMaxFrec': 0,
                       'ControlMode': 0,
                       'Enabled': 0,
                       'ReadyToRun': 0}
```

```
}
```

```
IP = '192.168.100.50'
```

```
RACK = 0
```

```
SLOT = 1
```

```
DB_NUMBER = 4
```

```
START_ADDRESS = 0
```

```
SIZE = 12
```

```
DB_NUMBER_1 = 5
```

```
START_ADDRESS_1 = 0
```

```
SIZE_1 = 30
```

```
bgColorScreen = "#333333"
```

```
fgColorLabels = "white"
```

```
bgColorButton = "#11868B"
```

```
bgColorGaugeV = "#17C1D0"
```

```
bgColorGaugeI = "#F93E2B"
```

```
bgColorGaugeS = "#17CF64"
```

```
bgColorGaugeF = "#873ABC"
```

```
fgColorColor = "white"
```

```
global vg
```

```
vg = 0
```

```
def signal_handler(signum, frame):
```

```
    sys.exit()
```

```
signal.signal(signal.SIGINT, signal_handler)
```

```
class Graphics():
```

```
    pass
```

```
def btn_clicked():
```



```

start_btn.configure(bg = 'green')
print("Button Clicked")

def connect_menu_init():
    global root, connect_btn, refresh_btn, graph, graph1, graph2, graph3, start_btn, stop_btn,
    inv_h_btn, inv_a_btn, style, plc, editFrecuencia

    root = Tk()
    root.title("Control PLC Variador")
    root.geometry("1280x720")
    root.config(bg=bgColorScreen)

    current_value = DoubleVar()

    font_port_bd = tkFont.Font(family="Arial", size=12, weight="bold")
    font_port_lb = tkFont.Font(family="Verdana", size=16, weight="bold")

    port_label = Label(root, text="CONTROLES VARIADOR: ", bg=bgColorScreen,
fg=fgColorLabels)
    port_label.grid(column=1, row=2, pady=20, padx=10)
    port_label.configure(font=font_port_lb)

    start_btn = Button(root, text="START", height=2, width=10, bg = bgColorButton,
fg=fgColorColor, command=lambda: btnClicked(DB_NUMBER,0,1,0,0,plc))
    start_btn.place(
        x = 150, y = 80,
        width = 130,
        height = 55)
    #start_btn.grid(column=1, row=3, pady = 20)
    start_btn.configure(font=font_port_bd)

    stop_btn = Button(root, text="STOP", height=2, width=10, bg = bgColorButton,
fg=fgColorColor, command= lambda: btnClicked(DB_NUMBER,0,1,0,1,plc))
    stop_btn.place(
        x = 150, y = 155,
        width = 130,

```

```

height = 55)
stop_btn.configure(font=font_port_bd)

inv_h_btn = Button(root, text="HORARIO", height=2, width=10, bg = bgColorButton,
fg=fgColorColor, command=lambda: btnClicked(DB_NUMBER,0,1,0,0,plc))
inv_h_btn.place(
x = 70, y = 230,
width = 130,
height = 55)
inv_h_btn.configure(font=font_port_bd)

inv_a_btn = Button(root, text="ANTIHORARIO", height=2, width=10, bg = bgColorButton,
fg=fgColorColor, command=lambda: btnClicked(DB_NUMBER,0,1,0,2,plc))
inv_a_btn.place(
x = 225, y = 230,
width = 130,
height = 55)
inv_a_btn.configure(font=font_port_bd)

connect_btn = Button(root, text="CONNECT", height=2, width=10, state="disabled", bg =
bgColorButton, fg=fgColorColor, command=connection_select)
connect_btn.place(
x = 150, y = 480,
width = 130,
height = 55)
connect_btn.configure(font=font_port_bd)

freq_lable = Label(root, text="VELOCIDAD(0-100 %): ", bg=bgColorScreen,
fg=fgColorColor)
freq_lable.place(
x = 20, y = 300,
width = 280,
height = 55)
freq_lable.configure(font=font_port_lb)

editFrecuencia = Entry(root,justify = 'center', bd =5)
editFrecuencia.place(

```

```

        x = 70, y = 375,
        width = 130,
        height = 53)
editFrecuencia.configure(font=font_port_bd)

enviar_btn = Button(root, text="FRECUENCIA", height=2, width=10, bg = bgColorButton,
fg=fgColorColor, command=freq_changed)
enviar_btn.place(
    x = 225, y = 375,
    width = 130,
    height = 55)
enviar_btn.configure(font=font_port_bd)

connection_select()

graph = Graphics()

gaudgeVolt_lable    =    Label(root,    text="TEMPERATURA",    bg=bgColorScreen,
fg=fgColorColor)
gaudgeVolt_lable.place(
    x = 625, y = 60,
    width = 150,
    height = 55)
gaudgeVolt_lable.configure(font=font_port_lb)

graph.canvas    =    Canvas(root,    width=200,    height=200,    bg=bgColorScreen,
highlightthickness=0)
graph.canvas.place(
    x = 600, y = 100,
    width = 200,
    height = 200)
# Dynamic update
graph.outer = graph.canvas.create_arc(
    10, 10, 190, 190, start=90, extent=100, outline= bgColorGaugeV, fill=bgColorGaugeV,
width=2)
# Static

```

```

graph.canvas.create_oval(
    75, 75, 125, 125, outline= bgColorGaugeV, fill="white", width=2)
# Dynamic update
graph.text = graph.canvas.create_text(
    95, 100, anchor=E, font=("Helvetica", "12"), text="---")
# Static
graph.canvas.create_text(
    110, 100, anchor=CENTER, font=("Helvetica", "12"), text="°C")

graph1 = Graphics()

gaugeCurr_lable = Label(root, text="CORRIENTE", bg=bgColorScreen, fg=fgColorColor)
gaugeCurr_lable.place(
    x = 925, y = 60,
    width = 150,
    height = 55)
gaugeCurr_lable.configure(font=font_port_lb)

graph1.canvas = Canvas(root, width=200, height=200, bg=bgColorScreen,
highlightthickness=0)
graph1.canvas.place(
    x = 900, y = 100,
    width = 200,
    height = 200)
# Dynamic update
graph1.outer = graph1.canvas.create_arc(
    10, 10, 190, 190, start=90, extent=100, outline=bgColorGaugeI, fill=bgColorGaugeI,
width=2)
# Static
graph1.canvas.create_oval(
    75, 75, 125, 125, outline=bgColorGaugeI, fill="white", width=2)
# Dynamic update
graph1.text = graph1.canvas.create_text(
    95, 100, anchor=E, font=("Helvetica", "12"), text="---")
# Static
graph1.canvas.create_text(
    110, 100, anchor=CENTER, font=("Helvetica", "12"), text="A")

```

```

graph2 = Graphics()

gaudgeVel_label = Label(root, text="VELOCIDAD", bg=bgColorScreen, fg=fgColorColor)
gaudgeVel_label.place(
    x = 625, y = 360,
    width = 150,
    height = 55)
gaudgeVel_label.configure(font=font_port_lb)

graph2.canvas = Canvas(root, width=200, height=200, bg=bgColorScreen,
highlightthickness=0)
graph2.canvas.place(
    x = 600, y = 400,
    width = 200,
    height = 200)
# Dynamic update
graph2.outer = graph2.canvas.create_arc(
    10, 10, 190, 190, start=90, extent=100, outline=bgColorGaudgeS, fill=bgColorGaudgeS,
width=2)
# Static
graph2.canvas.create_oval(
    75, 75, 125, 125, outline=bgColorGaudgeS, fill="white", width=2)
# Dynamic update
graph2.text = graph2.canvas.create_text(
    95, 100, anchor=E, font=("Helvetica", "10"), text="---")
# Static
graph2.canvas.create_text(
    110, 100, anchor=CENTER, font=("Helvetica", "10"), text=" RPM")

graph3 = Graphics()

gaudgeFeq_label = Label(root, text="FRECUENCIA", bg=bgColorScreen, fg=fgColorColor)
gaudgeFeq_label.place(
    x = 925, y = 360,
    width = 150,
    height = 55)

```

```

gaugeFreq_label.configure(font=font_port_lb)

graph3.canvas = Canvas(root, width=200, height=200, bg=bgColorScreen,
highlightthickness=0)
graph3.canvas.place(
x = 900, y = 400,
width = 200,
height = 200)
# Dynamic update
graph3.outer = graph3.canvas.create_arc(
    10, 10, 190, 190, start=90, extent=100, outline=bgColorGaugeF, fill=bgColorGaugeF,
width=2)
# Static
graph3.canvas.create_oval(
    75, 75, 125, 125, outline=bgColorGaugeF, fill="white", width=2)
# Dynamic update
graph3.text = graph3.canvas.create_text(
    95, 100, anchor=E, font=("Helvetica", "12"), text="---")
# Static
graph3.canvas.create_text(
    110, 100, anchor=CENTER, font=("Helvetica", "12"), text="Hz")

def connect_check(status):
    if status:
        connect_btn["state"] = "disable"
        connect_btn.configure(bg = 'green', text="CONNECTED")
    else:
        connect_btn["state"] = "active"
        connect_btn.configure(bg = 'white', text="CONNECT")

def connection_select():
    global plc
    try:
        plc = snap7.client.Client()
        plc.connect(IP, RACK, SLOT)

```

```

        state = plc.get_cpu_state()
        connect_check(True)
        connection()
except:
    connect_check(False)
    pass

def btnClicked(dbNumber, startAdd, lenDB, byte, bit, plc):
    try:
        db = plc.db_read(dbNumber,startAdd,lenDB)
        snap7.util.set_bool(db,byte,bit,True)
        plc.db_write(dbNumber,startAdd,db)
        time.sleep(0.100)
        snap7.util.set_bool(db,byte,bit,False)
        plc.db_write(dbNumber,startAdd,db)
        print("Button Clicked")
        boolstate=snap7.util.get_bool(db,byte,bit)
        if (boolstate == TRUE):
            snap7.util.set_bool(db,byte,bit,False)
            print(boolstate)
    except:
        btnClicked(dbNumber, startAdd, lenDB, byte, bit, plc)
        pass

def freq_changed():
    global plc, editFrecuencia
    db = plc.db_read(4,2,2)
    print(editFrecuencia.get())
    snap7.util.set_int(db,0,int(editFrecuencia.get()))
    plc.db_write(4,2,db)
    time.sleep(0.100)

def graph_control(graph,graph1,graph2,graph3):
    graph.canvas.itemconfig(
        graph.outer, exten=int(359*graph.voltage/100))
    graph.canvas.itemconfig(

```

```

graph.text, text=f"{float(graph.voltage)}")

graph1.canvas.itemconfig(
    graph1.outer, exten=int(359*graph1.current/6))
graph1.canvas.itemconfig(
    graph1.text, text=f"{int(graph1.current)}")

graph2.canvas.itemconfig(
    graph2.outer, exten=int(359*graph2.speed/1200))
graph2.canvas.itemconfig(
    graph2.text, text=f"{int(graph2.speed)}")

graph3.canvas.itemconfig(
    graph3.outer, exten=int(359*graph3.Frecuency/61))
graph3.canvas.itemconfig(
    graph3.text, text=f"{int(graph3.Frecuency)}")

def readDB():
    #print("thread start")
    global plc, graph ,graph1, graph2, graph3, vg

    dbM = { }
    dbL = { }

    while plc.get_cpu_state():
        try:
            j = 0
            dbmonitoreo = plc.db_read(DB_NUMBER_1, START_ADDRESS_1, SIZE_1)
            for k, v in parametrosVariador.items():
                v = int.from_bytes(dbmonitoreo[j:j+2], byteorder='big', signed=True)
                dbM.setdefault(k,v)
                j = j + 2
            #print(dbM)
            dblogo = plc.db_read(6, 0, 2)
            tempPLC = int.from_bytes(dblogo[0:2], byteorder='big', signed=True)
            dbL.setdefault("Temperatura",tempPLC)

```



```

#print(tempPLC)
voltage = dbM.get('OutputVolts')
current = dbM.get('Current')/100
speed = (dbM.get('Speed'))
Frecuency = dbM.get('Frequency')/100
turn = dbM.get('FWDREV')
temperatura = tempPLC
#print(speed)
#print(temperatura)
estate = 0
if (turn == 1):
    inv_h_btn.configure(fg="#79da73")
    inv_a_btn.configure(fg="white")
    estate = 1
else:
    if(turn == 0 ):
        inv_a_btn.configure(fg="#79da73")
        inv_h_btn.configure(fg="white")
        estate = 0

dbM = { }

graph.voltage = temperatura/10
graph1.current = current
graph2.speed = speed
graph3.Frecuency = Frecuency

#print(graph.voltage)

#if vg == 0:
t2 = threading.Thread(target=graph_control, args=(graph,graph1,graph2,graph3))
t2.daemon = True
t2.start()
vg = 1

time.sleep(0.4)

```

```
except:
    readDB()
    pass
```

```
def connection():
    try:
        t1 = threading.Thread(target=readDB)
        t1.daemon = True
        t1.start()
    except:
        pass
```

```
def close_window():
    global root, serialData
    serialData = False
    root.destroy()
    quit()
```

```
connect_menu_init()
root.protocol("WM_DELETE_WINDOW", close_window)
root.mainloop
```

Anexo B

