



UNIVERSIDAD
DE LOS ANDES
MÉRIDA VENEZUELA

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA

DESARROLLO CONCEPTUAL DE UN SISTEMA SCADA
BASADO EN SOFTWARE LIBRE Y DISEÑO DE LOS
MÓDULOS SERVIDOR Y COMUNICACIÓN, PARA EL
LABORATORIO DE CONTROL DE LA ESCUELA DE
INGENIERÍA ELÉCTRICA DE LA UNIVERSIDAD DE LOS
ANDES.

Br. Leonel Leandro Rondón Santiago

Mérida, Julio, 2022



UNIVERSIDAD
DE LOS ANDES
MÉRIDA VENEZUELA

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA

DESARROLLO CONCEPTUAL DE UN SISTEMA SCADA
BASADO EN SOFTWARE LIBRE Y DISEÑO DE LOS
MÓDULOS SERVIDOR Y COMUNICACIÓN, PARA EL
LABORATORIO DE CONTROL DE LA ESCUELA DE
INGENIERÍA ELÉCTRICA DE LA UNIVERSIDAD DE LOS
ANDES.

Trabajo de Grado presentado como requisito parcial para optar al título de Ingeniero
Electricista

Br. Leonel Leandro Rondón Santiago

Tutor(es): M.Sc. Oscar Enrique Blanco Ortíz

Mérida, Julio, 2022

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA

**DESARROLLO CONCEPTUAL DE UN SISTEMA SCADA BASADO
EN SOFTWARE LIBRE Y DISEÑO DE LOS MÓDULOS SERVIDOR
Y COMUNICACIÓN, PARA EL LABORATORIO DE CONTROL DE
LA ESCUELA DE INGENIERÍA ELÉCTRICA DE LA UNIVERSIDAD
DE LOS ANDES.**

Br. Leonel Leandro Rondón Santiago

Trabajo de Grado, presentado en cumplimiento parcial de los requisitos exigidos para optar al título de Ingeniero Electricista, aprobado en nombre de la Universidad de Los Andes por el siguiente Jurado.

MSc. Francisco Javier Vilorio M.

Ing. David Alejandro Quintero Avendaño

MSc. Oscar Blanco

DEDICATORIA

A Dios padre Todopoderoso, por estar siempre a mi lado en este camino tan largo y darme las herramientas necesarias para cumplir esta meta tan añorada.

A mis padres, que han sido pilar fundamental durante toda mi carrera y por todo el apoyo brindado en cada momento, gracias por estar siempre pendiente de mí.

A mi abuela Hermelinda, quién está orgullosa de mí donde quiera que esté.

A mis demás familiares que siempre aportaron algo de ellos para llegar a cumplir esta meta, siempre lo he sabido valorar y agradecer.

A todos mis amigos que me han acompañado durante este camino, que me han ayudado a seguir adelante ante las adversidades y me han brindado su apoyo incondicional.

AGRADECIMIENTOS

Primeramente a Dios, que me ha dado la motivación para culminar éste trabajo de grado entre tantas adversidades.

A mis padres, que me han brindado toda la ayuda posible durante toda mi carrera y formación personal.

A todos los docentes y técnicos que durante mi trayectoria universitaria favorecieron mi formación académica, no sólo por los conocimientos impartidos o el tiempo invertido, sino por darme las herramientas que harán de mí un profesional capaz de enfrentar el más difícil de los desafíos.

Al ingeniero en sistemas Miguel Peña, que me ha apoyado incondicionalmente durante toda mi trayectoria universitaria y fue pieza fundamental en el logro de éste trabajo de grado.

A mi tutor Oscar Blanco, por darme la oportunidad y guiarme a realizar el trabajo de grado para culminar con mis requisitos y optar al título.

A Omaira Sánchez, quien me brindó su apoyo y motivación durante la etapa de investigación y desarrollo de éste trabajo.

A todos los compañeros y amigos que he conseguido durante todo este camino que me han ayudado de una u otra manera, en especial a Juan Mago, quien incondicionalmente me brindó su mano para ayudarme en cada momento de mi trayectoria universitaria.

A la ilustre alma máter, la Universidad de Los Andes, que me ha permitido vivir experiencias inolvidables, me ha dado la oportunidad de adquirir conocimientos valiosos para mi formación profesional.

Leonel Leandro Rondón Santiago. Desarrollo conceptual de un sistema SCADA basado en software libre y diseño de los módulos servidor y comunicación, para el laboratorio de control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes. Universidad de Los Andes. Tutor(es): M.Sc. Oscar Enrique Blanco Ortíz. Julio, 2022.

Resumen

En la automatización industrial se ha desarrollado una gran cantidad de sistemas SCADA, que son utilizados para la supervisión y control de procesos industriales, en su mayoría, estos sistemas son elaborados por los fabricantes de los equipos de control y utilizan software de tipo propietario, en los que las exigencias de software y hardware varían dependiendo de la aplicación, debido a esto, la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes, propone la elaboración y aplicación de un sistema SCADA de poca exigencia de hardware para su uso en el Laboratorio de Control que permita supervisar los procesos implementados en las prácticas de laboratorio de forma remota; para llevar esto a cabo, se hace un estudio de las herramientas de programación basadas en software libre y disponibles en la red, posteriormente se realice una selección adecuada a los criterios, necesidades y disponibilidades de los equipos que se encuentran en dicho laboratorio. Además, se hace un estudio sobre los módulos servidor y de comunicación del sistema, ya que con éstos se logra mantener activo el sistema en un servidor local; y se establece la comunicación del sistema con un PLC Siemens S7-300 que está disponible en el laboratorio, a través de un servidor OPC, el cual es un elemento de software que sirve como traductor de datos entre un punto y otro. Finalmente, se hace una integración de éstos módulos con otros desarrollados anteriormente, como son: el módulo sinóptico, módulo gráfico de tendencias, módulo de alarmas y eventos, y módulo base de datos; todo esto para almacenar variables, observar en pantalla el estado de las variables del PLC y consultar su comportamiento a través de gráficas.

Descriptor: Sistemas SCADA, PLC, software libre, comunicaciones industriales, protocolos de comunicación, Servidores OPC.

ÍNDICE GENERAL

DEDICATORIA	
.....	¡Error! Marcador no definido.v
AGRADECIMIENTO	v
RESUMEN	vi
ÍNDICE GENERAL	vii
ÍNDICE DE FIGURAS	xii
ÍNDICE DE TABLAS	xiv
INTRODUCCIÓN	1
CAPÍTULO I GENERALIDADES	3
1.1 EL PROBLEMA	3
1.2 JUSTIFICACIÓN	5
1.3 OBJETIVOS	6
1.3.1 Objetivo general.....	6
1.3.2 Objetivos específicos	7
1.4 METODOLOGÍA.....	7
1.5 ALCANCE.....	8
1.6 LIMITACIONES	9
CAPÍTULO II MARCO REFERENCIAL	10
2.1 ANTECEDENTES	10
2.2 MARCO TEÓRICO	12

2.2.1 Software libre	12
2.2.2 Lenguajes de programación	12
2.2.3 Lenguajes de programación <i>backend</i>	12
C - Programming	13
C++	13
JAVA	14
PHP	14
PYTHON	15
Visual Basic.NET	15
2.2.4 Lenguajes de programación <i>frontend</i>	15
HTML	15
JavaScript	16
CSS	16
2.2.5 Frameworks	16
Zend (PHP)	17
Openxava (JAVA)	17
Django (Python)	17
Flask	18
2.2.6 Editores de texto	19
Visual Studio Code	18
2.2.7 Sistema SCADA	18
2.2.8 Características de un sistema SCADA	20
2.2.9 Requisitos de un SCADA	21

2.2.10 Arquitectura general de un sistema SCADA.....	21
2.2.11 Componentes del hardware de un SCADA.....	22
Ordenador central o MTU.....	22
Unidades remotas o RTU.....	22
Red de comunicación.....	23
Instrumentos de campo.....	23
PLC.....	23
Sensores.....	23
Actuador.....	23
2.2.12 Estructura y componentes de un software SCADA.....	23
Configuración.....	23
Interfaz gráfica del operador o HMI.....	24
Módulo de proceso.....	24
Gestión y archivo de datos.....	24
2.2.13 Protocolos de comunicación.....	25
P-Net.....	25
Modbus.....	26
ASCII.....	26
RTU.....	26
TCP/IP.....	26
PROFINET.....	27
2.2.14 Automatización industrial.....	27
2.2.15 PLC.....	28

2.2.16 Siemens S7-300.....	29
2.2.17 Multi point interfaz.....	30
2.2.18 Servidor OPC	31
CAPÍTULO III SELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN Y DEFINICIÓN DE MÓDULOS SCADA.....	33
3.1 Selección del lenguaje de programación.....	33
3.2 Definición de los módulos que contemplan el sistema SCADA	39
3.2.1 Módulo sinóptico.....	39
3.2.2 Módulo de alarmas y eventos.....	40
3.2.3 Módulo de base de datos	40
3.2.4 Módulo de gráfico de tendencias	41
3.2.5 Módulo servidor	41
3.2.6 Módulo de comunicación	41
CAPÍTULO IV DESARROLLO DE MÓDULOS DEL SCADA	43
4.1 Desarrollo del módulo servidor	43
4.2 Desarrollo del módulo de comunicación	45
4.2.1 Librería OpenOPC	51
CAPÍTULO V INTEGRACIÓN DE MÓDULOS Y ANÁLISIS DE RESULTADOS	54
5.1 Integración del módulo servidor con el SCADA.....	54
5.2 Integración del módulo de comunicación con el SCADA.....	56
5.3 Análisis de resultados	63

CONCLUSIONES	68
RECOMENDACIONES	68
REFERENCIAS	69

www.bdigital.ula.ve

ÍNDICE DE FIGURAS

Figura N°1 Tipos de lenguajes de programación	13
Figura N°2 Diagrama de un sistema SCADA escalable	20
Figura N°3 Arquitectura básica de un sistema SCADA.....	21
Figura N°4 Componentes básicos de un SCADA	25
Figura N°5 Puertos de comunicación utilizados en SCADA	27
Figura N°6 Automatización industrial con un sistema SCADA	28
Figura N°7 PLC Siemens S7-300.....	30
Figura N°8 Adaptador MPI/USB para Siemens S7-300	31
Figura N°9 Servidor OPC.....	32
Figura N°10 Módulos del sistema SCADA – ULA.	39
Figura N°11 Etapas del Módulo servidor	44
Figura N°12 Diagrama de flujo del módulo servidor.....	46
Figura N°13 Etapas del módulo de comunicación	47
Figura N°14 Diagrama de flujo del módulo de comunicación.....	48
Figura N°15 Diagrama escalera para las variables en el Step7	49
Figura N°16 Conexión entre el PLC y el servidor IBH.....	49
Figura N°17 Verificación de variables del PLC en el IBH	50
Figura N°18 Transferencia de datos al servidor IBH	50
Figura N°19 Cliente OPC en aplicación testopc	51

Figura N°20 Diagrama de flujo del encendido de las variables del PLC	52
Figura N°21 Diagrama de flujo de la escritura del estado de las variables desde archivo “views.py”	53
Figura N°22 Integración del módulo servidor con el SCADA.....	55
Figura N°23 Integración del módulo de comunicación con el SCADA.	57
Figura N°24 Diagrama de flujo para establecer condiciones iniciales en cada proceso .	58
Figura N°25 Condiciones iniciales en OFF vistas desde la interfaz del usuario.....	59
Figura N°26 Diagrama de flujo para encender las variables del proceso 1.....	61
Figura N°27 Diagrama de flujo para encender las variables del proceso 2.....	61
Figura N°28 Condiciones necesarias para encender el Disyuntor_A en el proceso 2.....	62
Figura N°29 Diagrama de flujo para encender las variables del proceso 3.....	63
Figura N°30 Condiciones necesarias para apagar las variables del proceso 3	64
Figura N°31 Gráfica de datos históricos para la variable “BOMBA_PRINCIPAL”.....	65
Figura N°32 Alarmas históricas generadas en el proceso 3	66
Figura N°33 HMI del proceso 2 mientras se realiza la lectura continua de las variables	66
Figura N°34 Funcionamiento del SCADA conectado al PLC	67

ÍNDICE DE TABLAS

Tabla N°1 Comparación de lenguajes de programación <i>backend</i>	33
Tabla N°1 Comparación de lenguajes de programación <i>backend</i> . (continuación).....	34
Tabla N°2 Características de Python.....	35
Tabla N°3 Características de PostgreSQL.....	36
Tabla N°4 Comparación entre Django y Flask.....	37
Tabla N°5 Comparación de lenguajes <i>frontend</i>	37
Tabla N°5 Comparación de lenguajes <i>frontend</i> (continuación).....	38

www.bdigital.ula.ve

INTRODUCCIÓN

En la actualidad y desde hace una gran cantidad de años, la industria se ha encargado de innovar y contar con diferentes sistemas que permiten controlar y manipular procesos industriales, así como también, observar este control a través de una interfaz gráfica amigable para el usuario, lo que permite obtener información detallada de las variables del proceso a través de los diferentes dispositivos controladores que se utilizan para este fin, como los Controladores Lógicos Programables (PLC, *Programmable Logic Controller* - por sus siglas en inglés), que son de amplio uso debido a su gran cantidad de servicios y utilidades que prestan para supervisar los procesos; adicionalmente se tiene el uso de las Interfaces Humano - Máquina (HMI, *Human - Machine Interface* - por sus siglas en inglés) como los dispositivos más comunes para la visualización, ya que proporcionan una integración eficiente con los PLC.

Adicionalmente, en la industria se están implementando sistemas de Supervisión, Control y Adquisición de Datos (SCADA, *Supervisory, Control And Data Acquisition* - por sus siglas en inglés); los cuales son sistemas capaces de obtener, procesar y mostrar información de distintos procesos industriales para controlar múltiples variables de forma automática desde la pantalla de un computador. Es por ello que permiten una visualización del proceso en tiempo real y a distancia, lo cual es importante si se presentan dificultades para que un operario observe el proceso durante todo su transcurso; debido a que mediante éstos sistemas se realiza una adquisición y almacenamiento de datos de gran relevancia al momento de tomar decisiones y presentar informes de la industria.

Por otro lado, los sistemas SCADA normalmente son desarrollados por empresas de software propietarias dentro del área del control de procesos industriales y automatización, que son los mismos fabricantes de muchos dispositivos instalados en campo, y que además de vender su software a precios elevados no suelen prestar servicios a pequeñas empresas o

instituciones académicas. En consecuencia, este trabajo se desarrolla debido a la oportunidad de crear y diseñar sistemas SCADA bajo programación basada en software libre, ya que éstos permiten crear procesos pequeños y acordes con la necesidad de quien los diseñe.

Cabe destacar que, aunque esto presenta una rentabilidad económica debido a que no se paga ningún servicio externo, se requieren conocimientos en el área de programación y control de procesos industriales, así como en lenguajes de programación, desarrollo de aplicaciones web, montajes de proyectos en servidores web, manejos de bases de datos, protocolos de comunicaciones compatibles con los PLC, programación de PLC, entre otros.

Es por ello que se presenta la oportunidad de adquirir conocimientos necesarios en el área de la programación y aplicarlos para el diseño de un sistema SCADA basado en software libre y aplicar control avanzado de procesos, haciendo uso a su vez de las herramientas que se consiguen en el Laboratorio de Control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes, como es el PLC Siemens S7-300.

www.bdigital.ula.ve

CAPÍTULO I

GENERALIDADES

El presente capítulo describe las bases necesarias para ejecutar esta investigación, estableciendo las distintas razones que permiten crear las condiciones del proyecto, las mismas se describen a continuación:

1.1 EL PROBLEMA

El desarrollo de un país está altamente influenciado por el desarrollo tecnológico de sus industrias, de sus empresas y de sus instituciones educativas; por lo tanto, para un país como Venezuela, estas juegan un papel indispensable y fundamental para su crecimiento. Es así como cada una de ellas necesita de equipos tecnológicos y vitales para mantener activos sus procesos productivos o en su defecto hacer pruebas y simulaciones previas al proceso industrial. Entre los equipos disponibles y utilizados en la industria se encuentran los PLC, y cada uno de ellos viene acompañado de su respectivo software para computadora para así tener un uso óptimo de los aparatos.

Dentro del ámbito de la industria, los PLC son utilizados para trabajar en conjunto con sistemas SCADA, con el fin de que el operador del sistema tenga la posibilidad de supervisar, monitorear y controlar el estado operativo del proceso y de los distintos componentes del sistema que se está operando, sin necesidad de estar presente en cada uno de los dispositivos; y de esta manera poder disminuir los problemas que surgen y resolverlos de forma rápida.

Los fabricantes de los PLC ofrecen entre sus productos, los software para configurar sus dispositivos y los sistemas para ejecutar su supervisión de manera remota, los cuales

forman parte del universo de los software propietarios, es decir, están elaborados con tecnologías que son cerradas por las que hay que pagar costosas licencias de uso [1], lo que representa una desventaja para las instituciones o empresas que no cuentan con los recursos económicos o un presupuesto necesario para adquirirlo. Además se generan otros inconvenientes, ya que los fabricantes también imponen restricciones en el uso de éstos paquetes computacionales; pues impiden copiar, reproducir, distribuir, modificar y personalizar el software.

Es por ello que, se han constituido y se han impulsado estudios, desarrollos y distribuciones de sistemas SCADA bajo software libre o código abierto por parte de personas, comunidades y empresas con el fin de ayudar y ofrecer alternativas flexibles a quienes no pueden adquirir éstos software de propietarios convencionales. El software libre se encuentra en una etapa fuerte de crecimiento, compitiendo al mismo nivel que el software propietario, debido a sus grandes beneficios como compatibilidad con distintos fabricantes, mayor seguridad que estos ofrecen en comparación a los comerciales y sobre todo por tener un bajo costo y en algunos casos inclusive gratuito. [2]

Muchas de estas alternativas aprovechan tecnologías web que no estaban disponibles durante la concepción de los primeros sistemas SCADA, por lo que esto facilita la creación y acceso a HMI con capacidades equivalentes a las de una interfaz escrita en código nativo por un propietario, pero al usar estándares web como plataforma, se rompen los inconvenientes de sistemas operativos y topologías de red que normalmente limitan a otros sistemas SCADA.

Desde el punto de vista de una institución educativa, resulta conveniente el uso de un sistema SCADA de software libre. El hecho de que sea de código abierto, implica que se tiene acceso a su código fuente y a todos los detalles sobre su funcionamiento. Por lo tanto, futuros estudiantes pueden modificarlo y mejorarlo libremente con el fin de utilizarlo como una plataforma para la exploración de nuevas tecnologías y nuevos avances dentro del área de control de procesos.

En la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes se han realizado varios trabajos de grado relacionados con los sistemas SCADA desarrollados en software libre, por ejemplo “Diseño e Implementación de un Sistema SCADA para Prácticas Demostrativas del Laboratorio de Control Utilizando el PLC Siemens S7-300 y Software Libre” por Uzcátegui, M. [3]. Además del diseño e implementación de varios módulos para un sistema SCADA en el laboratorio de control, titulados “Desarrollo de módulo sinóptico para un sistema SCADA basado en software libre para ser implementado en el Laboratorio de Control de la Facultad de Ingeniería” por Rojas, J. [4] y “Diseño de los módulos base de datos, gráfico de tendencias, alarmas y eventos para un sistema SCADA basado en software libre que será implementado en el Laboratorio de Control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes” por Montaña, A. [5]. Sin embargo, las posibilidades de comunicar el SCADA con algún PLC no han sido exploradas hasta la fecha.

El Laboratorio de Control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes dispone de varios PLC que tienen gran potencial didáctico y académico que sirven como experiencia para el uso que se le puede dar dentro de la industria en el momento que los estudiantes culminen sus estudios. Por lo tanto la materia de Laboratorio de Control se ve en la necesidad de innovar e incorporar continuamente nuevas tecnologías para el desarrollo de sus temas académicos, lo que les brindará a los estudiantes mayor experiencia al realizar prácticas utilizando PLC junto a sistemas SCADA.

1.2 JUSTIFICACIÓN

Motivado a la importancia en el uso de PLC en el ámbito educativo e industrial, por supervisar y controlar procesos en la producción con el uso de sistemas SCADA, las empresas han invertido cada día más en tecnología y recursos en busca de mejorar los procesos, aumentando la producción, optimizando los tiempos de trabajo, y elevando la calidad para obtener mayores ganancias con menores costos. Por lo que se tiene un continuo desarrollo de nuevos dispositivos de automatización y control con mayores beneficios para el operador que permita la interacción entre los distintos componentes conectados al sistema.

La búsqueda de nuevos desarrollos de automatización de procesos, contempla el uso de programas que permitan supervisar, controlar y adquirir datos dentro de la planta en tiempo real y con la facilidad de observar todos los dispositivos en una sola interfaz. Al mismo tiempo, los avances tecnológicos y los altos costos que conlleva el uso de un sistema SCADA de propietario, crea la necesidad de encontrar nuevas formas de realizar la automatización con nuevas tecnologías libres y de código abierto, por lo que trabajar con un software libre además de ser innovador y de poco impacto económico, brinda la oportunidad de poder desarrollarlo según sus requerimientos.

En concordancia a lo anterior, se presenta la necesidad de crear un sistema SCADA dentro del Laboratorio de Control de la escuela de Ingeniería Eléctrica de la Universidad de Los Andes, pues es de gran ayuda y apoyo para la comunidad estudiantil contar con este sistema dentro de las instalaciones, ya que, promueve la integración a sistemas que se encuentran dentro de una industria sin necesidad de estar en las instalaciones de una.

Adicionalmente, el diseño de un sistema SCADA bajo software libre para el laboratorio de control de la escuela de ingeniería eléctrica de la Universidad de Los Andes, ayuda a fomentar el desarrollo de nuevas estrategias de estudio e impulsar las clases a distancia utilizando la tecnología, pues con éste sistema la comunidad estudiantil no tendría la necesidad de asistir con frecuencia al laboratorio, ya que se promoverían jornadas por parte de los docentes en los que el estudiante realice prácticas a través del sistema SCADA y monitorice variables de un proceso a distancia. El diseño de un sistema SCADA bajo uso de software libre ha surgido debido a dos motivos primordiales: Uso gratuito y de código abierto; por lo tanto al diseñar e implementar el SCADA con un programa de este tipo permitiría explorar áreas nuevas e innovadoras como el desarrollo de software y la programación, descubriendo en detalle sus ventajas y posibilidades.

1.3 OBJETIVOS

1.3.1 Objetivo general

- Conceptualizar un sistema SCADA basado en software libre y diseñar los módulos servidor y de comunicación para el Laboratorio de Control de la escuela de ingeniería eléctrica de la Universidad de Los Andes.

1.3.2 Objetivos específicos

- Definir todos los elementos que conforman un sistema SCADA basado en software libre para el laboratorio de control de la escuela de ingeniería eléctrica de la Universidad de Los Andes.
- Analizar las herramientas de programación de software libre más adecuadas para el diseño de los módulos del sistema SCADA.
- Diseñar los módulos de servidor y comunicación en software libre del sistema SCADA.
- Integrar los módulos desarrollados con el HMI del sistema SCADA.
- Realizar las pruebas de funcionamiento del sistema SCADA con un controlador lógico programable Siemens S7-300.

1.4 METODOLOGÍA

Debido al enfoque del proyecto, existen diferentes métodos de investigación. Según Hurtado [6], el método permite el análisis capaz de capturar la realidad en su proceso y en sus perspectivas de desarrollo y, al mismo tiempo, da la manera de actuar, el método contiene la estrategia y la táctica. Por lo tanto, se indica que la elección del método estará condicionada por el criterio del investigador, a su vez, éste trabajo se puede ubicar bajo el método analítico-sintético, ya que consiste en la descomposición del funcionamiento de un sistema SCADA para estudiar sus distintos elementos, partes o componentes y así obtener nuevos conocimientos acerca de dicho sistema. La finalidad del análisis radica pues, en conocer las partes de un sistema, determinar los nexos o relaciones que hay entre ellas y las leyes que rigen su desarrollo. Pues esto permite descubrir la estructura del mismo y poder diseñarlo acorde a lo planteado para relacionarlo con el PLC Simatic S7-300 y posteriormente realizar una conexión satisfactoria con él.

De la misma manera, según el propósito de la investigación existen diferentes estudios que la clasifican, de acuerdo al Manual de Trabajos de Grado de Especialización y Maestría y Tesis Doctorales de la Universidad Pedagógica Experimental Libertador [7], este trabajo de grado es una investigación de campo de tipo experimental y orientada a decisiones. La primera etapa está basada en la revisión bibliográfica, en la cual se indaga sobre los

aspectos teóricos de lo que conforma un sistema SCADA y las diferentes maneras de crear uno basado en software libre. Luego se realizará comparaciones de los diferentes aspectos que construyen un sistema SCADA en donde la búsqueda, recopilación, organización, valoración y crítica de la información permite la visión panorámica del problema y el planteamiento de soluciones.

Posteriormente la segunda etapa de la investigación está basada en la toma de decisiones en cuanto a las herramientas a utilizar de programación de software libre entre las disponibles para el desarrollo del sistema SCADA que se utilizará en el laboratorio de control de la escuela de ingeniería eléctrica de la Universidad de Los Andes.

Asimismo, la tercera etapa de la investigación está conformada por el diseño de los módulos servidor y comunicación en software libre del sistema SCADA, bajo las herramientas de programación seleccionadas anteriormente, además de integrarlos con los demás módulos existentes y la implementación de las pruebas del sistema SCADA con el PLC Siemens Simatic S7-300 del laboratorio de control.

Por último, la etapa final constará de la redacción y presentación de toda la información recaudada, comparaciones realizadas y limitaciones del sistema SCADA.

1.5 ALCANCE

Esta investigación ha logrado comunicar un PLC disponible del laboratorio de control, con un SCADA iniciado desde cero en el cual se han definido todos sus módulos, dicho sistema sirve para ser usado por el personal docente de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes y así mejorar las clases de la cátedra “Laboratorio de Control”, ya que con ello se pueden realizar prácticas y mostrar al estudiantado cómo se lleva a cabo la automatización industrial en las empresas. De igual manera se tiene acceso total al código fuente para realizar mejoras en un futuro, en beneficio del avance y los conocimientos impartidos por los docentes.

1.6 LIMITACIONES

Se presentan pocas limitaciones dentro del sistema SCADA implantado en el laboratorio de control, pero de gran importancia y que llevan a tener en cuenta ciertas cosas para el desarrollo de las clases y para modificaciones futuras; la mayor limitante que se tiene es que para comunicar el PLC con el SCADA se necesita un servidor OPC compatible con el PLC, el cual por no contar con una licencia paga sólo se puede usar por intervalos de 30 minutos; existen y se aplicaron varios servidores OPC (MatrikonOPC, Matrikon para Siemens, KepServerEX) de licencia gratuita pero no fueron compatibles con el PLC, debido a que los puertos de comunicación del PLC no son aceptados por éstos servidores.

www.bdigital.ula.ve

CAPÍTULO II

MARCO REFERENCIAL

En este capítulo se expone la revisión bibliográfica relacionada con SCADA basados en software libre, además de la revisión teórica de las herramientas necesarias para desarrollar los módulos de interés del sistema.

2.1 ANTECEDENTES

Santa López [8] en su proyecto de investigación “Plataforma web econtinua ASEUTP”, el autor desarrolla una aplicación web, automatizando los gestores de formularios utilizados para las inscripciones de sus cursos de educación, analiza los diferentes lenguajes de programación y herramientas de desarrollo web bajo software libre para una óptima ejecución del proyecto; detallando la diferencia entre los lenguajes de programación de uso *Backend* y *Frontend*, además de explicar las diferencias entre los patrones de diseño de una aplicación web como Modelo – Vista – Controlador, y Modelo – Vista – Template. Igualmente se explican las herramientas de desarrollo como los frameworks de cada lenguaje de programación. En efecto, el autor despliega las virtudes y características de los distintos lenguajes de desarrollo web disponible bajo software libre, en el que destaca el uso de Python y su *framework* Django para la creación de aplicaciones web.

Carrero [9] en su trabajo especial de grado “Diseño de un sistema de control supervisorio y adquisición de datos (SCADA) para el monitoreo remoto de los sistema de energía ininterrumpida (UPS) perteneciente al sistema eléctrico de una refinería en el país”, enfoca el desarrollo de un sistema SCADA para usarlo dentro del área industrial del país, específicamente en una refinería, en la que permite encontrar la solución a varias

problemáticas planteadas en el campo del suministro de energía eléctrica; mediante el trabajo destaca y define cada parámetro y componente que conforman al sistema SCADA, así como también destaca el nivel de importancia que representan estos sistemas para la continuidad de un proceso productivo dentro de una empresa.

González [10] en su trabajo especial de grado “Implementación de sistema de supervisión y control (SCADA) vía web mediante recursos de código abierto”, despliega todo el estudio de lo que es un sistema SCADA resaltando cada una de sus etapas que lo conforman, además evalúa diferentes alternativas de software libre disponibles para comparar entre ventajas y desventajas; además tiene un soporte de múltiples protocolos de comunicación, un apartado para histórico de mediciones, una Interfaz Hombre – Máquina diseñada en una aplicación web, además de un sistema de alarmas. Igualmente realizó una comparación de las diferencias existentes entre los sistemas SCADA propietarios respecto a las alternativas de software libre.

Uzcátegui, M.[3] en su trabajo de grado “Diseño e implementación de un sistema SCADA para prácticas demostrativas del laboratorio de control utilizando el PLC Siemens S7-300 y software libre”, diseña un sistema SCADA utilizando el software libre IndigoSCADA, destacando por encima de los demás que permite una comunicación bidireccional entre él y el PLC Siemens S7-300, utilizando un servidor OPC compatible con el autómatas; además define todo el funcionamiento que tiene un sistema SCADA bajo software libre y todas las características de funcionamiento del PLC Siemens S7-300; de ésta manera logra establecer satisfactoriamente, prácticas en el Laboratorio de Control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes, aunque limitadas por temporizaciones de 30 minutos.

Rojas, J. [4] en su proyecto de grado “Desarrollo de módulo sinóptico para un sistema SCADA basado en software libre para ser implementado en el laboratorio de control de la Facultad de Ingeniería”, se centra en diseñar el módulo sinóptico de un SCADA, que se implementa en el laboratorio de control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes, en el cual se analiza el funcionamiento de los sistemas SCADA, además, considera la utilización de redes como medio de comunicación, para así crear el

entorno gráfico interpretado por un explorador, donde se eligió Python como software de desarrollo y Django como *framework* de trabajo. Así mismo, se diseñaron tres procesos que se incorporan al módulo sinóptico del sistema SCADA, y para ser controlados con los Controladores Lógicos Programables (PLC) disponibles en el laboratorio.

Montaña, A. [5] en su tesis de grado “Diseño de los módulos base de datos, gráfico de tendencias, alarmas y eventos para un sistema SCADA basado en software libre que será implementado en el Laboratorio de Control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes”, implementa el desarrollo de los módulos de base de datos el cual permite almacenar toda la información del sistema de una forma organizada, contando con interfaces gráficas de usuarios que facilitan la creación, modificación y eliminación de variables y su configuración de alarmas, así como también implementa módulos para la gestión de alarmas y eventos, además de un módulo que permite la visualización de gráficos de tendencias que existen en el proceso tanto en tiempo real como históricas.

2.2 MARCO TEÓRICO

2.2.1 Software libre

Es el tipo de software que le da al usuario la libertad de usarlo, estudiarlo, modificarlo, mejorarlo, adaptarlo y redistribuirlo, con la única restricción de no agregar ninguna restricción adicional al software modificado, mejorado, adaptado o redistribuido (copyleft). Se debe aclarar que debe permitir el acceso al código fuente, debido a que ello es una condición imprescindible para ejercer las libertades de estudiarlo, modificarlo, mejorarlo y adaptarlo. De esta manera es socialmente justo, económicamente viable y tecnológicamente sustentable en el tiempo; ya que permite construir en términos de igualdad e inclusión de todos los seres humanos y los pueblos del mundo. [11]

2.2.2 Lenguajes de programación

Se conoce como lenguaje de programación a un programa destinado a la construcción de otros programas informáticos. Su nombre se debe a que comprende un lenguaje formal que está diseñado para organizar algoritmos y procesos lógicos que luego serán llevados a cabo por un ordenador o sistema informático. Se dividen en

lenguajes *backend* que se desarrollan desde la parte del servidor, comprende el almacenamiento de datos, llenado de tablas y demás acciones; y además, en los lenguajes *frontend*, que se desarrollan desde la parte del cliente, básicamente se centra en la parte visual del producto final. [11]



Figura N°1 Tipos de lenguajes de programación [11]

2.2.3 Lenguajes de programación *backend*

Los lenguajes de programación *backend* son aquellos que implementan sus desarrollos del lado del servidor, es decir, allí se programan funcionalidades tales como conexiones a las bases de datos, plantillas del lado del servidor, entre otras. Básicamente en estos lenguajes se desarrollan funcionalidades e instrucciones para recibir la información, procesarla y enviarla a las vistas del usuario ya sea en código HTML o también se podrían enviar los datos de manera pura para luego ser procesados por JavaScript y finalmente ser entregados en las vistas del usuario. Dentro de los lenguajes de programación *backend* más utilizados para desarrollo en código abierto de aplicaciones, se consiguen los siguientes: [11]

- **C Programming:** Es un lenguaje procedimental imperativo y estructurado con sistema de tipos débil y estático. Hereda características directamente de Fortran, denotando su antigüedad. C Programming se usa para desarrollar sistemas operativos, aplicaciones de escritorio, herramientas científicas e industriales, simuladores, animación 3D y otros usos avanzados. Además es un lenguaje de medio nivel con compatibilidad para programación a alto y bajo nivel, pero tiene

una gran desventaja ya que, no tiene soporte para abstracción, ocultación de datos, encapsulación, polimorfismos o herencia. De igual modo faltan constructores y depuradores. [11]

- **C++:** Es una extensión del lenguaje C Programming ya tratado anteriormente. Fue desarrollado en 1979 como lenguaje de programación multiparadigma con sistema de tipos fuerte, estático y nominativo. Es ampliamente utilizado en cualquier aplicación siendo casi omnipresente, los casos más excepcionales en los que C++ no tiene cabida son los sistemas extensos como aplicaciones webs ejecutadas desde exploradores, *backends*, en servidores y webs, así como en entornos empresariales con abundancia de lógica. Tiene una alta disponibilidad de librerías, compiladores y documentación al alcance del usuario; además de un número reducido de restricciones al contar con una librería estándar de pequeñas dimensiones. Por otra parte, obliga al usuario a definir múltiples tipos de datos básicos, su sintaxis es compleja y estricta. [11]
- **JAVA:** Se trata de un lenguaje imperativo orientado a objetos, de plataforma independiente y con un sistema de tipos fuerte y estático; además toma múltiples ideas de Pascal y C++. Java es pionero en traer flexibilidad y desarrollo guiado por pruebas a la programación, además busca principalmente que las aplicaciones sean escritas una sola vez y de allí pueda ser compilado en cualquier otro dispositivo. Así, Java es usado comúnmente para desarrollar aplicaciones para el sistema operativo Android, diversas soluciones de software orientado a usuarios, programas para el mundo financiero y mercantil, código para terminales punto de venta y soluciones de macro datos, por citar algunos de sus usos. [11]
- **PHP:** Es un lenguaje de programación *backend* el cual se ejecuta del lado del servidor, lo que quiere decir que antes de que la información o páginas sean vistas por el usuario, en el servidor las páginas pueden realizar una serie de diferentes acciones para las cuales están programadas (solicitudes a bases de datos, conexiones de red, entre otras) para posteriormente ser enviadas al usuario con la información que estos esperaban. [11]

- **PYTHON:** Se trata de un lenguaje de programación multiparadigma fuertemente tipado y dinámico. Toma prestadas características de una variedad de lenguajes anteriores, entre ellos Haskell, Lisp, Perl y Java. Actualmente es propiedad de la Python Software Foundation, una organización sin fines de lucro que distribuye la licencia de código abierto. Python se usa especialmente en robótica, scripting, inteligencia artificial, aprendizaje de máquina, diseño asistido por ordenador, desarrollo multimedia (excepto entornos interactivos 3D) y desarrollo de aplicaciones web bastante ligeras con un gran potencial para su uso. Presenta otras aplicaciones empresariales. Presenta una alta versatilidad, sencillez que favorece su uso y aprendizaje, y rapidez en el desarrollo. [11]
- **Visual Basic .NET:** es un lenguaje de programación orientado a objetos moderno, multiparadigma y con sistema de tipos estático, dinámico, fuerte, seguro y nominal. Es una evolución de Visual Basic, lenguaje con el que no es compatible. El software es diseño y propiedad de Microsoft, y lleva casi dos décadas en el mercado. [11]

2.2.4 Lenguajes de programación *frontend*

Los lenguajes de programación *frontend* son aquellos con los cuales se realizan diseños o implementaciones que están del lado del cliente, normalmente estas implementaciones son realizadas por desarrolladores *frontend* pero hay algunos diseñadores los cuales también manejan de excelente forma estos lenguajes, debido a que en el *frontend* se centra básicamente en el diseño y la estructura de lo que el usuario va a visualizar. Entre ellos se consiguen los siguientes: [12]

- **HTML:** Se encarga de definir la estructura básica de las páginas web mediante un código el cual está basado en etiquetas. Por lo tanto es un lenguaje interpretado por los navegadores web que en sí dentro de sus etiquetas solo contiene texto y para agregar archivos multimedia lo hace utilizando un sistema de referencias. Debido a los diferentes cambios que presenta HTML agregando y suprimiendo funcionalidades a través de sus diferentes versiones en las que se busca siempre mejorar y hacer más eficiente la creación de páginas web se tiene que para que los

navegadores web ejecuten correctamente las páginas creadas en las últimas versiones de HTML estos también deberán estar actualizados a sus últimas versiones. [12]

- **JavaScript:** Es un lenguaje interpretado, lo que quiere decir que son los navegadores los encargados de procesar e interpretar sus implementaciones. También tiene similitudes con lenguajes tales como Java y C, pero a diferencia de estos dos, JavaScript no es un lenguaje orientado a objetos, sino basado en prototipos. Es poco probable encontrar una aplicación web implementada netamente en JavaScript, regularmente las aplicaciones de JavaScript se encuentran en el cuerpo de páginas web creadas usando HTML y CSS para agregar dinamismo e interactividad a ellas, por ejemplo, para acceder a imágenes, chequear formularios en tiempo real, hacer validaciones de todo tipo, entre otras. En la actualidad se crearon unos motores para optimizar el tiempo de interpretación de JavaScript, la clave de estos es transformar código JavaScript en código máquina para que los tiempos de ejecución sean similares a los de aplicaciones de escritorio. [12]
- **CSS:** CSS es el lenguaje que se utiliza en las aplicaciones web para describir la presentación de documentos HTML o XML, en otras palabras mediante CSS se describe cómo se deben interpretar los atributos de estos documentos. Su uso está ligado a establecer el diseño de documentos HTML, XML o cualquier otro de este tipo. La corporación encargada de mantener CSS es la W3C. CSS se centra en establecer parámetros de diseño para páginas e interfaces de usuario, en cuanto a separación del contenido y la forma de presentar este, mediante algunas características como lo son los layouts o capas, la fuente de la letra, su tamaño, los colores, fondos, entre otros. Aunque no solo en el ámbito visual gracias a las hojas de estilo auditivas. La unión entre HTML, JavaScript y CSS permiten la creación de sitios web con interfaces atractivas visualmente, lo que hace que esta tecnología sea muy usada actualmente. [12]

2.2.5 Frameworks

Este término se refiere a una estructura de software predeterminada la cual está compuesta por un grupo de componentes personalizables e intercambiables los cuales

sirven como plantilla y facilitan la creación de aplicaciones. Se puede hacer referencia a un *framework* como un programa genérico incompleto el cual se puede configurar de tal manera que se logren resultados específicos. Existen diferentes tipos de *frameworks* y por lo general cada lenguaje de programación tiene uno o varios de estos. A continuación, se darán a conocer unos de los más populares y sus características. [11]

- **Zend (PHP):** Zend es un *framework* de código abierto el cual está diseñado para facilitar la creación de aplicaciones web, está basado en el uso de servicios de PHP, su implementación utiliza en un 100% el paradigma orientado a objetos y a su vez un bajo acoplamiento entre componentes para así facilitar el uso de cada uno de estos componentes por separado. [7]
- **Openxava (JAVA):** Openxava es un *framework* basado en estándares java para el desarrollo de aplicaciones empresariales de forma rápida que presenta como principales características su alta productividad, es decir, no requiere que los desarrolladores escriban interfaces de usuario o conectividad en bases de datos, estas se proveen automáticamente al escribir la lógica del negocio y la estructura de los datos. [7]
- **Django (Python)** es un *framework* web de alto nivel que permite el desarrollo rápido de sitios web seguros y de fácil mantenimiento. Django se encarga de gran parte de las complicaciones del desarrollo web; es gratuito y de código abierto, tiene una comunidad próspera y activa además de una gran documentación. Proporciona una serie de características que facilitan el desarrollo rápido de páginas orientadas a contenidos. Por ejemplo, en lugar de requerir que los desarrolladores escriban controladores y vistas para las áreas de administración de la página, Django proporciona una aplicación incorporada para administrar los contenidos, que puede incluirse como parte de cualquier página hecha con Django y que puede administrar varias páginas hechas con Django a partir de una misma instalación; la aplicación administrativa permite la creación, actualización y eliminación de objetos de contenido, llevando un registro de todas las acciones realizadas sobre cada uno, proporciona una interfaz para administrar los usuarios y los grupos de usuarios.

Adicionalmente, incluye un servidor web liviano para realizar pruebas y trabajar en la etapa de desarrollo. [7]

- **Flask (Python):** Flask es un marco web y un módulo de Python que permite desarrollar aplicaciones web fácilmente, tiene un núcleo pequeño y fácil de ampliar. Es un micro *framework* que no incluye un mapeo objeto-relacional (ORM - *Object-Relational Mapping*) o características similares. Tiene característica como enrutamiento de URL, entre otras. Es un marco de aplicación web *WSGI*. Flask es un marco de aplicación web escrito en Python. Fue desarrollado por Armin Ronacher, quien dirigió un equipo de entusiastas internacionales de Python llamado Pocco. Flask se basa en el kit de herramientas *Werkzeug WSGI* y el motor de plantillas Jinja2, ambos proyectos de Pocco. [13]

2.2.6 Editores de texto

Para empezar a programar en los distintos lenguajes de programación, es necesario instalar un segundo programa en el equipo. Se hace referencia a un editor de código fuente, también conocido como Entorno de Desarrollo Integrado (IDE, *Integrated Development Environment*). Se trata de una herramienta diseñada para editar el código fuente de diversos lenguajes de programación como Python. Existen muchos editores que pueden utilizarse en todas las plataformas, en este caso se usará Visual Studio Code.

- **Visual Studio Code:** Es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. Visual Studio Code es un editor de código fuente compatible con varios lenguajes de programación y un conjunto de características que pueden o no estar disponibles para un lenguaje dado. [14]

2.2.7 Sistema SCADA

Un SCADA es un Sistema de Control de Supervisión y de Adquisición de Datos, es una aplicación software de control (ejecutada en una estación de trabajo) que intercambia información con los elementos de campo y que supervisa y/o controla el proceso de forma

automática, y que permite en tiempo real generar información para diversos usuarios como operadores, supervisores, mantenimiento, gestores, administradores, entre otros.

Un SCADA es una aplicación de software especialmente diseñada para proveer comunicación con los dispositivos de campo y controlar el proceso de forma automática desde la pantalla de una estación de operación. Además, proporciona toda la información que se genera en el proceso productivo a diversos usuarios, tanto del mismo nivel como de otros supervisores dentro de la empresa: control de calidad, supervisión y mantenimiento, entre otros.

Los sistemas SCADA están diseñados para funcionar sobre ordenadores de control de producción, con acceso a la planta mediante la comunicación digital con instrumentos y actuadores, e interfaz gráfica de alto nivel para el operador (pantallas táctiles, ratones o cursores, lápices ópticos, entre otros.).

Aunque inicialmente solo era un programa que permitía la supervisión y adquisición de datos en procesos de control, en los últimos tiempos ha surgido una serie de productos de hardware y buses especialmente diseñados o adaptados para este tipo de sistemas. La interconexión de los sistemas SCADA también es propia, y se realiza mediante una interfaz del PC a la planta centralizada, cerrando el lazo sobre el ordenador principal de supervisión. El sistema permite comunicarse con los dispositivos de campo (controladores autónomos, autómatas programables, sistemas de dosificación, entre otros.) para controlar el proceso en forma automática desde la pantalla del ordenador, que es configurada por el usuario y puede ser modificada con facilidad. Además, provee a diversos usuarios de toda la información que se genera en el proceso productivo. [17]

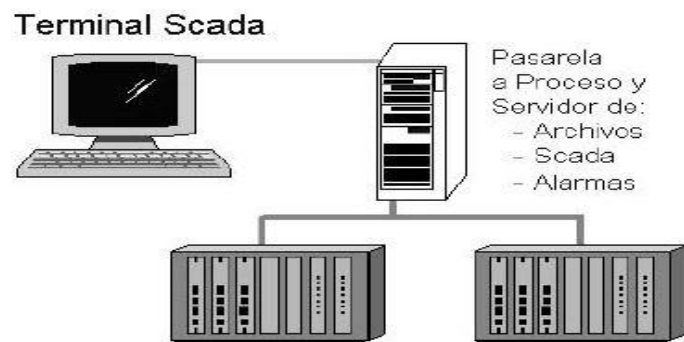


Figura N°2 Diagrama de un sistema SCADA escalable. Fuente [16]

2.2.8 Características de un sistema SCADA

- Puede adquirir, procesar y almacenar un conjunto enorme de datos para utilizar la información recibida de forma continua y confiable dentro del proceso productivo de la empresa.
- Pueden representar gráficamente todo el proceso productivo para controlar de primera mano las diferentes variables y monitorizarlas mediante alarmas.
- Ejecutan acciones de control mediante las cuales se puede modificar la evolución de todo el proceso industrial.
- Permite la ampliación y adaptación de todo el sistema gracias a que cuenta con una arquitectura abierta y flexible, que permite funcionar en base a las necesidades de cada cliente.
- Ofrece una conectividad total con otro tipo de aplicaciones industriales y bases de datos, ya sean de origen local o estén distribuidos en redes de comunicación.
- La supervisión se puede realizar en remoto, ya que mediante un sistema de pantallas los ingenieros encargados del mantenimiento y control de una serie de dispositivos, pueden llevar a cabo la monitorización sin problemas.
- Permite la explotación de los datos recabados en el día a día para mejorar la gestión de la calidad, el control estadístico y la gestión de la producción.
- Monitorizar y controlar las operaciones en tiempo real.
- Recolectar información histórica para hacer un análisis completo.
- Programar eventos y tareas automáticas. [15]

2.2.9 Requisitos de un SCADA

Los sistemas SCADA deben tener arquitecturas abiertas, capaces de crecer o adaptarse según las necesidades cambiantes de la empresa. Deben comunicarse con total facilidad y de forma transparente para el usuario con el equipo de planta (*drivers*) y con el resto de la empresa (acceso a redes locales y de gestión). Además los programas deben ser sencillos de instalar, sin excesivas exigencias, y fáciles de utilizar, con interfaces amables con el usuario (sonido, imágenes, pantallas táctiles, entre otros). [15]

2.2.10 Arquitectura general de un sistema SCADA

Los primeros ordenadores o computadores personales en el campo de la automatización localizaban todo el control en el PC y tendían progresivamente a la distribución del control en planta. De esta manera, el sistema queda dividido en tres bloques principales, los cuales son: software de adquisición de datos y control (SCADA), sistemas de adquisición y mando (sensores y actuadores), sistema de interconexión (comunicaciones).

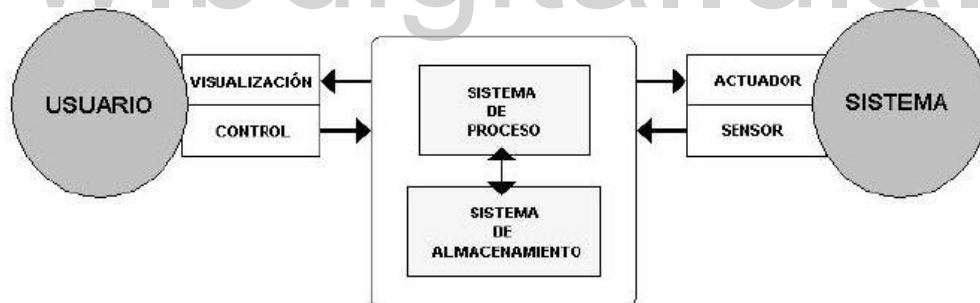


Figura N°3 Arquitectura básica de un sistema SCADA [16]

El usuario, mediante herramientas de visualización y control, tiene acceso al sistema de control de proceso, generalmente un ordenador donde reside la aplicación de control y supervisión (se trata de un sistema servidor). La comunicación entre estos dos sistemas se suele realizar a través de redes de comunicaciones corporativas (ethernet).

El sistema de proceso capta el estado del sistema a través de los elementos sensores e informa al usuario a través de las herramientas HMI. Basándose en los comandos

ejecutados por el usuario, el sistema de proceso inicia las acciones pertinentes para mantener el control del sistema a través de los elementos actuadores. La transmisión de los datos entre el sistema de proceso y los elementos de campo (sensores y actuadores) se lleva a cabo mediante los denominados buses de campo. La tendencia actual es englobar los sistemas de comunicación en una base común, como ethernet industrial. Toda la información generada durante la ejecución de las tareas de supervisión y control se almacena para disponer de los datos a posteriori. [16]

2.2.11 Componentes del hardware de un SCADA

Un sistema SCADA, como aplicación de software industrial, necesita ciertos componentes inherentes de hardware en su sistema para poder tratar y gestionar la información captada por los instrumentos de campo, a continuación se describen los componentes de hardware más importantes según Pérez López [15].

- **Ordenador central o MTU:** Éste se encarga de supervisar y recibe los datos de los equipos o instrumentos en campo que son enviados por las estaciones remotas (RTU). Procesa la información y envía comandos a las estaciones remotas para mantener las variables de los procesos dentro de los parámetros establecidos. Este ordenador suele ser un PC que soporta el HMI; de esto se deriva que el sistema SCADA más sencillo es el compuesto por un único ordenador, que es el MTU que supervisa toda la estación. La MTU se encarga de interrogar en forma periódica a las RTU y les transmite órdenes; siguiendo usualmente un esquema maestro-esclavo. Actúa como interfaz del operador, incluyendo la presentación de información de variables en tiempo real, la administración de alarmas y la recolección y presentación de información a manera de historial. [16]
- **Unidades remotas o RTU:** Están encargadas de controlar todas las señales de entrada y salida del campo como válvulas, equipos de medición, motores, entre otros. Además monitorean las condiciones de los dispositivos de campo y almacenan los estados de las alarmas. De igual manera envían los estados y alarmas de los equipos en campo y reciben comandos de la estación maestra. [16]

- **Red de comunicación:** Este es el nivel que gestiona la información que los instrumentos de campo envían a la red de ordenadores desde el sistema. El tipo de bus de datos utilizado en las comunicaciones puede ser muy variado según las necesidades del sistema y del software escogido para implementar el sistema SCADA, ya que no todos los tipos de software, ni los instrumentos de campo como PLC, pueden trabajar con todos los tipos de bus. A parte del tipo de bus, existen interfaces de comunicación especiales para la comunicación en un sistema SCADA, como pueden ser módems para estos sistemas que soportan los protocolos de comunicación SCADA y facilitan la implementación de la aplicación. [16]
- **Instrumentos de campo** Son todos aquellos equipos que permiten realizar tanto la automatización o control del sistema, como los que se encargan de la adquisición y recolección de información del sistema. Entre ellos se tienen los siguientes:
 - **PLC:** Son utilizados en el sistema como dispositivos de campo debido a que son más económicos, versátiles, flexibles y configurables que las RTU comentadas anteriormente.
 - **Sensores:** Son dispositivos que actúan como detectores de magnitudes físicas o químicas, denominadas variables de instrumentación, y las convierten en variables o señales eléctricas.
 - **Actuador:** Es un dispositivo mecánico que se utiliza para *actuar* u ofrecer movimiento sobre otro dispositivo mecánico. [16]

2.2.12 Estructura y componentes de un software SCADA

Según Pérez López [15], se contempla que los módulos o bloques de software que permiten las funciones de adquisición, supervisión y control en un SCADA son los siguientes.

- **Configuración:** Permite definir el entorno de trabajo de la aplicación según la disposición de pantallas requerida y los niveles de acceso para los distintos usuarios. En este módulo, el usuario define las pantallas gráficas o de texto que va a utilizar, importándolas desde otra aplicación o generándolas en el propio SCADA. Durante la configuración también se seleccionan los drivers de comunicación que permitirán el enlace con los elementos de campo y la conexión o no en red de estos últimos; se

selecciona el puerto de comunicación sobre el ordenador y sus parámetros. En algunos sistemas, también se indican las variables que se van a visualizar, procesar o controlar, en forma de lista o tabla en la que éstas pueden definirse y facilitar la programación posterior. [16]

- **Interfaz gráfica del operador o HMI:** Éste módulo proporciona al operador las funciones de control y supervisión de la planta de manera visual, de esta manera el proceso se representa mediante gráficos sinópticos almacenados en la MTU del SCADA. Los sinópticos están formados por un fondo fijo y varias zonas activas que cambian dinámicamente de formas y colores, según los valores leídos en la planta o en respuesta a las acciones del operador. Por lo general las pantallas deben tener apariencia consistente, con zonas diferenciadas para mostrar la planta (sinópticos), las botoneras y entradas de mando (control) y las salidas de mensajes del sistema (estados, alarmas, advertencias), la clasificación por colores ayuda a la comprensión rápida de la información; de igual manera los colores deben usarse de forma consistente en toda la aplicación: si rojo significa peligro o alarma y verde indica normalidad, estos serán sus significados en cualquier parte de la aplicación. [16]
- **Módulo de proceso:** Ejecuta las acciones de mando pre-programadas a partir de los valores actuales de variables leídas. Sobre cada pantalla se pueden programar relaciones entre variables del ordenador o del autómatas (PLC) que se ejecutan continuamente mientras esté activa. Es muy frecuente que el sistema SCADA confíe a los dispositivos de campo, principalmente autómatas, el trabajo de control directo de la planta, reservándose para sí las operaciones propias de la supervisión, como el control del proceso, análisis de tendencias o generación de históricos. [16]
- **Gestión y archivo de datos:** Se encarga del almacenamiento y procesado ordenado de los datos, según formatos inteligibles para elementos periféricos de hardware como impresoras, registradores, software como bases de datos u hojas de cálculo del sistema, de forma que otra aplicación o dispositivo pueda tener acceso a ellos. Además pueden seleccionarse datos de planta para ser capturados a intervalos periódicos y almacenados como un registro histórico de actividad, o para ser procesados inmediatamente por alguna aplicación de software para presentaciones estadísticas, análisis de calidad o mantenimiento; de esta manera una vez

procesados los datos, se presentan en forma de gráficas analógicas o histogramas, que permiten analizar la evolución global del proceso. [16]

En la figura N°4 se muestra un diagrama sobre la arquitectura básica de un sistema SCADA, detallando algunos de los componentes más importantes del sistema, tanto a nivel de software como a nivel de hardware; además de un operario que puede supervisar y manipular todas las acciones en el sistema.

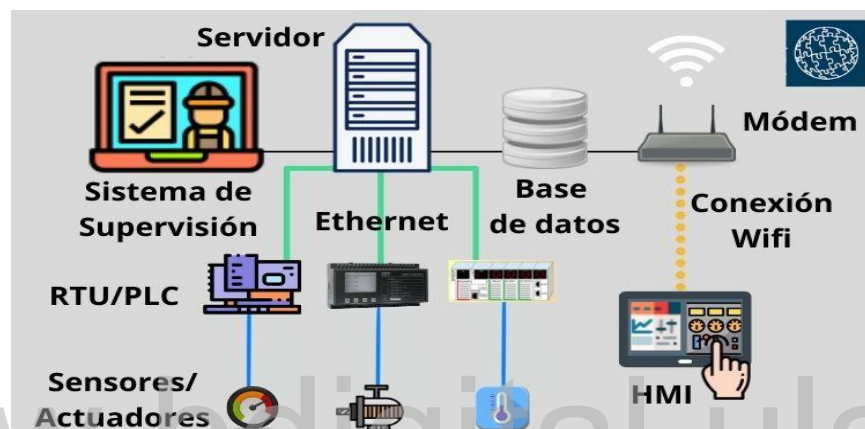


Figura N°4 Componentes básicos de un SCADA [20]

2.2.13 Protocolos de comunicación

En informática y telecomunicaciones, un protocolo de comunicaciones es un sistema de reglas que permiten que dos o más entidades de un sistema se comuniquen entre ellas para transmitir información a través de diferentes medios; éstos permiten posibles métodos de recuperación de errores. Los protocolos pueden ser implementados por hardware, o por software. También se define como un conjunto de normas que permite la comunicación entre ordenadores, estableciendo la forma de identificación de estos en la red, la forma de transmisión de los datos y la forma en que la información debe procesarse. Entre los protocolos de comunicación utilizados en sistemas SCADA se encuentran los siguientes: [3]

- **P-Net:** El protocolo P-NET para la conexión física usa el estándar RS-485 con transmisión asíncrona a 76.800 bps. Es una norma multiprotocolo y multired, es

decir, varios maestros (*másters*) pueden conectarse al mismo bus y varios buses pueden interconectarse formando una red mayor mediante pasarelas (*gateways*). La segmentación hace posible que cada segmento de bus tenga un tráfico local independiente, con lo que se incrementa el ancho de banda del sistema global. Hay 3 tipos de dispositivos que pueden ser conectados a una red P-NET: maestros (*másters*), esclavos (*slaves*) y pasarelas (*gateways*). Todas las comunicaciones están basadas en el principio de que un maestro envía una petición y la estación esclava direccionada devuelve una respuesta. [17]

- **Modbus:** El objetivo de este protocolo es lograr la transmisión de información entre distintos equipos electrónicos conectados a un mismo bus. Existiendo en dicho bus un solo dispositivo maestro y varios equipos esclavos conectados. En su origen estaba orientado a una conectividad a través de líneas serie como pueden ser RS-232 o RS-485, pero con el paso del tiempo han aparecido variantes como la Modbus TCP, que permite el encapsulamiento del Modbus serie en tramas Ethernet TCP/IP de forma sencilla. En Modbus existen 3 tipos de comunicaciones [17]
 - **ASCII (*American Standard Code for Information Interchange*).** El sistema de codificación es hexadecimal y cada carácter consta de 1 bit de inicio, 7 bits de codificación de los datos, 1 bit de paridad (opcional) y 1 o 2 bits de parada, o sea, un total de 9 a 11 bits por carácter. Se considera obsoleto en la actualidad [17]
 - **RTU (*Remote Terminal Unit*).** El sistema de codificación es binario y cada carácter consta de 1 bit de inicio, 8 bits de codificación de los datos, 1 bit de paridad (opcional) y 1 o 2 bits de parada, o sea, un total de 10 a 12 bits por carácter. Los dispositivos Modbus usan interfaces serie compatibles con RS-232C y RS-485, siendo el bus capaz de transferir datos a velocidades de 192 kbps y alcanzar distancias de 1 Km. [17]
 - **TCP/IP.** Es la estructura más importante, novedosa y más utilizada de Modbus, es un protocolo de comunicación basado en Ethernet para la técnica de automatización industrial. Modbus TCP corresponde a una implementación directa del establecido protocolo Modbus (RTU) en una comunicación basada en TCP/IP. Las aplicaciones Modbus existentes

pueden utilizar así muy fácilmente un intercambio de datos de procesos orientado a la conexión y protegido mediante una red Ethernet ya sea local o en la web existente. [17]

- **PROFINET:** Es un estándar Ethernet abierto que cumple la especificación IEC 61158 para la automatización industrial. Este tipo de red permite conectar equipos desde el nivel del campo (PLC y otros dispositivos) hasta el nivel de gestión (sistemas informáticos e internet), mediante redes ethernet y puertos TCP/IP. Además este protocolo permite una comunicación homogénea con la ingeniería cubriendo toda la planta industrial y de gestión apoyando las tecnologías de la información hasta el nivel del campo. [17]



Figura N°5 Puertos de comunicación utilizados en SCADA [27]

2.2.14 Automatización industrial

La automatización industrial es el uso de equipos eléctricos, electrónicos, electromecánicos (hardware) y software para controlar maquinarias que a su vez controlan procesos industriales. Es la rama de la ingeniería que abarca áreas como los sistemas de control, instrumentación industrial; que incluye los sensores, sistemas de transmisión, recolección de datos y las aplicaciones de software. La automatización puede definirse también como la utilización de tecnologías y sistemas de control automático que resultan en la operación y control de procesos industriales en forma automática, sin intervención

humana significativa, logrando un rendimiento superior al control manual. Estos dispositivos de automatización incluyen a los PLC [3]

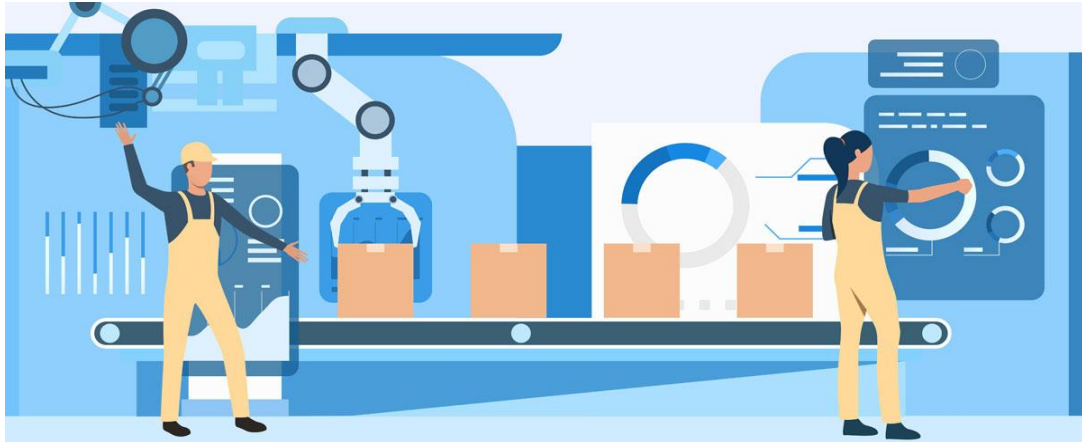


Figura N°6 Automatización industrial con un sistema SCADA. [26]

2.2.15 PLC

Es un dispositivo usado para controlar procesos. El control se realiza sobre la base de una lógica definida a través de un programa. Se define como un computador especializado utilizado para controlar máquinas y procesos. Utiliza una memoria programable para almacenar instrucciones y funciones específicas que incluyen control de encendido y apagado, temporizadores, conteo, secuenciador, manipulación aritmética y de información.

Surgió en los años 60 gracias a la aparición y posterior desarrollo del micro controlador. A medida que se aumentó la complejidad de los tableros de relés, los mismos se volvieron más grandes y costosos. Solo se podía utilizar una única lógica o trabajo específico, por lo que algún cambio de proceso implicaba cambio en el cableado o uso de más relés. Por lo tanto, el PLC se volvió una mejor opción ya que es compacto, ocupa mucho menos espacio que un tablero de relés, es flexible, fácil de programar y no hace falta volver a cablear.

El PLC es un equipo electrónico que activa salidas digitales (ceros y unos) y/o analógicas en función de las entradas que reciba, que pueden ser analógicas o digitales. Funciona como una interconexión de compuertas lógicas. En la industria se utilizan generalmente sensores

de nivel, temperatura, flujo y presión, interruptores de carrera, interruptores manuales. Las salidas del PLC generalmente se conectan a bobinas de contactores que a su vez controlan motores, calefactores, luminarias, entre otros. Presenta las siguientes ventajas:

- Flexible: Posibilidad de reemplazar la lógica cableada de un tablero o de un circuito impreso de un sistema electrónico, mediante un programa que corre en un PLC.
- Tiempos de respuesta más rápidos, ahorro de tiempo de trabajo, tanto en las conexiones a realizar, como en la puesta en marcha y en el ajuste del sistema.
- Menos cableados.
- Reducción de espacio.
- Estandarización.
- Diseño modular: Fácil de reparar y expandir.
- Manejo de sistemas complicados.
- Conjuntos de instrucciones sofisticadas disponibles.
- Permite realizar diagnósticos (fácil de reparar). [3]

2.2.16 Siemens S7-300

Es un controlador lógico programable multifuncional de la marca Siemens que se establece entre las gamas baja y media, con una amplia gama de módulos para una adaptación óptima a la tarea de automatización en múltiples procesos de la industria. Se provee de una aplicación flexible gracias a la posibilidad de realizar estructuras descentralizadas y conexiones por red; además tiene compatibilidad de conexión de distintos módulos adicionales de entradas y salidas, comunicaciones TCP/IP, módulos adaptadores RS-485/MPI/USB, o módulos para acoplarse con otros PLC.

Características de la CPU 312C:

- La CPU posee seguridad con gran memoria de programa y capacidad funcional para aplicaciones sofisticadas.
- Para configurar un sistema de automatización de seguridad positiva para instalaciones con altos requisitos de seguridad.

- Posee un paquete de 10 entradas y 8 salidas para 24 V en DC.
- 1 interfaz maestro/esclavo PROFIBUS DP y 1 interfaz DP maestro/esclavo/MPI.
- Ambos interfaces utilizables para la conexión de módulos de seguridad.
- Limitada a la comunicación serial, por lo tanto, sin tener algún módulo TCP/IP o MPI no se puede comunicar con otros PLC o computadoras que no cuenten con comunicación serial.
- Espacios para agregar más módulos de entradas y salidas; además de módulos compatibles con comunicación TCP/IP.
- Ideal para pequeñas aplicaciones que exigen gran capacidad de procesamiento. [18]



Figura N°7 PLC Siemens S7-300 [21]

2.2.17 Multi point interfaz

Por sus siglas en inglés *Multi Point Interfaz* o MPI, es una interfaz desarrollada por la empresa Siemens y su objetivo es la de comunicar múltiples PLC S7-300/400 y equipos de programación en una red de tipo multi punto donde puede haber más de dos equipos conectados. Para programar estos modelos de PLC se conectan a computadoras por medio de adaptadores MPI/USB.

Con este puerto se puede comunicar fácilmente a distancias reducidas sin requerir módulos adicionales. Se pueden enviar datos a 4 distintos aparatos al mismo tiempo y utilizando siempre el mismo puerto a una velocidad de 187,5 kbit cada segundo. Para pequeñas redes

de comunicación o pequeños volúmenes de datos la CPU ofrece el servicio de datos globales, que permiten intercambiar cíclicamente cantidades de datos en paquetes de hasta 22 bytes como máximo. [3]



Figura N°8 Adaptador MPI/USB para Siemens S7-300. [25]

2.2.18 Servidor OPC

Un servidor OPC (por sus siglas en inglés, *OLE for process control*), es una aplicación de software (*driver*) que cumple con una o más especificaciones definidas por la *OPC Foundation*. El Servidor OPC hace de interfaz comunicando por un lado con una o más fuentes de datos utilizando sus protocolo nativos (típicamente PLC, básculas, Módulos I/O, controladores, entre otros.) y por el otro lado con Clientes OPC (típicamente SCADAs, HMIs, generadores de informes, generadores de gráficos, aplicaciones de cálculos, entre otros.). En una arquitectura Cliente OPC/ Servidor OPC, el Servidor OPC es el esclavo mientras que el Cliente OPC es el maestro. Las comunicaciones entre el Cliente OPC y el Servidor OPC son bidireccionales, lo que significa que los Clientes pueden leer y escribir en los dispositivos a través del Servidor OPC. Existen cuatro tipos de servidores OPC definidos por la *OPC Foundation*, y son los siguientes:

- Servidor OPC DA – Basado en *Spezifikationsbasis: OPC Data Access* - especialmente diseñado para la transmisión de datos en tiempo real.
- Servidor OPC HDA – Basado en la especificación de acceso a datos historizados que provee al Cliente OPC HDA de datos históricos.

- Servidor OPC A&E *Server* – Basado en la especificación de Alarmas y Eventos – transfiere alarmas y eventos desde el dispositivo hacia el cliente OPC A&E.
- Servidor OPC UA – Basado en la especificación de Arquitectura Unificada – basado en el set más nuevo y avanzado de la *OPC Foundation*, permite a los Servidores OPC trabajar con cualquier tipo de datos. [19]

Adicionalmente el servidor OPC es, con mucha diferencia, la tecnología de comunicación industrial estándar. Permite el intercambio de información entre múltiples dispositivos y aplicaciones de control sin restricciones o límites impuestos por los fabricantes. Un servidor OPC puede estar comunicándose continuamente y en tiempo real con los PLC de campo, RTU, estaciones HMI u otras aplicaciones que lo requieran. Aunque el hardware y el software provengan de diferentes marcas comerciales, el cumplimiento del estándar OPC posibilita la comunicación continua en tiempo real. [28]



Figura N°9 Servidor OPC [19]

CAPÍTULO III

SELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN Y DEFINICIÓN DE MÓDULOS DEL SCADA

Este capítulo detalla un análisis de las herramientas de programación bajo software libre a utilizar en su etapa *backend* y en su etapa *frontend*, así como también el editor de texto idóneo para trabajar el sistema SCADA, teniendo en consideración una comparación previa detallada en el capítulo anterior, de manera que se pueda aprovechar al máximo cada una de sus características. De igual manera se emplea un análisis y definición de los módulos necesarios que tendrá el SCADA.

3.1 Selección del lenguaje de programación

Para la selección del lenguaje de programación *backend*, se estudiaron las herramientas disponibles en la red para poder hacer una comparación en base a las ventajas y desventajas de cada una, por lo tanto se presenta la tabla N°1 donde se destacan los aspectos más importantes a tener en cuenta al momento de seleccionar el más idóneo para la creación del SCADA y las necesidades del mismo.

Tabla N°1 Comparación de lenguajes de programación *backend*.

Lenguaje de programación	Características	Limitaciones
Python	<ul style="list-style-type: none">- Alta gama de librerías.- Fácil aprendizaje.- Código abierto.- Alta seguridad.- Programación orientada a objetos.- Código reutilizable.- Facilidad para aplicaciones web mediante <i>framework</i>	<ul style="list-style-type: none">- Depende en gran medida de bibliotecas y trabajos de terceros.- Las variabilidades de las versiones pueden ocasionar errores en las versiones.- Menor velocidad con respecto a otros programas

Tabla N°1 Comparación de lenguajes de programación *backend* (continuación).

Lenguaje de programación	Características	Limitaciones
C++	<ul style="list-style-type: none"> - Lenguaje robusto - Programación orientada a objetos. - Más rápido que la mayoría de los lenguajes de programación 	<ul style="list-style-type: none"> - Curva de aprendizaje alta - Poco atractivo visualmente - No soporta creaciones de aplicaciones web
PHP	<ul style="list-style-type: none"> - Multipropósito y fácil uso - Orientado al desarrollo de aplicaciones web - Versátil, código abierto y económico - Seguridad incorporada 	<ul style="list-style-type: none"> - No puede competir con tecnologías de desarrollo modernas. - La influencia PHP como tecnología está disminuyendo
C Programming	<ul style="list-style-type: none"> - Lenguaje estructurado. - No depende de hardware. - Programación de nivel medio. 	<ul style="list-style-type: none"> - No tiene soporte para abstracción, ocultación de datos o herencia. - Ausencia de constructores y depuradores. - Ausencia de un asistente de manejo de excepciones.
Java	<ul style="list-style-type: none"> - Programación orientada a objetos. - Lenguaje estable y actualizado con regularidad. - Gestión automática de memoria. 	<ul style="list-style-type: none"> - Desde 2019 se necesita licencia comercial para desarrollos de alto nivel. - Problemas de rendimiento. - Escasez de soluciones para crear interfaces gráficas.
Visual Basic.NET	<ul style="list-style-type: none"> - Compatibilidad con las utilidades del Visual Basic clásico. - Aplicaciones robustas. - Amplia seguridad en el código. - Recolección de basura muy eficiente. 	<ul style="list-style-type: none"> - Obliga al uso del espacio de trabajo .NET Framework. - Estrechos derechos de propiedad que reducen las posibilidades de VB.NET fuera de SO Windows. - Mejorable gestión de algunos tipos de datos, como las matrices, que no se pueden inicializar al declararlas

Con base al análisis de los aspectos presentados en la tabla N°1 y a los trabajos realizados por Rojas, J. [4] y Montaña, A. [5] se concluyó que el lenguaje Python es el seleccionado para el diseño de todos los módulos del SCADA, ya que este lenguaje implementa el desarrollo de proyectos de manera robusta, con fácil conexión a las bases de datos y fácil enlace con los lenguajes de programación en la etapa *frontend* para una mejor experiencia al momento de mostrar la aplicación ante los clientes. Posteriormente, se toma en cuenta, que Python es un lenguaje en código abierto, el cual puede ser reproducido en cualquier sistema operativo sin tener restricciones ante limitaciones prestadas por algún dispositivo o computadora, manteniendo así el acceso total a su código fuente y a todos sus scripts generados. Cabe destacar que Python ofrece una programación orientada a objetos, lo que quiere decir que su código se puede reusar entre diferentes scripts de manera organizada y de esta forma se disminuyen considerablemente los errores.

De igual manera es un lenguaje fácil de leer y escalable, lo cual es primordial para realizar modificaciones y actualizaciones en un futuro, además de que es muy robusto para cualquier tipo de amenaza exterior, asimismo, tiene una comunidad en la red que desarrolla gran cantidad de librerías aportando ejemplos funcionales y utilizables para quien quiera utilizar este lenguaje y adaptarlo a sus necesidades de manera totalmente gratis; las cuales son características aceptables al momento de desarrollar los distintos módulos del sistema SCADA.

Tabla N°2 Características de Python.

Lenguaje de programación seleccionado	Características
Python	<ul style="list-style-type: none"> - Es gratis y libre. - Multiplataforma. - Lenguaje interpretado por una máquina virtual. - Lectura clara del código. - Multiparadigma. - Facilidad en su curva de aprendizaje.

Adicionalmente, se debe considerar las dependencias de Python más favorables para el desarrollo de una aplicación web, como son las que manejarán las bases de datos con las

tablas y entidades necesarias durante la programación, o las que generarán el entorno gráfico y de servidor del sistema SCADA. De esta manera, se considera utilizar la base de datos PostgreSQL, mientras que para desarrollar el entorno gráfico y servidor se utiliza el *framework* Django escrito en el lenguaje Python.

PostgreSQL, también conocido como Postgres, es el sistema de gestión de bases de datos seleccionado por Montaña, A. [5] en su tesis de grado al momento de diseñar el módulo de base de datos. Su uso es totalmente gratuito y libre, el desarrollo de PostgreSQL no es manejado por una empresa o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, además de que ofrece una gran cantidad de opciones avanzadas. Debido a que, en Postgres no se requiere usar bloqueos de lectura al realizar una transacción, lo que brinda una mayor escalabilidad y legibilidad en la programación. También tiene alta concurrencia, lo que permite que los clientes hagan búsquedas (sólo de lectura) en los servidores mientras están en modo de recuperación o espera. Así se pueden hacer tareas de mantenimiento o recuperación sin bloquear completamente el sistema, adicionalmente permite que mientras un proceso escribe en una tabla, otros procesos accedan a la misma tabla sin necesidad de bloqueos.

Tabla N°3 Características de PostgreSQL.

Sistema de bases de datos	Características
PostgreSQL	<ul style="list-style-type: none"> - Texto de largo ilimitado. - Integridad referencial. - Llaves primarias y foráneas. - Replicación asíncrona/síncrona. - Copias de seguridad, múltiples métodos de autenticación.

Por consiguiente, éste sistema SCADA tiene la finalidad de adaptarse a múltiples plataformas de sistema operativo, distintos dispositivos y distintos exploradores web, es por ello que se selecciona el *framework* Django para Python, por delante de Flask para Python, ya que presenta suficiente información y librerías disponibles para un mejor desarrollo del trabajo; este *framework* reproduce aplicaciones web sin limitantes sobre software o hardware del equipo en el que se inicie, pues no presenta una carga o una ocupación de

espacio en la memoria del dispositivo, ya que la aplicación web se abre mediante la conexión a una red local o una red de internet.

Tabla N°4 Comparación entre Django y Flask.

Framework	Características
Django	<ul style="list-style-type: none"> - Cuenta con sistema de autenticación de usuario. - Gran comunidad activa. - Documentación de acceso gratuito. - Trabaja bajo un patrón MVC que permite un desarrollo ágil y reutilizable. - Las API's son mejores y se pueden convertir fácilmente en páginas HTML. - Realiza solicitudes GET y POST fácilmente. - Utiliza ORM para asignar objetos a tablas de bases de datos.
Flask	<ul style="list-style-type: none"> - Usa opciones nativas para crear sistemas de autenticación. - Condicionado para solicitudes GET y POST. - Mayor velocidad con respecto a Django. - Tiene extensiones para agregar ORM, validación de formularios o manejo de cargas.

Algo similar ocurre con la selección del lenguaje *frontend*, se analizaron las herramientas disponibles en la red y mencionadas en el capítulo anterior, por lo tanto se realiza una comparación entre ellas para seleccionar la mejor, o las mejores para poder mostrar al usuario una interfaz visual lo suficientemente clara y específica del SCADA, en la tabla N°5 se muestra la comparación realizada.

Tabla N°5 Comparación de lenguajes *frontend*.

Lenguaje de programación	Características	Limitaciones
HTML	<ul style="list-style-type: none"> - Permite describir hipertexto. - Tiene un despliegue rápido. - Compatibilidad con cualquier navegador web. 	<ul style="list-style-type: none"> - El diseño es lento. - Lenguaje estático. - Las etiquetas son limitadas.

Tabla N°5 Comparación de lenguajes *frontend* (continuación).

Lenguaje de programación	Características	Limitaciones
CSS	<ul style="list-style-type: none"> - Permite el apilamiento de instrucciones para definir formatos específicos. - Es utilizable en todos los navegadores y plataformas. -Optimiza el funcionamiento de aplicaciones web. 	<ul style="list-style-type: none"> - Escasez de seguridad. - Lo que funciona con un navegador no siempre funciona con los demás. - Al realizar los cambios, hay algunos que no son visibles hasta reiniciar el proyecto.
JavaScript	<ul style="list-style-type: none"> - Lenguaje orientado a objetos. - Lenguaje del lado del cliente. - Lenguaje interpretado - Tipado débil o no tipado. 	<ul style="list-style-type: none"> - Tiende a introducir gran cantidad de fragmentos de código en aplicaciones web. - No es compatible en todos los navegadores web de manera uniforme.

Es por ello que, debido a la comparación realizada se decide que, para mostrar una interfaz agradable al usuario y de fácil codificación, se selecciona el lenguaje HTML para generar las plantillas donde se diseña la interfaz, ya que los exploradores web sólo pueden interpretar lenguaje etiquetado, además de la incorporación del lenguaje CSS quien permite generar los estilos necesarios para hacer de la interfaz un entorno amigable para el usuario. Adicionalmente el uso del lenguaje Java Script permite darle dinamismo a la web con la finalidad de ser lo más optimizada posible.

Por último, se selecciona el software con el que se escribirán las líneas de código del SCADA, es así como se destaca entre los editores de texto el Visual Studio Code y se tienen en cuenta las características ofrecidas por él, ya que representa ser el editor de texto con mayores ventajas y mayor versatilidad a la hora de adaptarse a los distintos lenguajes de programación seleccionados anteriormente, esto se debe a que brinda una gama de extensiones para cada uno de los lenguajes que se usarán y, además presenta atajos y ayudas rápidas al programador que son muy útiles y hacen ahorrar tiempo al usuario.

3.2 Definición de los módulos que contemplan el sistema SCADA

De acuerdo a Pérez, E [15], Rodríguez, A. [16], y teniendo en cuenta el hardware y software disponible en el Laboratorio de Control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes, se llegó a la conclusión de que los módulos mínimos necesarios para un buen funcionamiento del SCADA son: módulo sinóptico, módulo de base de datos, módulo de gráfico de tendencias, módulo de alarmas y eventos, módulo servidor y módulo de comunicación; en la figura N°10 se muestran los módulos que tendrá el sistema.

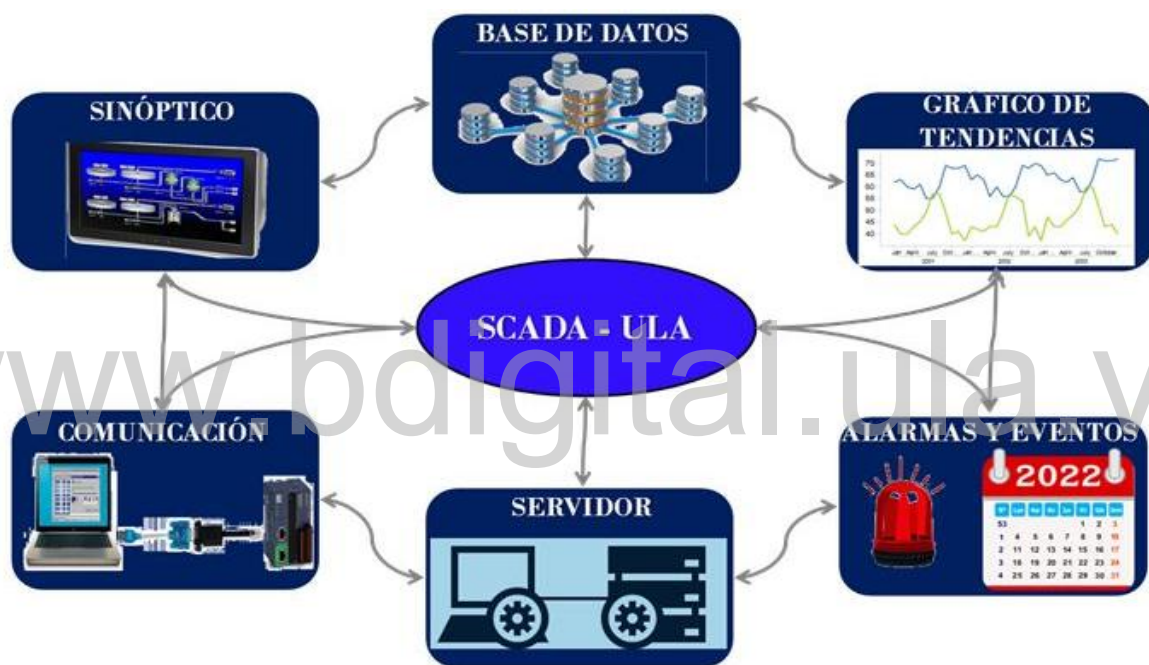


Figura N°10 Módulos del sistema SCADA – ULA. Fuente: autor.

3.2.1 Módulo sinóptico

Comprende las interfaces gráficas que permiten la elaboración de pantallas de usuario con múltiples combinaciones de imágenes y/o textos, definiendo así las funciones de control y supervisión de procesos. Gracias a las librerías de la programación orientada a objetos en Python es posible relacionar variables de sistema a objetos ya creados de forma muy sencilla. Se puede visualizar el estado de variables digitales mediante botones ON/OFF vinculados a actuadores de elementos conectados en un proceso como motores,

bandas transportadoras, disyuntores, sopladores, entre otros. Una vez en la pantalla, será posible editarlo y asignarle la variable a observar.

Permite definir el entorno de trabajo para adaptarlo a las necesidades de la aplicación:

- La estructura de pantallas se organiza de la forma más amigable al usuario, donde pueda ver los procesos de manera lineal (individual) o en árbol (conjunto).
- Los usuarios se clasifican según su importancia, creándose grupos con privilegios que permiten o limitan su influencia en el sistema. Herramientas de administración de usuarios, permiten una rápida estructuración de los permisos de acceso y utilización de la aplicación.
- Las pantallas de interface proporcionan una serie de herramientas que permiten realizar las tareas más comunes de forma rápida y sencilla. Las pantallas de alarmas se pueden organizar de manera distribuida (cada pantalla mostrará un grupo de alarmas) o centralizada (una pantalla única para todas las alarmas).

3.2.2 Módulo de alarma y eventos

Este módulo tiene la capacidad de reconocer eventos anormales dentro del proceso y reportar estos sucesos generando alarmas vinculadas a las variables de control en el sistema, las cuales están basadas en límites de control preestablecidos en cada variable medida durante el proceso. Además tiene la capacidad de registrar un historial de éstas alarmas y las acciones que son tomadas a cabo por el operador como respuesta a las mismas.

3.2.3 Módulo de base de datos

Este módulo se encarga del archivamiento y procesado ordenado de los datos, de forma que se pueda guardar toda la información del sistema para poder acceder a ella a través de otros dispositivos y hacer consultas cada vez que el sistema lo requiera, ya sea para comparar datos o modificarlos, también se pueden mostrar los archivos históricos de los datos del proceso que se encuentran almacenados, con el fin de optimización y corrección de procesos.

3.2.4 Módulo de gráfico de tendencias

Dentro de este módulo se encuentran los gráficos que permiten representar de forma agradable a la vista del usuario la evolución de variables de los procesos vinculados al sistema. Las utilidades más generales son:

- Representación en tiempo real de variables o recuperación de variables almacenadas de manera histórica.
- Visualización de valores.
- Desplazamiento a lo largo de todo el registro histórico.

3.2.5 Módulo servidor

Permite al sistema SCADA una funcionalidad total, ofreciendo su operatividad a través de cualquier navegador web estándar, gracias al *framework* de *django* se puede establecer el SCADA en un servidor local permitiendo que una computadora del laboratorio sea la MTU del SCADA, almacenando en ésta toda la información adquirida en los procesos y por ende en la base de datos. La información en tiempo real del proceso es inmediatamente accesible para cualquier persona o usuario autorizado. El trabajo en un entorno de red local es considerado normal para bastantes proveedores que incluyen funcionalidades de cliente y de servidor bajo aplicaciones web, además de que permite el uso de varias direcciones IP correspondientes a los dispositivos que están conectados a la red local.

3.2.6 Módulo de comunicación

El módulo de comunicación soporta el intercambio de información entre los elementos de planta, la arquitectura de hardware implementada y los elementos de gestión o enlace. Éste módulo comprende la parte del sistema que realizará el intercambio de información entre los elementos de campo como PLC y la computadora principal o MTU que realizará la recopilación de datos e información para guardar posteriormente en la base de datos. La conexión entre sistemas SCADA y PLC se realiza fundamentalmente a través de servidores OPC, que son traductores de datos definidos en el capítulo anterior; éstos

intérpretes de datos son muy importantes para que el sistema pueda monitorear y controlar las variables vinculadas al PLC.

Los servidores OPC también son conocidos como controladores genéricos, pues son controladores de tipo abierto independientes de los fabricantes de los PLC y SCADA. Están hechos en base a unas especificaciones concretas y de dominio público, cuya idea básica es definir una interface de comunicación estándar entre elementos de campo como PLC y aplicaciones como SCADA, partiendo de la declaración de las variables de cada proceso dentro del servidor OPC.

www.bdigital.ula.ve

CAPÍTULO IV

DESARROLLO DE MÓDULOS DEL SCADA

En el presente capítulo se detallará el diseño, el desarrollo y la implementación de los módulos servidor y comunicación del sistema SCADA a partir de su creación en Django, así como también las librerías disponibles en Python utilizadas para el perfeccionamiento de dichos módulos.

4.1 Desarrollo del módulo servidor

Para el desarrollo del módulo servidor se consideran 4 etapas fundamentales para llevar a cabo el mismo, éstas permiten crear el proyecto en Django, identificar la parte del código que permite realizar ajustes de servidor remoto mientras se hacen pruebas y cambios en el proyecto, servidor local al momento de terminar el proyecto, o servidor web para etapas de producción del mismo. En la figura N°11 se muestran dichas etapas



Figura N°11 Etapas del Módulo servidor. Fuente: autor.

El administrador de Django permite crear un proyecto a través del comando “*django-admin startproject NAME*”, donde el nombre del presente proyecto es “ScadaUla”, lo cual se realiza en cualquier terminal cmd, con esto se crea la carpeta que contiene toda la información del proyecto.

Una vez iniciado el proyecto se crea dentro de la carpeta raíz, un archivo llamado “*manage.py*” el cual es una versión específica de la librería “*Django-admin*” y le permite ejecutar comandos de administración en este proyecto, y también se crea otra carpeta con el mismo nombre del proyecto, la cual tiene toda la información de las bases de datos, vistas y *templates* del proyecto.

Posteriormente, para ejecutar el proyecto en un servidor de acceso remoto, se usa el comando “*python manage.py runserver*”; esto permite observar el proyecto desde su inicio hasta que se culmine, de manera que se lleve un seguimiento visual a los cambios que se realicen a través de las modificaciones en los scripts.

Luego en la carpeta que se creó el proyecto con el mismo nombre de éste, se consiguen varios archivos, como son: “*__init__.py*”, “*routing.py*”, “*asgi.py*”, “*urls.py*”, “*wsgi.py*”, “*settings.py*”; de esta manera para interés del servidor se debe hacer enfoque en el archivo “*settings.py*”; ya que en este script se maneja toda la configuración del proyecto SCADA, desde las aplicaciones instaladas, las librerías usadas, el motor de base de datos utilizado, los canales a usar, los paquetes de constructores de *django*, datos para el huso horario en el almacenaje de información en la base de datos, datos del panel de administración; información sobre los hosts a utilizar durante el desarrollo del proyecto hasta la finalización del mismo y su posterior etapa de producción.

Durante el desarrollo del SCADA, se mantiene la configuración de servidor remoto bajo los parámetros por defecto, ya que se usa un “local host: 127.0.0.1” y el puerto 8000; de esta manera cada vez que se inicie el servidor bajo el comando “*python manage.py runserver*”, se estará visualizando el proyecto a través de un explorador web bajo la dirección “127.0.0.1:8000”; pero con la limitante de que se puede observar sólo desde el ordenador de donde se está levantando el proyecto, debido a que se está usando un acceso remoto.

Posteriormente, se puede modificar la configuración de hosts permitidos, esto con el fin de poder visualizar el SCADA desde varios dispositivos conectados a una misma red local conformada por un módem o router de internet; para esto se debe acceder a la dirección IPv4 de la computadora conectada a la red a partir del comando *“ipconfig”* en una terminal de comandos cmd, el cual muestra la dirección IPv4 de la computadora. Aquí se puede verificar la dirección “IPv4” de la computadora que permitirá alojar el servidor local del SCADA; dicha dirección debe ser copiada en el archivo *“setting.py”*, en la sección de *“ALLOWED_HOSTS”* y en la sección *“CHANNELS_LAYERS”*.

A partir de allí, cada vez que se inicie el proyecto, se debe hacer bajo el comando *“python manage.py runserver (dirección IP de la computadora que se usará como servidor local):8000”*, se utiliza el mismo puerto 8000 que usa por defecto *django*, de esta manera se puede acceder desde cualquier otro dispositivo conectado a la red utilizando esa dirección web; éste procedimiento se realiza de ésta manera debido a que la máquina donde se encuentra el proyecto junto con la base de datos y todas sus tablas es la que se debe utilizar como servidor. En la figura N°12 se muestra un diagrama de flujo sobre el módulo de comunicación.

4.2 Desarrollo del módulo de comunicación

Para el módulo de comunicación se selecciona el servidor OPC IBH Softec, el cual está diseñado para trabajar específicamente con PLC de la marca Siemens; por ende, presenta compatibilidad con el PLC Siemens S7-300, especialmente con la CPU 312C que está disponible en el Laboratorio de Control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes; ya que permite comunicación con el autómatas a través del adaptador MPI, único adaptador disponible para comunicar al PLC con una computadora en el Laboratorio de Control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes; aunque se tiene la limitante de su uso temporizado por sesiones de 30 minutos.

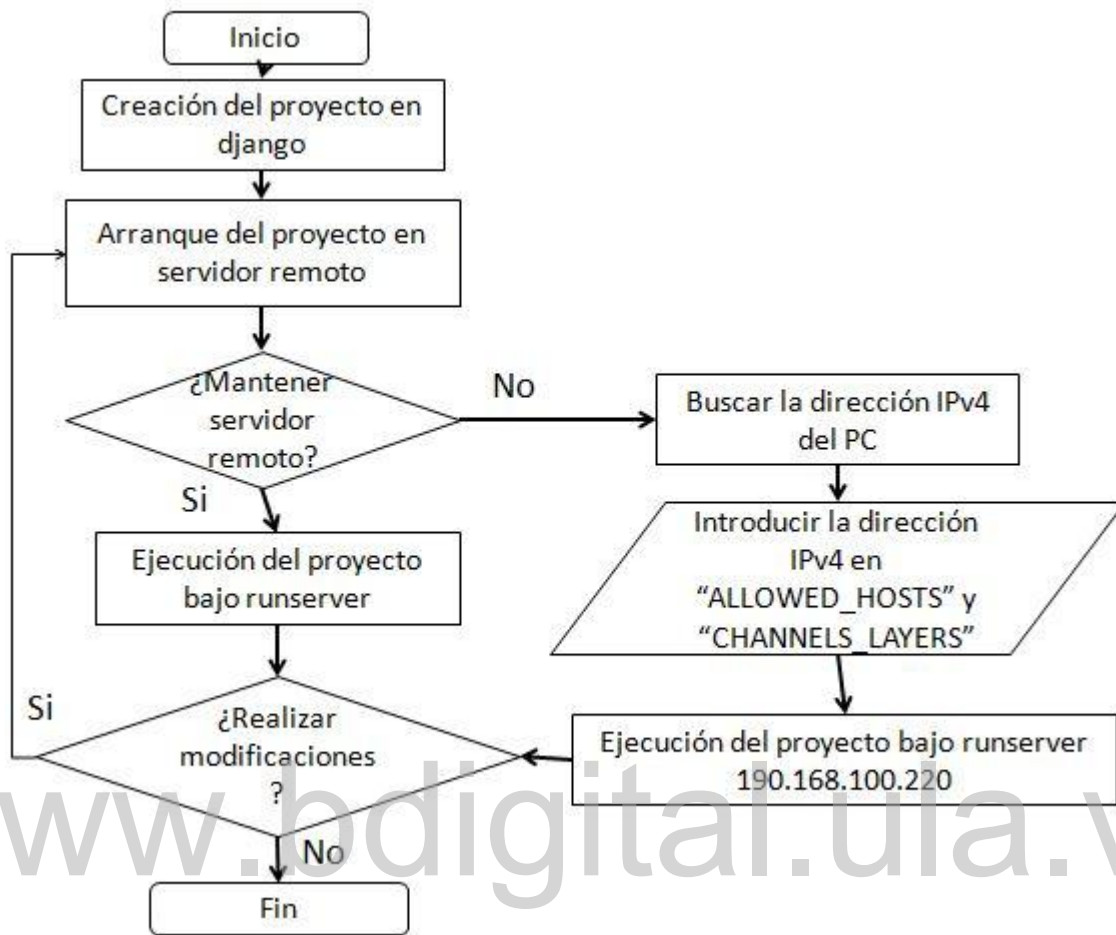


Figura N°12 Diagrama de flujo del módulo servidor. Fuente: autor.

Para el desarrollo de la comunicación entre el SCADA y el PLC, se destacan 7 etapas durante su diseño, en las que se presentan la programación del PLC, la conexión entre el PLC y el servidor OPC a través del “IBH OPC Editor”, la creación de todas las variables del PLC, la transferencia de éstas variables y sus direcciones al servidor OPC, la verificación de las variables desde un cliente OPC como “testopc”, la creación de un script de python que acceda a la información del servidor OPC y la manipulación de la misma a través de lecturas y escrituras; en la figura N°13 se muestran éstas etapas:

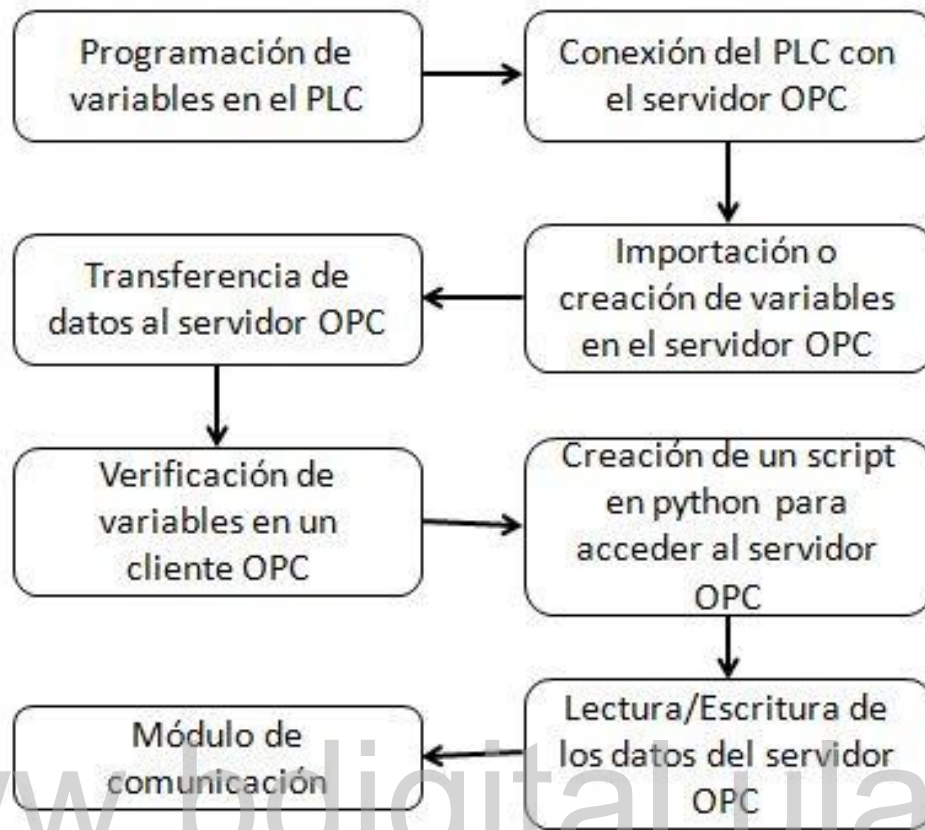


Figura N°13 Etapas del módulo de comunicación. Fuente: autor.

El progreso de éstas etapas se puede verificar y comprender a través de la figura N°14, la cual muestra un diagrama de flujo sobre la aplicación y las condiciones que se toman en cuenta durante el desarrollo del módulo de comunicación entre el SCADA y el PLC Siemens S7-300; en éste diagrama se bosqueja desde la programación de las variables en el PLC, la conexión del PLC con el servidor OPC, la transferencia de datos al servidor OPC para realizar consultas de las variables hasta culminar con la escritura o lectura de esas variables del PLC desde el servidor OPC.

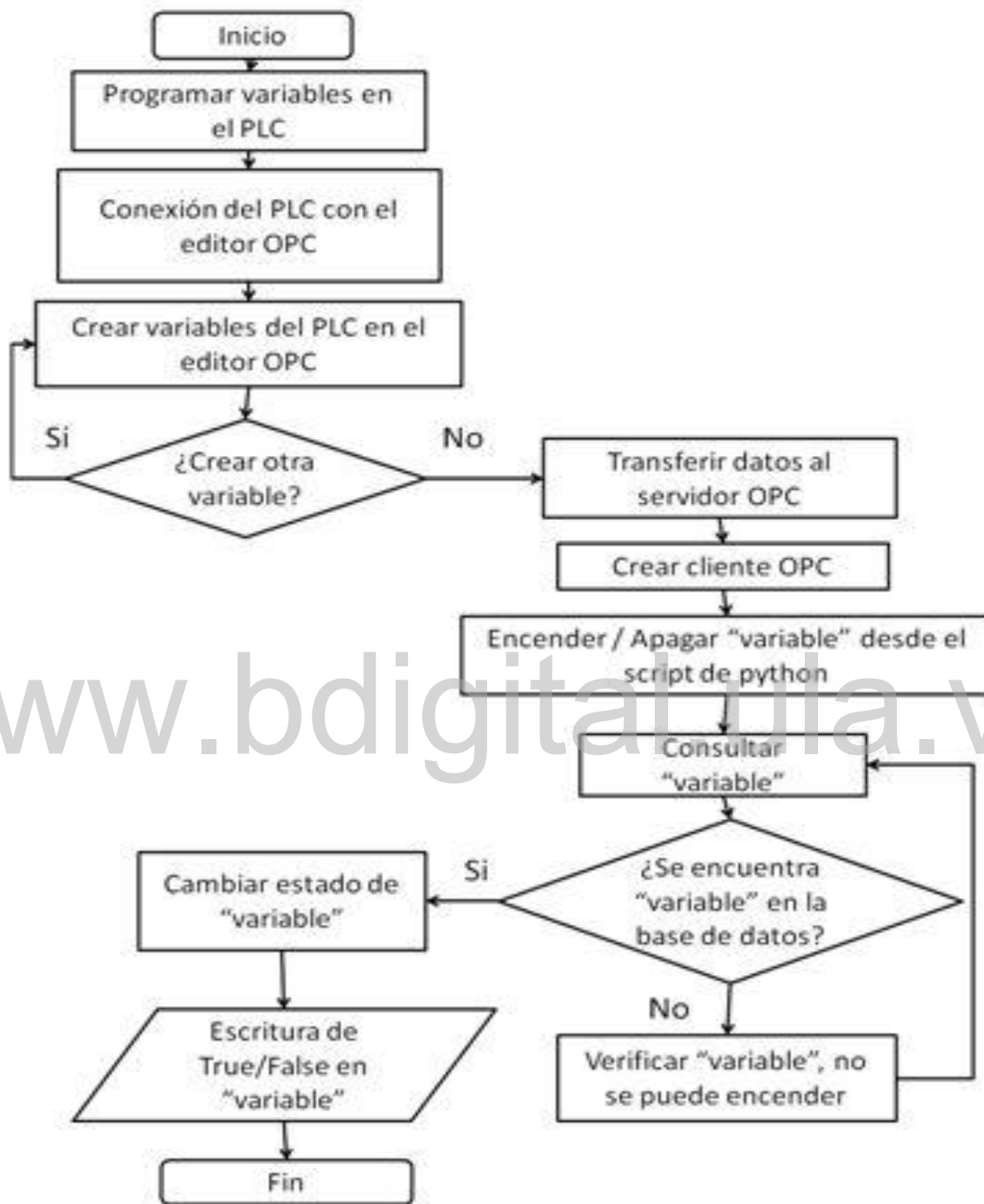


Figura N°14 Diagrama de flujo del módulo de comunicación. Fuente: autor.

Siguiendo el orden de las etapas anteriores se tiene que, primeramente se debe programar el PLC a través de su software Step7, donde se debe crear un programa, en éste caso es un diagrama en escalera para cada una de las variables de todos los procesos del SCADA, en

la figura N°15 se muestra la programación del PLC Siemens S7-300; esto permite encender y apagar las variables desde el SCADA.

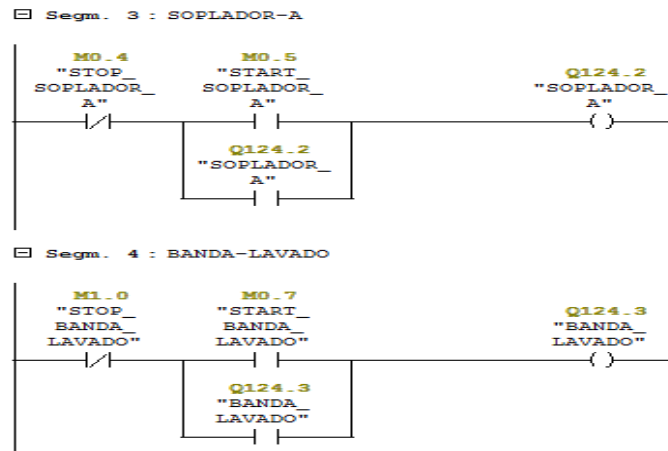


Figura N°15 Diagrama escalera para las variables en el Step7. Fuente: autor.

Posteriormente se ejecuta el OPC Editor, se abre el archivo “SCADA_IBH” y se establece conexión con el PLC a través de interface MPI, en la figura N°16 se muestra la conexión:

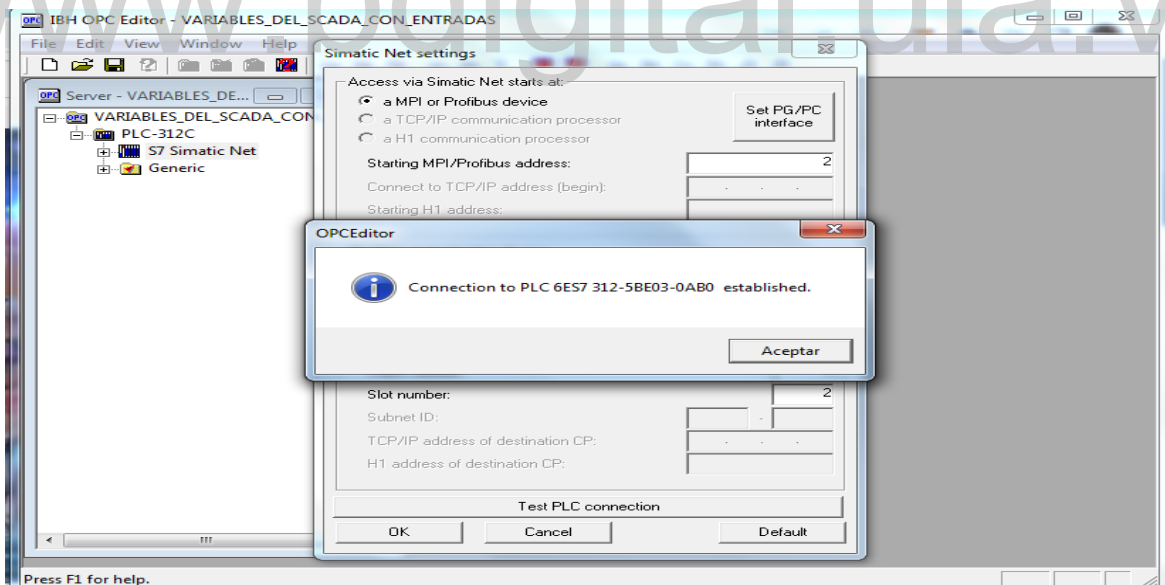


Figura N°16 Conexión entre el PLC y el Servidor IBH. Fuente: autor.

Luego se crean todas las variables del Step7 en el OPC Editor y se comprueba que el estado que se muestra en el OPC editor coincide con el estado de la variable en el PLC, la verificación del estado se realiza como se observa en la figura N°17:

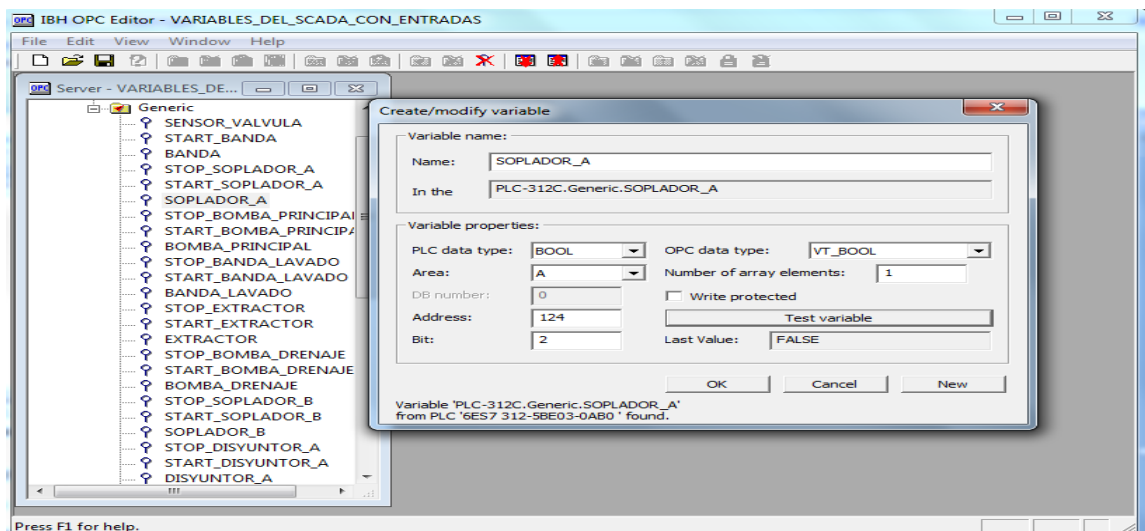


Figura N°17 Verificación de variables del PLC en el IBH. Fuente: autor.

Para que cualquier cliente OPC reconozca al servidor OPC es necesario transferir los datos al servidor OPC de la manera en que se indica en la figura N°18, seleccionando el servidor “IBHSoftec.IBHOPC.DA.1”. Este punto es de gran importancia, ya que de aquí en adelante se pueden manipular las variables del PLC a través del servidor OPC; además de aquí en adelante cualquier aplicación cliente puede reconocer al servidor IBH OPC server.

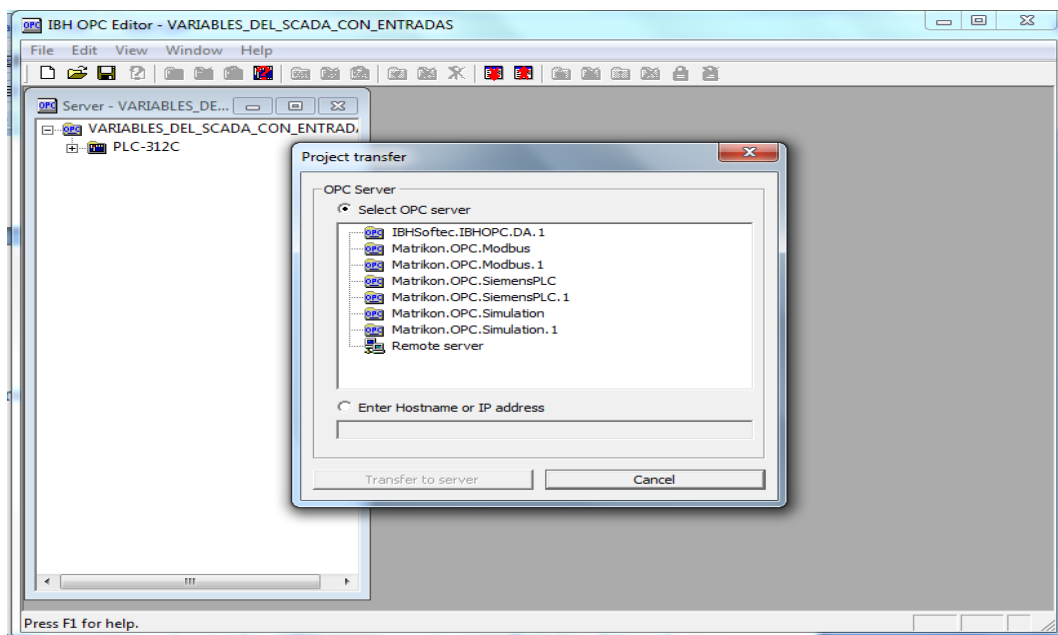
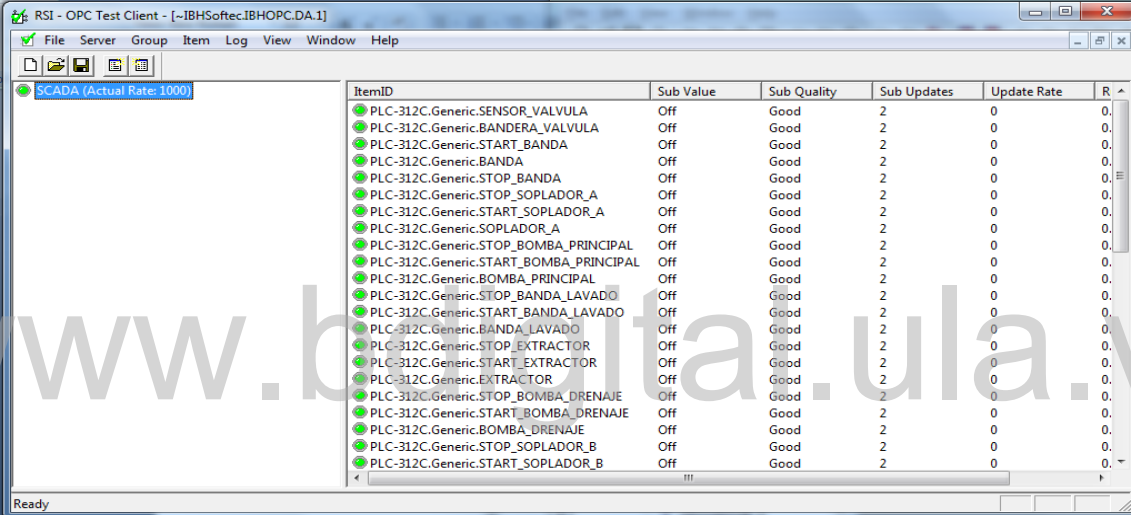


Figura N°18 Transferencia de datos al servidor IBH. Fuente: autor.

Seguidamente se necesita una aplicación cliente que mantenga la calidad de las entradas y salidas del PLC habilitadas como “Good” para realizar lecturas y escrituras desde el SCADA, para esto se ejecuta el archivo “Cliente_SCADA” en la aplicación portátil “testopc”, de aquí en adelante se da inicio al conteo de 30 minutos de duración del uso del servidor OPC debido a que es una versión *demo*; con este cliente simplemente se supervisa el estado de las variables del SCADA vinculadas al PLC, éste es un paso muy importante ya que sin la creación de éste cliente no se permite la comunicación entre el autómatas y el sistema SCADA. En la figura N°19 se muestra el cliente del “testopc”:



The screenshot shows the 'RSI - OPC Test Client' window with a table of data. The table has columns for ItemID, Sub Value, Sub Quality, Sub Updates, Update Rate, and R. The data rows list various PLC variables such as 'PLC-312C.Generic.SENSOR_VALVULA' and 'PLC-312C.Generic.BANDERA_VALVULA', all with a 'Sub Quality' of 'Good' and 'Update Rate' of 0.

ItemID	Sub Value	Sub Quality	Sub Updates	Update Rate	R
PLC-312C.Generic.SENSOR_VALVULA	Off	Good	2	0	0.
PLC-312C.Generic.BANDERA_VALVULA	Off	Good	2	0	0.
PLC-312C.Generic.START_BANDA	Off	Good	2	0	0.
PLC-312C.Generic.BANDA	Off	Good	2	0	0.
PLC-312C.Generic.STOP_BANDA	Off	Good	2	0	0.
PLC-312C.Generic.STOP_SOPLADOR_A	Off	Good	2	0	0.
PLC-312C.Generic.START_SOPLADOR_A	Off	Good	2	0	0.
PLC-312C.Generic.SOPLADOR_A	Off	Good	2	0	0.
PLC-312C.Generic.STOP_BOMBA_PRINCIPAL	Off	Good	2	0	0.
PLC-312C.Generic.START_BOMBA_PRINCIPAL	Off	Good	2	0	0.
PLC-312C.Generic.BOMBA_PRINCIPAL	Off	Good	2	0	0.
PLC-312C.Generic.STOP_BANDA_LAVADO	Off	Good	2	0	0.
PLC-312C.Generic.START_BANDA_LAVADO	Off	Good	2	0	0.
PLC-312C.Generic.BANDA_LAVADO	Off	Good	2	0	0.
PLC-312C.Generic.STOP_EXTRACTOR	Off	Good	2	0	0.
PLC-312C.Generic.START_EXTRACTOR	Off	Good	2	0	0.
PLC-312C.Generic.EXTRACTOR	Off	Good	2	0	0.
PLC-312C.Generic.STOP_BOMBA_DRENAJE	Off	Good	2	0	0.
PLC-312C.Generic.START_BOMBA_DRENAJE	Off	Good	2	0	0.
PLC-312C.Generic.BOMBA_DRENAJE	Off	Good	2	0	0.
PLC-312C.Generic.STOP_SOPLADOR_B	Off	Good	2	0	0.
PLC-312C.Generic.START_SOPLADOR_B	Off	Good	2	0	0.

Figura N°19 Cliente OPC en aplicación testopc. Fuente: autor.

4.2.1 Uso de la librería OpenOPC

Para el desarrollo del módulo de comunicación del SCADA, se hace uso de la librería OpenOPC disponible para Python, ésta es considerada una herramienta clave dentro del diseño del módulo, ya que permite generar un cliente OPC, acceder a los datos de cualquier servidor OPC y, realizar lecturas y escrituras en las variables que se encuentren dentro del servidor OPC; todo esto a partir de scripts de python, en pocas palabras traduce el Step7 y el IBH Softec, que son software de alto nivel, en scripts de bajo nivel.

OpenOPC para Python es un conjunto de herramientas OPC (OLE para control de procesos) gratuito y de código abierto diseñado para usarse con Python. Debido a que la

biblioteca OpenOPC implementa una cantidad mínima de funciones de Python que se pueden encadenar de diversas maneras, la biblioteca es fácil de aprender y recordar. En su forma más simple, puede leer y escribir uno o múltiples elementos OPC tan fácil como cualquier variable en su programa Python [24]

Posteriormente, se crea la lógica de programación mediante scripts de Python que se basan en el uso de la librería OpenOPC, fundamental en el desarrollo del módulo de comunicación; éstos scripts son llamados “servidor_ibh.py”, los cuales corresponden a los procesos del SCADA, en ellos se crea un cliente del servidor OPC, se accede a la información que está dentro del IBH softec y a partir de ahí se realizan las escrituras y lecturas de los estados de las variables del proceso en el que se encuentre el usuario; es por ello que acá radica la importancia de las escrituras realizadas, ya que estas permiten cambiar el estado de las variables del SCADA que están vinculadas a las salidas del PLC; en la figura N°20 se muestra un diagrama de flujo sobre el encendido de variables.

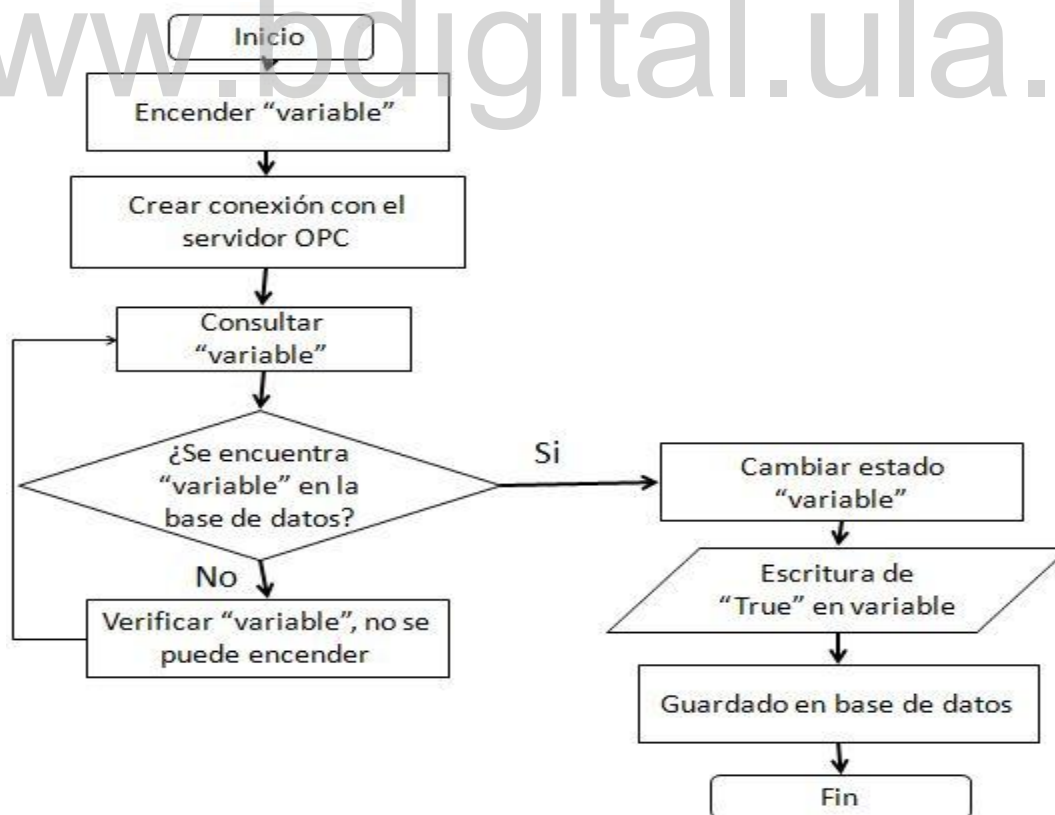


Figura N°20 Diagrama de flujo del encendido de las variables del PLC. Fuente: autor.

Desde el archivo “views.py” se realiza el llamado a la función que se encarga de la escritura de los valores de las variables, resaltando que dichos estados son cambiados por los botones “ON/OFF” respectivos a cada una de ellas, este proceso se indica en la figura N°21, a través de un diagrama de flujo sobre la escritura desde el archivo “views.py”:

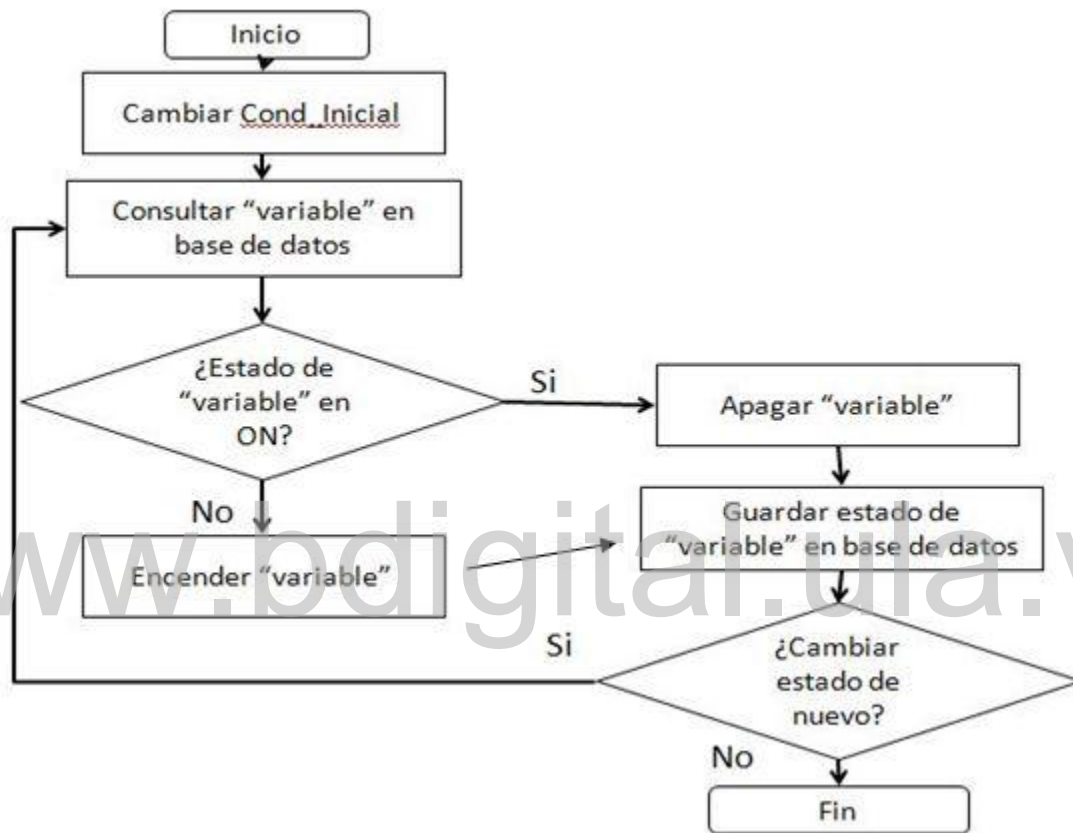


Figura N°21 Diagrama de flujo de la escritura del estado de las variables desde archivo “views.py”. Fuente: autor.

CAPÍTULO V

INTEGRACIÓN DE MÓDULOS Y ANÁLISIS DE RESULTADOS

En este capítulo se explicará cómo se realizó la integración de los módulos diseñados con los módulos existentes del SCADA en el laboratorio de control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes, resaltando que el módulo sinóptico ha sido diseñado en su trabajo de grado por Rojas, J. [4], mientras que los módulos base de datos, alarma y eventos, y gráfico de tendencias han sido diseñados en su trabajo de grado por Montaña, A. [5]. De igual manera se presenta un análisis de los resultados obtenidos vinculados al desarrollo de los módulos.

5.1 Integración del módulo servidor con el SCADA

Para realizar la integración del módulo servidor con el SCADA existente en el Laboratorio de Control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes, se deben cumplir una serie de etapas que ya se han demostrado en el capítulo anterior, dichas etapas se deben ejecutar en el orden específico para poder observar los módulos existentes desarrollados por Rojas, J. [4] y Montaña, A. [5] de manera correcta y que no surjan errores en la integración. En la figura N°22 se muestran las etapas que llevan a cabo la integración del servidor con el resto del sistema SCADA.

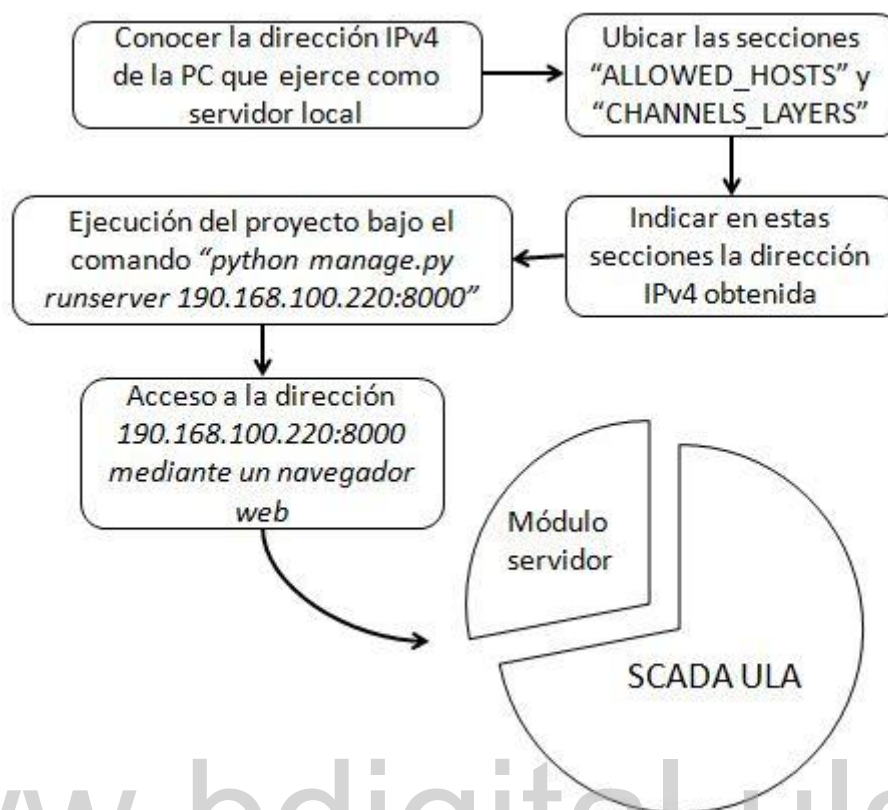


Figura N°22 Integración del módulo servidor con el SCADA. Fuente: autor.

Para ejecutar el servidor local en la computadora central o MTU del Laboratorio de Control se debe acceder a su dirección IPv4, a través del comando *“ipconfig”* en cualquier terminal de Windows como se estableció en el capítulo anterior, se tiene como resultado que la dirección IPv4 es *“190.168.100.220”*, sólo se debe indicar esa dirección en las secciones *“ALLOWED_HOSTS”* y *“CHANNELS_LAYERS”* del archivo *“settings.py”* del proyecto *“ScadaUla”*; con esto ya se permite abrir el proyecto bajo el comando *“python manage.py runserver 190.168.100.220:8000”* bajo la configuración de servidor local en cualquier explorador web.

Por otro lado, para lograr visualizar la interfaz en otros dispositivos con acceso a la red local será necesario anexar la dirección IP del *router* de igual manera en las secciones antes mencionadas del archivo *“settings.py”*, en este caso igualmente el puerto por el cual se ejecutará el tráfico de datos será el puerto 8000 que permite la comunicación web gracias al *framework django*. Por ejemplo se accede al comando *“python manage.py runserver*

190.168.100.107:8000", para observar el SCADA en otro dispositivo conectado a la red conformada por el router que se encuentra en el Laboratorio de Control.

5.2 Integración del módulo de comunicación con el SCADA

Para realizar la integración del módulo de comunicación con el PLC y el SCADA existente en el Laboratorio de Control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes, se deben cumplir una serie de etapas que ya se han demostrado en el capítulo anterior, dichas etapas se deben ejecutar en el orden específico para poder observar los módulos existentes desarrollados por Rojas, J. [4] y Montaña, A. [5] de manera correcta y que no surjan errores en la integración, esto es muy importante ya que se está trabajando con el servidor OPC tanto en alto nivel (software) como en bajo nivel (líneas de código).

La integración de éste módulo con el SCADA se refleja específicamente en el módulo sinóptico que es el encargado de mostrar al usuario cada cambio de estado en las variables; de igual manera mediante éste módulo se observan las gráficas históricas y en tiempo real de cada una de las variables que previamente han sido guardadas en la base de datos; en la figura N°23 se muestra la integración del módulo con el SCADA.

Hasta este punto ya se tiene una lectura y escritura satisfactoria de todas las variables del SCADA en el PLC, pero se deben establecer condiciones iniciales para evitar incoherencias entre el autómata y el sistema, pues se debe establecer que se separan las acciones de control para cada proceso en el que se está trabajando, para impedir problemas o confusiones al saber cuáles salidas del PLC están activas, para eso se debe hacer un apagado de las variables de los procesos que no se están ejecutando u observando, así como indicar que las condiciones iniciales para cada proceso son mantener todas las salidas en OFF, para poder encenderlas una por una y con un orden lógico según sea el proceso. Estas acciones se realizan en los archivos "views.py" de cada proceso y en la figura N°24 se muestra un diagrama de lo que hace el código.



Figura N°23 Integración del módulo de comunicación con el SCADA. Fuente: autor

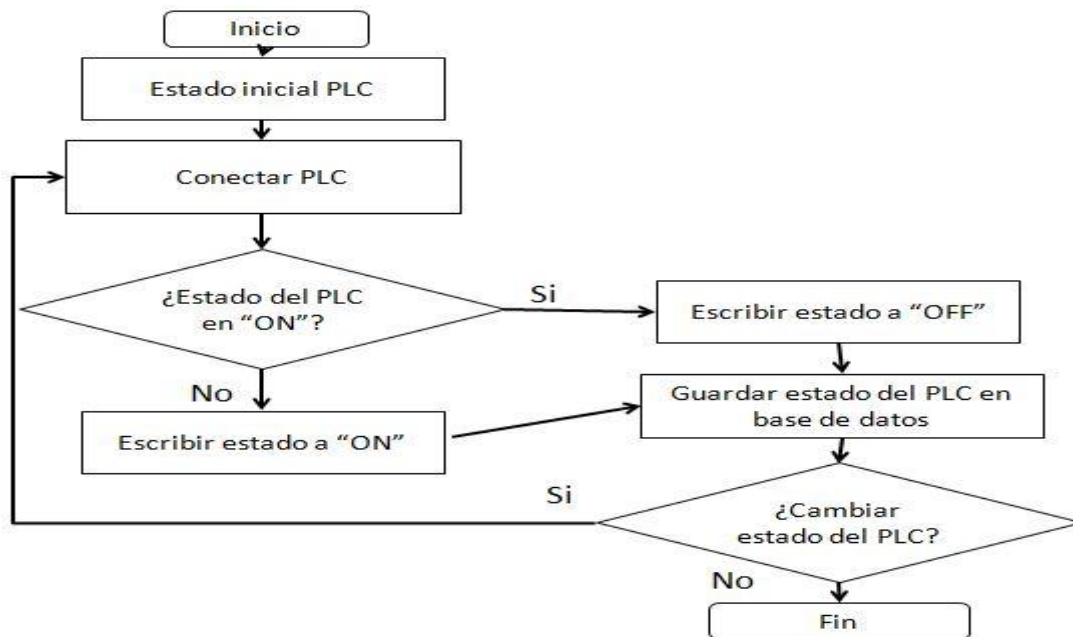


Figura N°24 Diagrama de flujo para establecer condiciones iniciales en cada proceso. Fuente: autor.

Por lo tanto ésta acción del algoritmo permite crear una conexión del PLC a cada proceso en el que se está trabajando y le indica al usuario a través de la interfaz visual que ya el PLC está conectado al proceso, además de indicarle las condiciones iniciales del mismo, en la figura N°25 se muestra lo descrito anteriormente.



Figura N°25 Condiciones iniciales en OFF vistas desde la interfaz del usuario. Fuente: autor.

Es necesario realizar este procedimiento ya que en el caso de reiniciar el SCADA, se accede a la última configuración que el sistema tenía, mientras que el PLC mantiene todas sus salidas apagadas ya que necesita la ejecución de acciones bajo código para encenderlas; esto se resuelve con el hecho de mantener en condiciones iniciales todas las variables apagadas, y así se mantiene una sincronía siempre entre el SCADA y el PLC, lo cual es óptimo para el buen funcionamiento del sistema.

Por otra parte, se establece un orden para realizar el encendido y apagado de las variables en cada proceso, con el fin de mantener una lógica dentro del SCADA y así acercarlo lo más posible a un entorno de procesos industriales.

De ésta manera se tiene que para el proceso 1, correspondiente al lavado de hojuelas en una fábrica de snacks, se debe considerar que el orden de encendido debe ser el siguiente:

1. BOMBA_PRINCIPAL
2. BANDA
3. SOPLADOR_A
4. BANDA_LAVADO
5. SOPLADOR_B
6. EXTRACTOR
7. BOMBA_DRENAJE

Con esto se consigue que para efectos de prácticas en el laboratorio, no se enciendan las variables de manera aleatoria, pues el orden de encendido está considerado para un proceso industrial no cíclico. De igual manera se tiene que para el apagado de las variables se debe usar el mismo orden, empezando por la “BOMBA_PRINCIPAL” para evitar el exceso de agua en el proceso durante el tiempo que se tarde en apagar las demás variables. En la figura N°26 se muestra un diagrama de flujo que describe lo que hace el código, de manera análoga se aplica para el resto de variables del proceso 1 y, en el código se consigue en el archivo “servidor_ibh.py”:

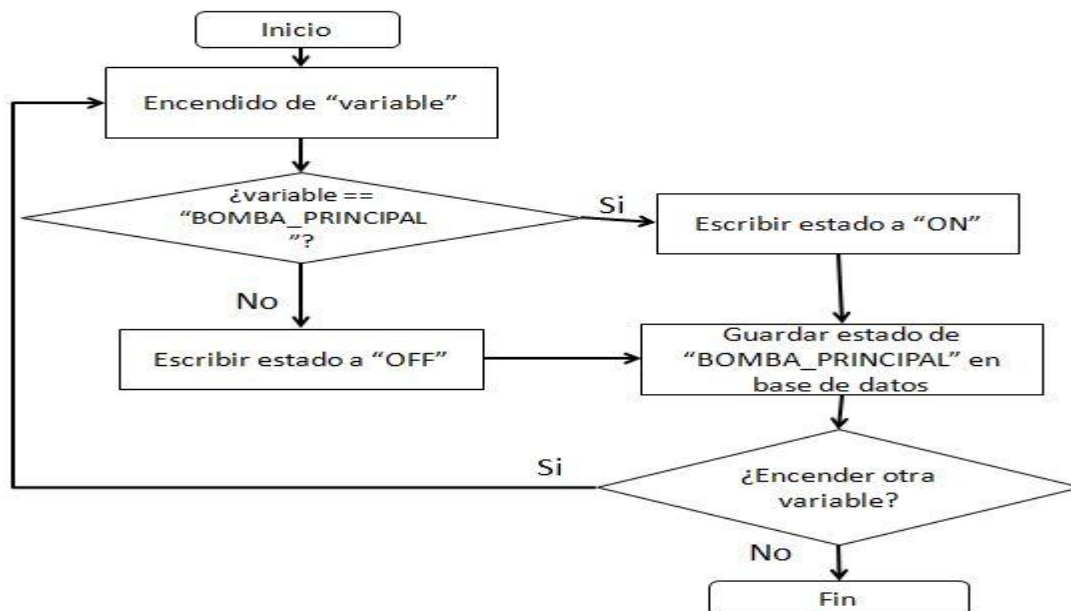


Figura N°26 Diagrama de flujo para encender las variables del proceso 1. Fuente: autor.

Para el proceso 2, correspondiente al p rtico de entrada de una Subestaci n el ctrica, se debe considerar que siempre se abre primero el "DISYUNTOR_A" o "DISYUNTOR_B" antes que el "SECCIONADOR_A" o "SECCIONADOR_B" (seg n sea el caso), pues si se hace de manera inversa se tienen problemas de diversa  ndole en la realidad; incluso ante la mayor a de fallas se abre s lo el disyuntor, pues el seccionador se abre s lo para realizar mantenimiento al sistema; adem s para el proceso del cierre o conexi n, se debe cerrar primero el seccionador y despu s el disyuntor; de igual manera para evitar problemas en la realidad. Todo esto se realiza mediante l gica de programaci n asumiendo las condiciones y estableciendo par metros; en la figura N 27 se muestra un diagrama de flujo alusivo a lo que se hace en el archivo "servidor_ibh.py" para el proceso 2.

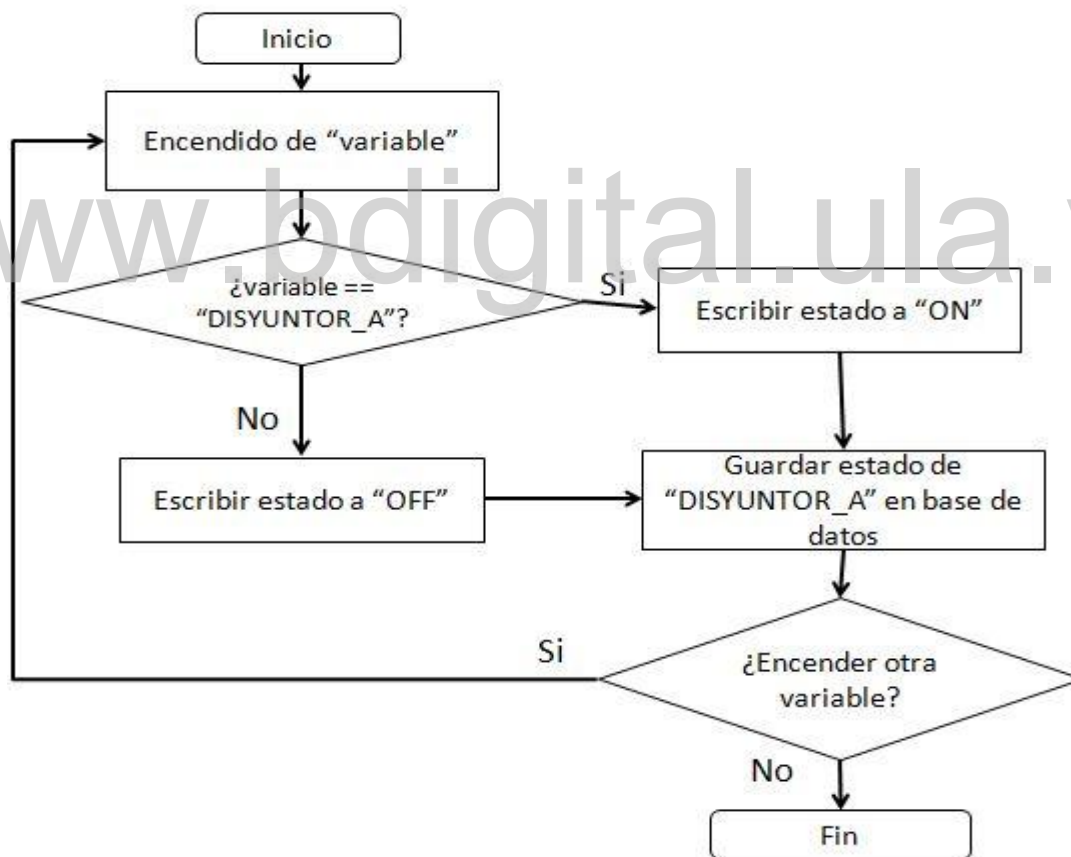


Figura N 27 Diagrama de flujo para encender las variables del proceso 2. Fuente: autor.

Para respetar el orden de encendido de variables en el proceso 2 se genera un mensaje de informaci n al usuario, en el que se indica por qu  no se puede encender las variables de

manera aleatoria; dicho mensaje es generado al usuario mediante el módulo sinóptico y en el que se comprueba una buena integración entre éste módulo y el de comunicaciones; en la figura N°28 se muestra la integración mencionada.

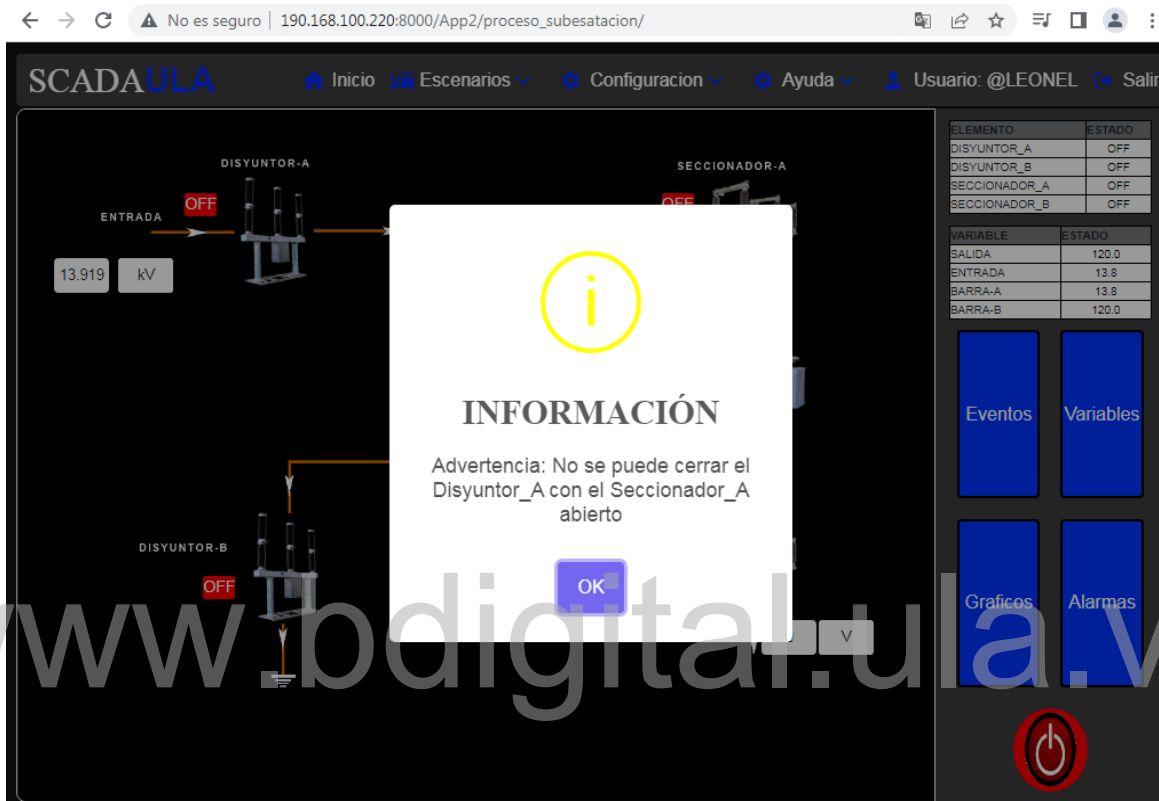


Figura N°28 Condiciones necesarias para encender el Disyuntor_A en el proceso 2. Fuente: autor.

Por último, lo correspondiente al proceso 3, del tueste de café, se debe considerar un orden de encendido de las variables asumiendo que el proceso industrial se realiza una sola vez y no es repetitivo, por lo tanto se tiene el siguiente orden de encendido:

1. MOTOR_TAMBOR
2. MOTOR_ENFRIADOR
3. EXTRACTOR

De igual manera se tiene el mismo orden para apagado debido a que se está asumiendo un proceso no cíclico, para realizar el encendido o apagado de las variables se muestra la figura N°29, en ella se realiza la consulta para la variable "MOTOR_TAMBOR", pero cabe

destacar que para el resto de las variables del proceso se realiza de manera análoga, esto se realiza desde el archivo “servidor_ibh.py, de igual manera en la figura N°30 se muestra la interfaz gráfica que observa el usuario para verificar el orden de encendido de las variables.

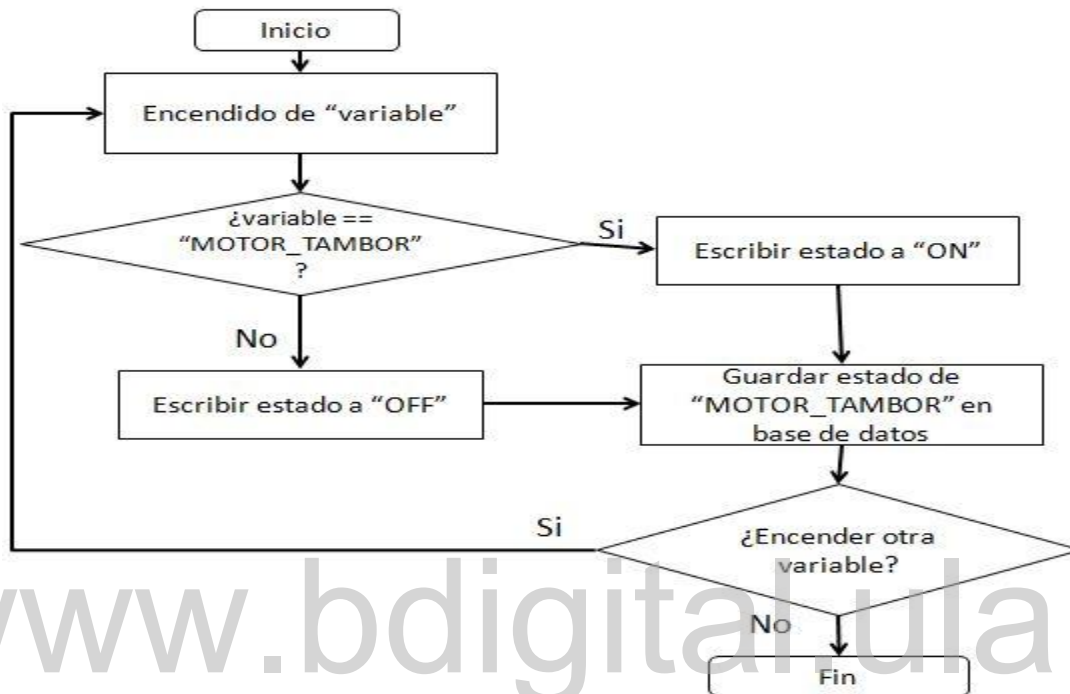


Figura N°29 Diagrama de flujo para encender las variables del proceso 3. Fuente: autor.



Figura N°30 Condiciones necesarias para apagar las variables del proceso 3. Fuente: autor.

5.3 Análisis de Resultados

Con los resultados obtenidos se considera que los módulos desarrollados son satisfactorios y compatibles con el SCADA establecido en el Laboratorio de Control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes; pues se ha realizado una integración exitosa de los módulos servidor y comunicación con los existentes, los cuales son: módulo sinóptico, base de datos, gráfico de tendencias, alarmas y eventos.

De igual manera se tiene que la integración de estos módulos con el SCADA ha sido efectiva debido a que cada cambio generado en las salidas del PLC se ven reflejados en las gráficas de cada proceso en tiempo real y en históricas; esto se hace desde el archivo “connet.ipynb”, en el cual se hace la lectura constante de todas las variables de cada proceso y las envía mediante la librería websocket hacia la base de datos para realizar un guardado de la información, en la figura N°31 se puede ver la muestra de una gráfica para la variable “BOMBA_PRINCIPAL”.

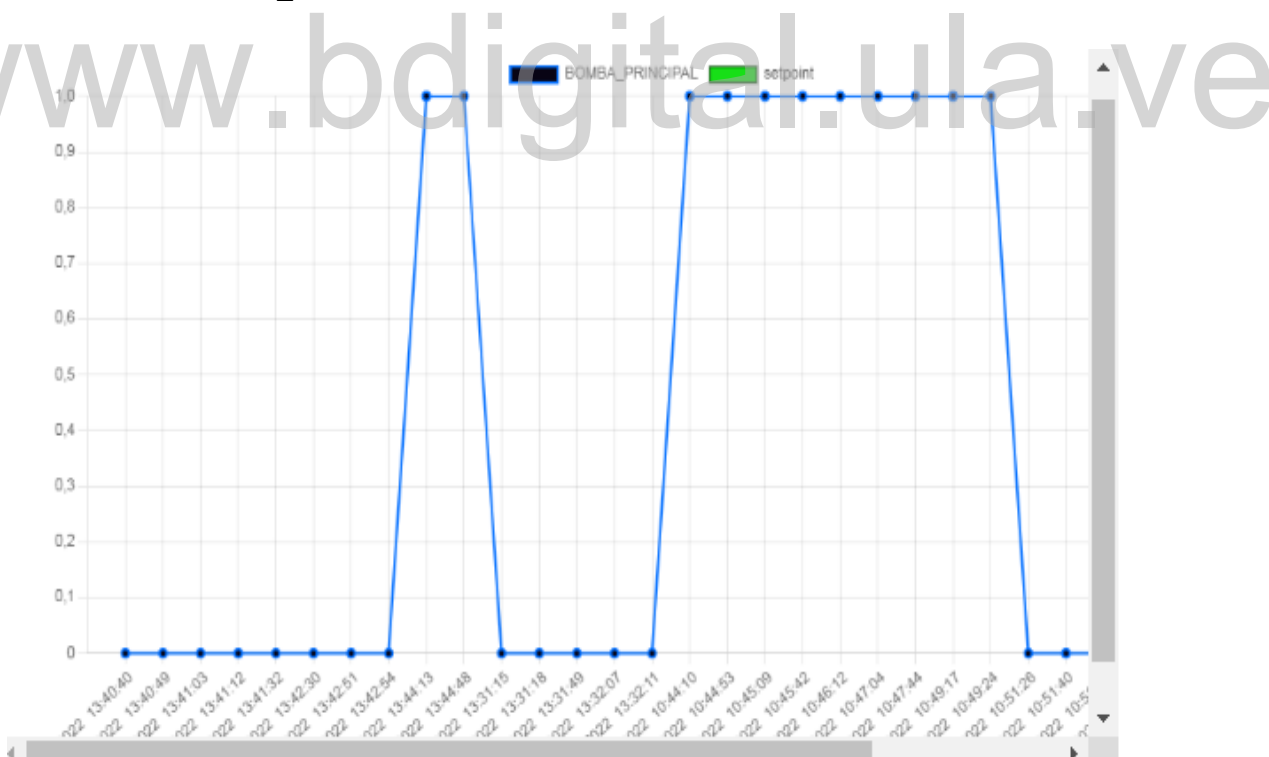


Figura N°31 Gráfica de datos históricos para la variable “BOMBA_PRINCIPAL”. Fuente: autor.

El sistema desarrollado utiliza el archivo “connet.ipynb” para hacer las lecturas continuas de las variables en cada proceso y así consultar en la base de datos cuando se genere alguna alarma según sea el caso, y en caso de que se genere la muestra en el HMI y se almacena en la base de datos; esto es muy importante porque se verifica que cada acción de control que se le aplique al proceso, ya sea cambio de estado en las variables o alguna lectura de las mismas, va a guardarse directamente en la base de datos gracias a la librería websocket. Respecto a lo mencionado anteriormente, en la figura N°32 se muestra un histórico de alarmas generadas en el proceso 1; de igual manera en la figura N°33 se muestra el HMI del proceso 2 mientras se hace la lectura continua de sus variables. Por último en la figura N°34 se muestra el SCADA funcionando en el laboratorio conectado con el PLC de manera óptima y compatible.

Fecha	Nombre	Clase	Valor	Limite	Tipo	Prioridad	Estado
03-06-2022 17:08:35	EXTRACTOR	DSC	1,0	1,0	DSC	3	Reconocida
03-06-2022 17:06:22	BANDA_LAVADO	DSC	0,0	0,0	DSC	3	Reconocida
03-06-2022 17:06:17	EXTRACTOR	DSC	0,0	0,0	DSC	3	Reconocida
03-06-2022 17:06:04	SOPLADOR_B	DSC	0,0	0,0	DSC	3	Reconocida
03-06-2022 15:48:52	SOPLADOR_B	DSC	1,0	1,0	DSC	3	Reconocida
03-06-2022 15:48:51	BANDA_LAVADO	DSC	1,0	1,0	DSC	3	Reconocida
03-06-2022 15:47:56	BANDA	DSC	0,0	0,0	DSC	3	Reconocida
03-06-2022 15:47:53	BANDA	DSC	0,0	0,0	DSC	3	Reconocida

Figura N°32 Alarmas históricas generadas en el proceso 3. Fuente: autor.

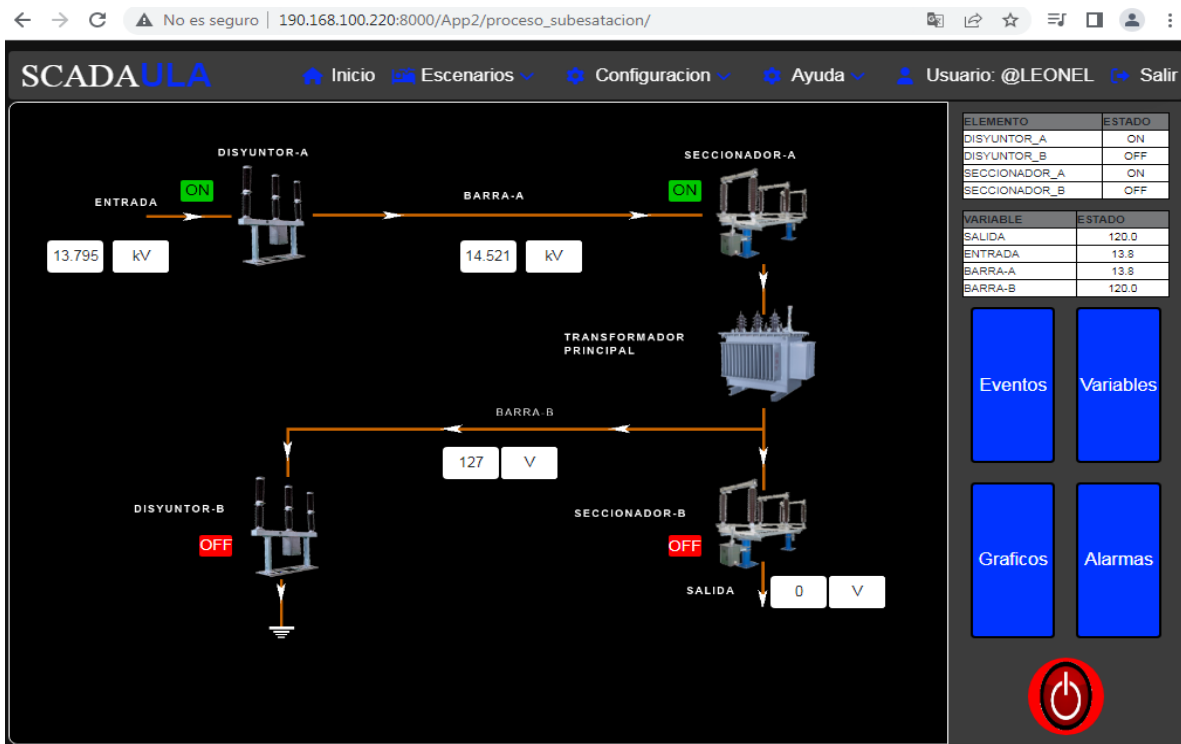


Figura N°33 HMI del proceso 2 mientras se realiza la lectura continua de las variables.

Fuente: autor.



Figura N°34 Funcionamiento del SCADA conectado al PLC. Fuente: autor.

CONCLUSIONES

Durante el desarrollo y diseño del sistema SCADA para el Laboratorio de Control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes, se llevó a cabo una investigación continua sobre el software libre más acorde a las necesidades del proyecto y la disponibilidad de hardware dentro de las instalaciones del laboratorio; teniendo esto en consideración, se seleccionó Python y su *framework Django*, el cual presenta facilidades para realizar aplicaciones web, pues muchas plataformas web están basadas en este *framework*, ya que está enfocado a la programación orientada a objetos, lo que quiere decir que el código se puede reutilizar de manera que se disminuyen las líneas de código y acciones a llevar a cabo, por ende, disminuye el tamaño del proyecto de manera considerable, y disminuyendo así las exigencias para el hardware donde se lleve a cabo.

El lenguaje de programación Python permitió cumplir de manera satisfactoria el objetivo de diseñar el módulo servidor, ya que para el desarrollo de éste se ha utilizado la librería del administrador de *Django*, el cual permite cambiar el host donde quedará alojado el sistema SCADA, permitiendo a través de un solo script la modificación del mismo. Se comprobó que es posible aplicar estos cambios desde un servidor remoto como lo es *localhost*, hasta una red local que amerita conectar la computadora a un router dentro del laboratorio.

Adicionalmente, se concluye que la herramienta de programación Python presta las suficientes alternativas para poder acoplar y usar de manera armónica las librerías existentes y disponibles en la red; ya que con ellas se lograron cumplir todos los objetivos planteados, y así obtener una integración total y exitosa de los módulos desarrollados con los demás módulos existentes del SCADA. La integración del sistema se basó en realizar lecturas desde el SCADA hacia los datos del PLC, que se almacenan de manera satisfactoria en la base de datos para ser consultados posteriormente según se requiera, mostrándose gráficas tanto en tiempo real como históricas de cada una de las variables. De

igual manera se realiza un monitoreo y se manipula a través del módulo sinóptico y del HMI del sistema, el estado de las variables que están vinculadas al PLC.

Por otra parte, para realizar una comunicación entre el PLC Siemens S7-300 y el sistema SCADA, se necesita un servidor OPC, que tiene la función de traducir los datos entre uno y otro de manera bidireccional y en tiempo real, además, el servidor OPC tiene la particularidad de que puede aplicar cambios en el PLC y en el sistema SCADA según se requiera. Debido a esto se concluye el servidor OPC es una herramienta de gran importancia para este tipo de aplicaciones, ya que con ella se logró cumplir el objetivo de diseñar el módulo de comunicación entre el PLC y el SCADA.

www.bdigital.ula.ve

RECOMENDACIONES

Es de gran importancia impartir al estudiantado el uso avanzado de programación digital para poder familiarizarse con éste tipo de investigaciones.

De igual manera se recomienda el mejoramiento del sistema SCADA de manera continua, para darle al sistema mayor robustez y funcionalidad para mayor cantidad de procesos de supervisión y control, especialmente en el módulo sinóptico para agregar más procesos y permitir crear cualquier proceso al usuario partiendo de elementos que se tengan por defecto.

Se recomienda realizar las conexiones con otros PLC disponibles en el laboratorio de control, para que el SCADA tenga mayor funcionalidad ante el posible aumento de procesos.

Se recomienda subir el SCADA a un servidor web gratuito disponible en la red, para monitorear los procesos desde la distancia, con el fin de darle un avance ante nuevas tecnologías.

REFERENCIAS

- [1] Suarez, J. (2015) *Diseño e implementación de sistemas SCADA para automatismos, basados en hardware y software libre* [Archivo PDF] <https://docplayer.es/11306149-Diseño-e-implementación-de-sistemas-scada-para-automatismos-basados-en-hardware-y-software-libre-jorge-eliecer-suarez-pinzon.html>
- [2] Carlozama, G. (2018) *SCADA para invernadero sobre software libre* [Archivo PDF] <https://1library.co/document/yjknpl6q-scada-para-invernadero-sobre-software-libre.html>
- [3] Uzcátegui, M. «Diseño e implementación de un sistema SCADA para prácticas demostrativas del laboratorio de control utilizando el PLC Siemens S7-300 y software libre», Universidad de Los Andes, 2018.
- [4] Rojas, J. «Desarrollo de módulo sinóptico para un sistema SCADA basado en software libre para ser implementado en el laboratorio de control de la Facultad de Ingeniería», Universidad de Los Andes, 2022.
- [5] Montaña, A. «Diseño de los módulos base de datos, gráfico de tendencias, alarmas y eventos para un sistema SCADA basado en software libre que será implementado en el Laboratorio de Control de la Escuela de Ingeniería Eléctrica de la Universidad de Los Andes», Universidad de Los Andes, 2022.
- [6] Hurtado, I. y Toro, J. «Paradigmas y métodos de investigación en tiempos de cambio», EPISTEME, Consultores Asociados C.A., Carabobo, 2005.
- [7] Hernández, M. «Manual de trabajo de Grado de Especialización y Maestría y Tesis Doctorales,» FEDUPEL. La editorial pedagógica de Venezuela, Caracas, 2016.
- [8] Santa, F. y Pulgarín, D. (2016) *Plataforma web econtinua ASEUTP* [Archivo PDF] https://www.academia.edu/31803770/Manual_UPEL_2016_pdf
- [9] Carrero, D. (2008) *Diseño de un sistema de control supervisorio y adquisición de datos (SCADA) para el monitoreo remoto de los sistema de energía ininterrumpida (UPS)*

pertenciente al sistema eléctrico de una refinería en el país [Archivo PDF]
<http://mriuc.bc.uc.edu.ve/bitstream/handle/123456789/31/dcarrero.pdf?sequence=4>.

[10] González, A. (2013) *Implementación de sistema de supervisión y control (SCADA) vía web mediante recursos de código abierto* [Archivo PDF]
<http://saber.ucv.ve/bitstream/10872/18641/1/TRABAJO%20ESPECIAL%20DE%20GRADO.pdf>.

[11] Sigcho, M. (2011) *El software libre GUADALINEX en el aprendizaje de informática básica en el primer año de bachillerato del colegio nacional chambo período 2009 – 2010* [Archivo PDF] <https://repositorio.uta.edu.ec/bitstream/123456789/5957/1/FCHE-MTIM-789.pdf>.

[12] Chavarría, B. y Gudiño, E. (2017) *Implementación de un servidor web y un diseño de una página utilizando herramientas de software libre para el dispensario “Sagrada Familia” de la ciudad de Guayaquil* [Archivo PDF]
<https://dspace.ups.edu.ec/bitstream/123456789/14162/1/GT001840.pdf>.

[13] Python basics, «¿Qué es Flask Python?», 2021. [En línea]. Disponible: <https://pythonbasics.org/what-is-flask-python/>. [Último acceso: 12-01-2022].

[14] Wikipedia, «Visual Studio Code», 2021. [En línea]. Disponible: https://es.wikipedia.org/wiki/Visual_Studio_Code. [Último acceso: 12-10-2021].

[15] Pérez López, E. «Los sistemas SCADA en la automatización industrial», Tecnología en marcha, Costa Rica, 2015.

[16] Rodríguez Penin, A. «Sistemas SCADA», 3ra edición, Alfaomega Grupo Editor, México, 2013.

[17] Hurtado, J. (2019) *Introducción a las redes de comunicación industrial* [Archivo PDF]
http://www.infoplcn.net/files/documentacion/comunicaciones/infoPLC_net_introduccion-a-las-redes-de-comunicacion-industrial.pdf.

[18] «SIMATIC S7-300 Sistema de automatización S7-300 Manual de producto», Siemens (2017).

[19] Matrikon OPC, «¿Qué es un servidor OPC?», [En línea]. Disponible: <https://www.matrikonopc.es/opc-servidor/index.aspx> [Último acceso: 03-02-2022].

[20] Sicma, «SCADA, ¿Qué es y cómo funciona?», [En línea]. Disponible: <https://www.sicma21.com/scada-que-es-y-como-funciona/> [Último acceso: 05-12-2021]

[21] Siemens, «SIMATIC S7-300, características y funcionamiento», [En línea]. Disponible: <https://new.siemens.com/global/en/products/automation/systems/industrial/plc/simatic-s7-300.html> [Último acceso: 12-12-2021]

[22] Neural covenant, «Django, el framework para Python», [En línea]. Disponible: <https://neuralcovenant.com/2020/09/14/django-rest-framework-cheatsheet/> [Último acceso: 15 - 10 - 2021]

[23] IBH Softec, «Bienvenido a IBHSoftec», [En línea]. Disponible: https://www.ibhsoftec.com/en_GB [Último acceso: 23 - 02 - 2022].

[24] OpenOPC, «OpenOPC para python», [En línea]. Disponible: <http://openopc.sourceforge.net/> [Último acceso: 15 - 02 - 2022].

[25] PLC-City, «Siemens industrial communication», [En línea]. Disponible: <https://www.plc-city.com/shop/en/siemens-industrial-communication/6gk1571-0ba00-0aa0.html> [Último acceso: 16 - 06 - 2022].

[26] Atria Innovation, «Automatización industrial», [En línea]. Disponible: <https://www.atriainnovation.com/wp-content/uploads/2020/06/automatizacion-industrial-1-bis.jpg> [Último acceso: 16 - 06 - 2022].

[27] «Buses de campo», [En línea]. Disponible: http://1.bp.blogspot.com/-IkEy62Hy0Ns/VdPjFtm-15I/AAAAAAAAAYs/CfKMA-IOoh8/s1600/buses_de_campo2.jpg [Último acceso: 16 - 06 - 2022].

[28] KepserverEXOPC, «Qué es OPC y qué es un OPCServer», [En línea]. Disponible: <https://www.kepserversop.com/que-es-opc-y-que-es-un-opc-server/> [Último acceso: 16 - 06 - 2022].

[29] Canal «Dennis Izquierdo - ELECTROTRAKS» (20 de Noviembre de 2021) *Modbus RTU con simulación, OPC-KepServerEX, comunicación Modbus TCP con TIA Portal y*

Modbus Poll [Archivo de Video]. YouTube, <https://www.youtube.com/watch?v=YUfUZJhhA1Y>.

[30] Canal «Fusion Automate» (25 de Noviembre de 2021) *Modbus + Python (Part 0 – Part 1 – Part 2 – Part 3 – Part 4 – Part 5 - Part – 6 – Part 7 – Part - 8)* [Archivo de Video]. YouTube, <https://www.youtube.com/playlist?list=PLxrSjjYyzaaLKgpWVZKHnMGiCwciAmln4>

[31] Canal «Fusion Automate» (12 de Diciembre de 2021) *OPC – DA + Python (Part 1 - Part 2 – Part 3 – Part 4)* [Archivo de Video], YouTube. <https://www.youtube.com/playlist?list=PLxrSjjYyzaaIMru10eZDRXN-0vDexEO0j>

[32] Canal «General Technology Knowledge» (18 de Diciembre de 2021) *PLC S7-300 connect with KepServerEX OPC Server and GE SCADA Proficy Cimplicity HMI* [Archivo de Video], YouTube, <https://www.youtube.com/watch?v=GRvMKdRWEmI&t=2103s>

[33] Canal «General Technology Knowledge» (3 de Abril de 2022) *PLC S7-300 CPU 314C-2DP connect with KepServerEX via Profinet module* [Archivo de Video], YouTube. <https://www.youtube.com/watch?v=nGgTTT3koOs&t=849s>

[34] Canal «Fusion Automate» (12 de Enero de 2022) *SIEMENS PLC S7-300 Communication with KepServerEX in Simulation mode* [Archivo de Video], YouTube, https://www.youtube.com/watch?v=usCWJY_sdv8

[35] Canal «General Technology Knowledge» (08 de Marzo de 2022) *PLC S7-300 connect with Node-Red full tutorial* [Archivo de Video], YouTube, <https://www.youtube.com/watch?v=Rv2PdKRdAao>