UCUENCA

Universidad de Cuenca

Facultad de Ingeniería

Carrera de Ingeniería en Telecomunicaciones

ANÁLISIS DE LA FACTIBILIDAD DE MEJORAR LA DISPONIBILIDAD DEL SISTEMA SCADA DURANTE LOS PROCESOS DE MANTENIMIENTO DE LOS EQUIPOS DE LA MICRO-RED DE LA UNIVERSIDAD DE CUENCA.

Trabajo de titulación previo a la obtención del título de Ingeniero en Telecomunicaciones.

Autores:
Lourdes Verónica Gutiérrez Otavalo
Claudia Estefanía Padilla Guamán

Director:
Darwin Fabian Astudillo Salinas
ORCID: 0000-0001-7644-0270
Co-Director:
Edisson Andrés Villa Ávila
ORCID: 0000-0002-2766-5913

Cuenca, Ecuador

2024-09-06

Resumen

El presente informe se enfoca en los desafíos asociados con el aumento de la disponibilidad de sistemas *Supervisory Control and Data Acquisition* (SCADA) en el ámbito industrial, específicamente en el Laboratorio de Micro-Red Eléctrica de la Universidad de Cuenca. Este sistema supervisa y controla diferentes equipos a través de una red de *Operational Technologies* (OT), ya sea cableada.

El problema principal abordado radica en las interrupciones ocasionadas en la comunicación y control del SCADA durante los procedimientos de mantenimiento de los dispositivos, lo cual afecta la funcionalidad del sistema. La desconexión de la alimentación eléctrica para realizar mantenimiento resulta en una pérdida temporal de la detección y control de los dispositivos por parte del SCADA, generando así una paralización forzosa del sistema.

Para resolver este inconveniente, se proponen tres hipótesis potenciales. En primer lugar, se explora la gestión de la redundancia de la red mediante *switches Weidmüller*, utilizando protocolos como *Spanning Tree Protocol* (STP), *Rapid Spanning Tree Protocol* (RSTP), *Spanning Tree* y *Turbo-Ring*. En segundo lugar, se plantea la instalación de un *bypass* físico en la red para prevenir interrupciones, reconfigurando la infraestructura de fibra óptica. Por último, se analiza la implementación en Laboratory Virtual Instrument Engineering Workbech (LabVIEW) del SCADA y se corrigen las fallas detectadas. La implementación de las soluciones seleccionadas han logrado mantener la continuidad del control y la comunicación en el sistema SCADA durante los procesos de mantenimiento.

Finalmente, el trabajo de titulación ha concluido con el cumplimiento de todos los objetivos planteados y la mejora considerable de la disponibilidad del SCADA. Esto se logró a través de todos los experimentos realizados, principalmente la modificación de la programación en LabVIEW.

Palabras clave del autor: solución, sistema, control, adquisición, software, comunicación, métodos, experimentos, modbus, proceso.

Abstract

This report focuses on the challenges associated with increasing the availability of *Supervisory Control and Data Acquisition* (SCADA) systems in the industrial sector, specifically in the Micro-Grid Laboratory of the University of Cuenca. This system monitors and controls various equipment through a wired *Operational Technologies* (OT) network.

The primary problem addressed lies in the interruptions caused in the communication and control of the SCADA system during device maintenance procedures, which affects the system's functionality. Disconnecting the power supply for maintenance results in a temporary loss of device detection and control by the SCADA, thus causing a forced system shutdown.

To address this issue, three potential hypotheses are proposed. Firstly, the management of network redundancy through Weidmüller switches is explored, using protocols such as *Spanning Tree Protocol* (STP), *Rapid Spanning Tree Protocol* (RSTP), Spanning Tree, and Turbo-Ring. Secondly, the installation of a physical bypass in the network to prevent interruptions is suggested by reconfiguring the fiber optic infrastructure. Lastly, the SCADA implementation is analyzed, focusing on the algorithm implemented in Laboratory Virtual Instrument Engineering Workbech (LabVIEW) and correcting the detected flaws. Each hypothesis was validated through the research methods utilized. The implementation of the selected solutions has achieved continuity in the control and communication of the SCADA system during maintenance processes.

Finally, the thesis work concluded with the fulfillment of all the objectives set and a considerable improvement in the system's initialization time, achieved through all the experiments conducted, mainly by modifying the LabVIEW programming.

Autor keywords: solution, system, control, acquisition, software, communication, methods, experiments, modbus, process.

Índice de contenido

Índice de figuras

Índice de tablas

Agradecimientos.

A Dios, por todo lo que ha permitido. Agradecemos a todas las personas que nos apoyaron a lo largo de estos años y durante el desarrollo de este trabajo. A los profesores que llenaron nuestras vidas de esperanza y posibilidades. Especialmente, al personal de la Micro-red de la Universidad de Cuenca por su apertura y apoyo hacia este trabajo de titulación.

Nuestra gratitud al Ingeniero Darwin Astudillo, director de este trabajo, por su gran empeño y apoyo en todo momento, y al Ingeniero Edisson Villa, codirector, por la ayuda brindada. A todos ustedes, gracias infinitas y nuestros mejores deseos.

Dedicatoria.

Al concluir este camino a lo largo de mi carrera, quiero dedicar el presente trabajo de titulación a mis padres, Fabián y Lorena. Ellos han sido los pilares fundamentales en mi vida personal y académica, brindándome cada día su amor y apoyo incondicional. Gracias a su entrega y sacrificio, he recibido la mejor educación y las mejores oportunidades posibles. Su duro trabajo, las madrugadas a mi lado y los momentos compartidos han dado frutos, siendo ellos la inspiración detrás de cada uno de mis logros. No hay palabras suficientes para expresar cuánto los amo y cuánto les debo.

Quiero expresar mi gratitud también a mi familia: a mis abuelos Alejandrina, Teresa y Moisés, y a mis tíos Rosa, Germán, Diana y Ricardo, que han sido un apoyo constante. De igual manera, a mi querido hermano Daniel, por su ayuda y respaldo a lo largo de mi vida. Estas personas han caminado a mi lado, construyendo juntos el sendero que me ha llevado hasta aquí. Sus palabras de aliento y su compañía han sido fuente de fuerza y motivación.

Dedico este trabajo también a Sebastián, que desde el primer día ha sido mi mano derecha y el hombro en el que he descansado en los buenos y malos momentos. Gracias por tu apoyo inquebrantable, por ser una luz en los días oscuros, y una persona tan esencial en este arduo camino. De igual manera a su familia, les agradezco profundamente por su apoyo en estos últimos años.

Asimismo, quiero reconocer a mi compañera y amiga de este trabajo de titulación, Claudia. Gracias a su dedicación, confianza y conocimiento, estos años han sido más llevaderos y enriquecedores. Sin duda, ha sido la mejor persona con quien compartir este desafío. Su presencia ha hecho de este viaje algo memorable y lleno de aprendizajes compartidos.

Agradezco también a los amigos que me han acompañado desde el colegio hasta la universidad: Melissa, Andrés y Felipe. Gracias por su amistad inquebrantable, su cariño y apoyo constante. Han sido mi red de apoyo en cada etapa de esta travesía. Su lealtad y capacidad para estar ahí, sin importar las circunstancias, han sido invaluables para mí. En los momentos más difíciles, su presencia me ha dado la fuerza para seguir adelante, y en los momentos de alegría, su compañía ha hecho cada celebración aún más especial.

Finalmente, quiero agradecer a Dios, quien me ha sostenido en cada momento, y a la Virgen María, cuya guía materna ha sido esencial para la culminación de mis estudios. Con su ayuda, he encontrado la fortaleza y el consuelo necesarios para llegar hasta aquí. Su presencia en mi vida ha sido una fuente constante de esperanza y fe, iluminando mi camino y brindándome paz en los momentos de incertidumbre.

A todos ustedes, mi más sincero agradecimiento.

Lourdes

Dedicatoria.

Al finalizar la etapa más desafiante de mi vida, dedico este trabajo a Dios, para ello a mi corazón le sobran razones.

A mis padres, Luz María y Bolívar, cuyo esfuerzo incansable me ha permitido salir de casa para perseguir un sueño que en muchas ocasiones parecía inalcanzable, pero que gracias a su apoyo nunca se desvaneció. Su amor y sacrificio han sido el motor de mi perseverancia.

A mis hermanas, Viviana y Valentina, los pilares más importantes de mi lucha diaria. Gracias por su inagotable paciencia y por ser siempre mi fuente de inspiración y fuerza.

A Martina, mi abuela, con quien hubiese sido un honor compartir este momento. Donde quiera que estés, esto también lo logré por ti. A mi familia, gracias por creer en mí y por su apoyo en cualquier forma.

A mis entrañables amigos, Christian y Jonnathan, quienes han sido mi refugio en más de una tormenta y convirtieron esta ciudad en mi segundo hogar. A ustedes, mi profundo agradecimiento por su amistad invaluable y por los momentos de alegría y consuelo.

A Diana, mi mejor amiga, quien nunca ha dejado de creer en mí y estar a mi lado durante cada temporal. Tu fe en mí ha sido un faro en los días más oscuros.

A Lourdes, amiga y compañera en este desafío, por los momentos compartidos, su esfuerzo y compromiso. Sin duda, has sido el complemento idóneo para lograr esta meta, y tu compañía ha sido esencial en este viaje.

A quienes han sido un regalo de Dios en mi vida. Gracias infinitas por vuestro apoyo y por ser el ancla que necesitaba en mi travesía.

Dedico también este trabajo a los ingenieros Fabián Astudillo y Kenneth Palacio, quienes no solo han sido profesores, sino también amigos. Su ayuda y enseñanza han hecho este recorrido más placentero. De manera especial, a Fabián, por su dedicación y apoyo en cada situación en la que he requerido su ayuda. El labor docente de personas como ustedes ha hecho posibles los sueños de personas como yo.

Finalmente, a quienes enfrentan la sombra de la tristeza, les recuerdo con cariño que cada problema es superable y aún no ha llegado el momento en que un día malo nos derrote.

A ustedes y a cada uno de los amigos que conocí en estos años, mi eterna gratitud.

Claudia

Glosario

ACK Acknowledgment. 37–39

API *Application Programming Interface*. 26, 50, 78

APIS *Application Programming Interfaces*. 5–7, 16, 39, 50, 51, 58–64

ARP Address Resolution Protocol. 37–39

CCTI-B Centro Científico, Tecnológico y de Investigación Balzay. 15

CD Compact Disc. 22

DCS Distributed Control System. 25

DER Distributed Energy Resource. 20

DNS Domain Name System. 41

DoE Design of experiments. 32–34

DSC Datalogging and Supervisory Control. 23, 55

DVD Digital Versatile Disc. 22

GB GigaBytes. 22, 74, 77

GELF Graylog Extended Log Format. 24

GLP Gas Licuado de Petróleo. 16

GNU GNU's Not Unix. 21

GPG GNU Privacy Guard. 76

HTTP Hypertext Transfer Protocol. 38, 39, 77

HTTPS Hypertext Transfer Protocol Secure. 76

I/O Input/Output. 26, 60

IoT Internet of Things. 68

IP Internet Protocol. 27, 35, 37–42, 49, 74

ISO International Organization for Standardization. 74

LabVIEW Laboratory Virtual Instrument Engineering Workbech. 2, 3, 6, 15, 18, 22–24, 34, 40, 48, 49, 53–58, 60–63, 66, 67

LED Light Emitting Diode. 28

LTS Long Term Support. 74

MAC Media Access Control. 41

MAX Measurement and Automation Explorer. 25

MB MegaBytes. 77

NI National Instruments. 24, 25

NI MAX NI Measurement & Automation Explorer. 55

ODBC Open Database Connectivity. 24

OT *Operational Technologies*. 2, 3, 15, 16, 19, 25, 35, 36, 49, 52, 66, 68

PAC *Programmable Automation Controller*. 15

PLC *Programmable Logic Controller*. 5–7, 15, 25–27, 32, 39, 40, 42, 43, 47, 50–53, 58–64, 66, 68

PON Red Óptica Pasiva. 28

RACK Risk Aware Consensual Kink. 4, 26

RAM Random Access Memory. 22, 74

RSTP *Rapid Spanning Tree Protocol*. 2, 3, 18, 30, 31

RTU Remote Terminal Unit. 27

SCADA *Supervisory Control and Data Acquisition*. 2, 3, 5–7, 15–19, 22–27, 32, 34, 35, 39–41, 43, 45, 48–58, 60–64, 66–68

SHA Secure Hash Algorithm. 78

SIEM Gestión de Eventos e Información de Seguridad. 23

SQL Structured Query Language. 55

SSDP Simple Service Discovery Protocol. 38, 39

SSL Secure Sockets Layer. 77

SSMS SQL Server Management Studio. 55

STP *Spanning Tree Protocol*. 2, 3, 18, 30

SYN Synchronous Idle Mode. 39

Syslog System Logging Protocol. 78

TCP Transmission Control Protocol. 6, 27, 37–39, 53, 54

TI Tecnologías de la Información. 23, 25

TIC Tecnologías de la Información y Comunicación. 15

TLS Transport Layer Security. 38

UDP User Datagram Protocol. 37, 38, 43

URI Uniform Resource Identifier. 78

URL Uniform Resource Locator. 78

USB Universal Serial Bus. 22

USD United States Dollar. 15

Web World Wide Web. 21, 78

1. Introducción

1.1. Antecedentes

Los sistemas *Supervisory Control and Data Acquisition* (SCADA) se utilizan en aplicaciones industriales para supervisar una amplia gama de parámetros que dependen de los equipos que conforman el sistema. Estos equipos recopilan datos a través de una red *Operational Technologies* (OT) cableada ó inalámbrica [2]. Preliminarmente, un sistema SCADA diseñado para abordar los desafíos, incorpora componentes automáticos, como los controladores *Programmable Logic Controller* (PLC) y *Programmable Automation Controller* (PAC). Sin mencionar que, estos dispositivos electrónicos tienen la capacidad de operar automáticamente, eliminando la necesidad de que un operador esté presente en el lugar de la falla [3].

En lo que a monitoreo y control se refiere, se engloban dispositivos que reciben datos para su visualización o análisis, los cuales son recopilados a partir de múltiples sensores. Estos últimos registran parámetros como el rendimiento, la potencia de salida, la temperatura, vibraciones, tensión, corriente, entre otros [3]. Además, los inversores habilitan a todos los sistemas de almacenamiento, ciertas fuentes de generación y los vehículos eléctricos a suministrar energía a la red mediante un convertidor de corriente continua a corriente alterna. Este dispositivo controla y supervisa aspectos como el voltaje, la frecuencia, la corriente y el ángulo de fase como parte integral de su sistema de gestión estándar [4].

En este contexto, el Laboratorio de la Micro-Red del Centro Científico, Tecnológico y de Investigación Balzay (CCTI-B) de la Universidad de Cuenca se estableció con el propósito de fomentar la investigación aplicada, la innovación y la educación en el ámbito de la energía sostenible. Financiado a través del Programa Canje de Deuda Externa del Ecuador frente a España, este laboratorio fue equipado en dos etapas con un total de 3’298,000.00 United States Dollar (USD). Su infraestructura está diseñada para apoyar la formación técnica y tecnológica, pregrado, posgrado y formación continua, enfocándose en campos como ingeniería, industria, Tecnologías de la Información y Comunicación (TIC), y específicamente en micro-redes, energías renovables no convencionales, control y operación de sistemas eléctricos, electrónica y automatización. Además, su objetivo es contribuir al cambio en las matrices energética y productiva del país desde la Universidad de Cuenca, alineándose con la visión de la institución de ser una comunidad universitaria innovadora y resiliente, comprometida con la generación de conocimiento de calidad y pertinente, y con un fuerte compromiso con la sociedad [5].

Entonces, como parte de las investigaciones más recientes llevadas a cabo en el laboratorio, se ha puesto en funcionamiento un sistema SCADA. Mismo que integra y monitorea la mayoría de los dispositivos disponibles en el laboratorio antes mencionado. El *dashboard* del sistema SCADA está implementado en una interfaz de Laboratory Virtual Instrument Engineering Workbech (LabVIEW) que permite la gestión de cada uno de estos equipos [4]. Adicionalmente, se dispone de una red OT de fibra óptica con una topología en forma

de anillo para la comunicación física de todos los equipos que pertenecen a la red y que se han incluido en el sistema SCADA. Estos últimos incluyen generadores fotovoltaicos, grupos electrógenos que funcionan con Gas Licuado de Petróleo (GLP) y diésel, una micro-turbina generadora, así como sistemas de almacenamiento [6]. En este sentido, el SCADA desempeña un papel esencial al posibilitar el monitoreo y control en tiempo real de los datos captados por cada dispositivo perteneciente al sistema de adquisición de datos [7].

Por otro lado, como parte fundamental del proceso industrial se ha considerado proporcionar asistencia técnica y servicios de mantenimiento a los dispositivos entregados dentro del período de garantía técnica. Esto abarca servicios de post-venta, que comprenden tanto el mantenimiento preventivo como el correctivo, además de la atención en casos de garantía [8].

1.2. Justificación

Los dispositivos que integran el laboratorio de Micro-red desempeñan un papel esencial en las funciones de adquisición y control ejecutadas por el sistema SCADA. En consecuencia, como se mencionó previamente, resulta necesario llevar a cabo rutinariamente procedimientos de mantenimiento en estos equipos. Sin embargo, al desconectar la alimentación eléctrica para realizar dichas tareas de mantenimiento, se produce una interrupción en la comunicación y en el control de los dispositivos por parte del sistema SCADA [8].

Para comprender el modo en el que se distribuyen los equipos de la Micro-Red, se analiza la topología de la misma. En este sentido, la red OT se despliega en una topología en anillo, como se muestra en la Figura 1.1. En esta topología, los datos se transmiten de un nodo a otro siguiendo su patrón de configuración. La red se divide en diferentes *Application Programming Interfaces* (APIS), mismas que albergan equipos relacionados con funciones específicas. En este contexto, el laboratorio cuenta con un total de 6 APIS. De estos, los APIS 1, 2 y 3 están integrados en el SCADA. Por otro lado, los APIS 5 y 6 actualmente se encuentra fuera de funcionamiento, lo que sugiere que no está operativo en la red en este momento. Finalmente, el APIS 4 se reserva específicamente para su uso en el sistema Irquis.

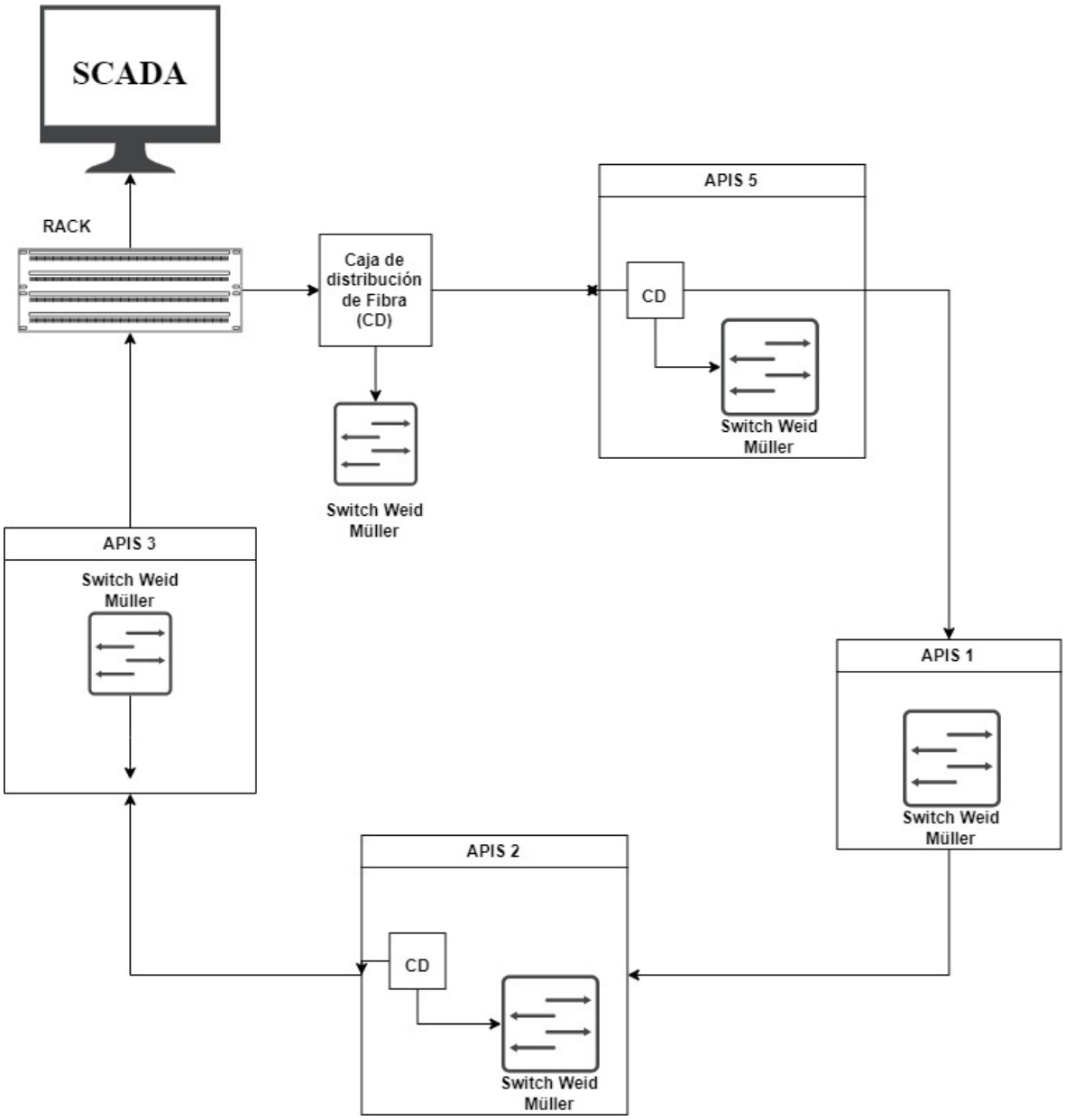


Figura 1.1: Red de fibra óptica de los equipos conectados a SCADA (topología de tipo anillo)

Durante los procesos de mantenimiento que se han realizado dentro del laboratorio, se ha observado que la pérdida de la detección de uno de estos equipos por parte del SCADA provoca la generación de un mensaje de alerta. Este último conlleva a la detención forzosa del sistema, lo que impide el control de los demás dispositivos. En consecuencia, el SCADA queda inoperable durante el tiempo destinado al mantenimiento, lo cual impide las funcionalidades que provee el sistema dentro de la operación normal del laboratorio de la Micro-red. En este contexto, se plantea un desafío significativo en la gestión de estos equipos, el cual requiere soluciones específicas para asegurar que el mantenimiento se realice de manera eficiente sin afectar la funcionalidad del sistema SCADA que permite el monitoreo eficiente de los mismos.

1.3. Alcance

El propósito principal de este estudio es afrontar el problema de la pérdida de control y comunicación en el sistema SCADA que ocurre durante los procedimientos de mantenimiento de los dispositivos. Se han considerado diversas hipótesis de solución, cada una respaldada por diferentes estrategias destinadas a resolver el inconveniente de la interrupción. En primer lugar, se explora la gestión de la redundancia de la red a través de los *switches Weidmüller*. Esta opción se enfoca en protocolos como *Spanning Tree Protocol* (STP), *Rapid Spanning Tree Protocol* (RSTP), *Turbo-Ring* y *Turbo-Chain* en el *switch* correspondiente [9].

Otra propuesta considerada en este trabajo, implica establecer un *bypass* físico en la red con el propósito de prevenir las interrupciones. Esta estrategia busca que la red sea tolerante a fallos y se apoya en componentes como un divisor de potencia pasivo óptico, fibras de derivación, etc [10]. En otras palabras, se busca reconfigurar la infraestructura de fibra, garantizando la continuidad de la operación y la comunicación sin interrupciones durante los procesos de mantenimiento [11]. De esto, se establecerán las pautas requeridas para la reconexión de la red, mismas que deberán seguir los funcionarios del laboratorio durante estos eventos.

Por otro lado, se plantea una tercera solución que implica el análisis de la comunicación proporcionada por la arquitectura cliente-servidor utilizada por SCADA. En esta opción, se observará la forma en que los datos son adquiridos por SCADA, para evaluar la implementación del *Software* LabVIEW y la posibilidad de corregir las fallas detectadas [12]. La meta es lograr la reconfiguración de modo que no se genere la interrupción forzada del sistema. En lugar de eso, la red se recuperará de las fallas registradas y continuará funcionando [13].

Finalmente, una vez evaluadas estas hipótesis, según los criterios pertinentes para el funcionamiento requerido en esta red, se procederá a seleccionar la opción más adecuada [14]. Los criterios serán definidos durante el desarrollo del trabajo de titulación. Con la implementación de la solución seleccionada se busca mantener la continuidad del control y la comunicación en el sistema SCADA durante los procesos de mantenimiento de los equipos. Se proporcionarán las pruebas y validaciones correspondientes como parte integral del proceso. Finalmente, se brindarán las conclusiones y recomendaciones para trabajos futuros.

1.4. Objetivos

1.4.1. Objetivo general

Analizar la factibilidad de mejorar la disponibilidad del sistema SCADA durante los procesos de mantenimiento de los equipos de la Micro-red de la Universidad de Cuenca.

1.4.2. Objetivos específicos

- Identificar las posibles soluciones para evitar la pérdida de comunicación y control del sistema SCADA durante procesos de mantenimiento de los equipos congregados.
- Identificar y evaluar las características configurables de los equipos con el fin de gestionar eficazmente la redundancia de la red OT.
- Implementar la solución más factible ante la pérdida de control del sistema SCADA durante procesos de mantenimiento.

2. Marco Teórico

En este capítulo se abordarán todos los conceptos teóricos considerados de importancia, los cuales permitieron desarrollar el presente trabajo de titulación. Así mismo, se incluyen los principales sistemas operativos y elementos utilizados en el laboratorio de la Micro-red. Cabe destacar que, no se ha definido una Sección de trabajos relacionados para el desarrollo de este trabajo de titulación; debido a que, la revisión de la literatura no ha proporcionado artículos que aporten pautas concretas orientadas a cumplir con los objetivos planteados.

Por otro lado, en esta revisión se ha incluido la metodología de investigación para definir los métodos que avalan el desarrollo de este trabajo.

2.1. Micro-Red

Una Micro-Red es un sistema energético autosuficiente y localizado, que opera independientemente a la red principal de energía o como una entidad controlable en relación con la red principal de energía. Este sistema está compuesto por Distributed Energy Resource (DER), como plantas solares fotovoltaicas, turbinas eólicas, sistemas de almacenamiento como baterías y generadores convencionales, todos integrados y controlados mediante herramientas de *software* avanzadas y tecnologías de comunicación. Una Micro-Red es capaz de atender a una pequeña comunidad energética, un complejo de edificios o incluso a un solo hogar. Su diseño busca mejorar la resiliencia, aumentar la eficiencia y reducir las emisiones de carbono [15]. En la Figura 2.1 se observa un diagrama de la interacción de la Micro-Red con sus recursos y su comunidad energética.

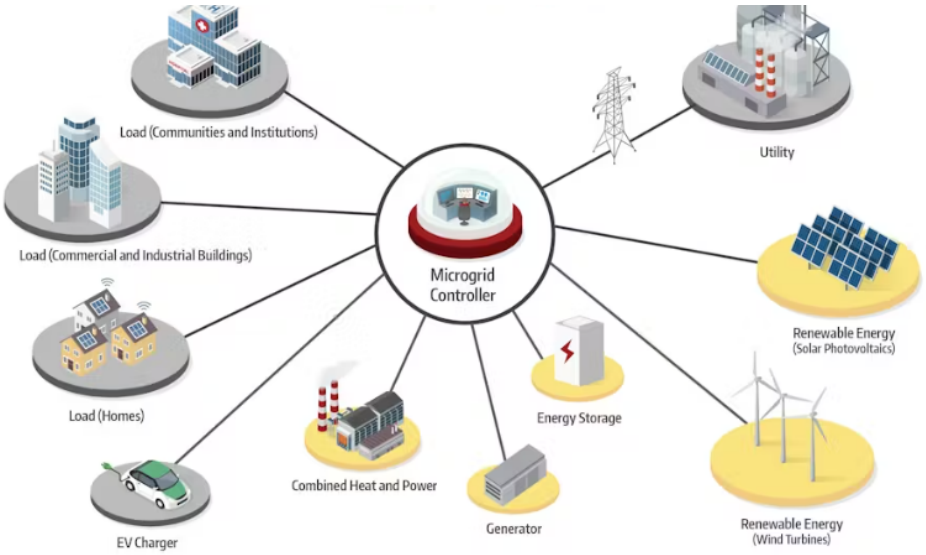


Figura 2.1: Diagrama de la interacción de una Micro-Red. Tomado de: [1]

Además, una Micro-Red genera energía para clientes cercanos, lo que la distingue de las grandes redes centralizadas. Al estar cerca de los usuarios, los generadores de la Micro-

Red, como los paneles solares o los generadores, evitan las pérdidas de electricidad que ocurren en la transmisión de larga distancia. Además, este sistema se desconecta de la red principal y operar de forma independiente, lo que le permite suministrar energía a sus clientes en caso de fallo en la red central.

En una Micro-Red, los sistemas avanzados son inteligentes y están controlados por un *software* central que gestiona los generadores, las baterías y los sistemas energéticos de los edificios cercanos con alta sofisticación. Este controlador orquesta múltiples recursos para cumplir con los objetivos energéticos establecidos por los clientes de la Micro-Red, como la reducción de costos, la utilización de energía limpia o la mejora de la fiabilidad eléctrica. A través de algoritmos complejos, los recursos de la Micro-Red interactúan de manera autónoma para maximizar la eficiencia y la fiabilidad del sistema [1].

2.2. *Software*

2.2.1. Ubuntu 22.04 LTS

Ubuntu 22.04 LTS es una distribución de *Linux*, un sistema operativo que trabaja bajo la Licencia General Pública de GNU's Not Unix (GNU) [16]. Esta distribución está basada en *Debian* y fue fundada por el empresario sudafricano *Mark Shuttleworth*, con el objetivo de desarrollar un sistema operativo accesible para todos. Su variante *Ubuntu Server*, diseñada para servidores, es muy utilizada en entornos tanto particulares como profesionales, especialmente en servidores World Wide Web (Web). De hecho, su principal característica es que se gestiona a través de la línea de comandos y se administra comúnmente de forma remota [17]. Esta versión de *Ubuntu Server* fue lanzada el 21 de abril de 2022 y tiene soporte hasta abril de 2027 para actualizaciones de seguridad estándar y hasta abril de 2032 con soporte extendido [18].

En el laboratorio de la Micro-Red de la Universidad de Cuenca, la distribución *Ubuntu 22.04 LTS* se implementa como la plataforma principal para el servidor de registros.

2.2.2. Windows Server 2012

Windows Server 2012 R2 es una versión de *Windows* que incorpora funciones de gestión de servidores, almacenamiento, redes definidas por *software*, automatización, protección de datos y escritorio virtual [19]. La versión R2 fue lanzada el 18 de octubre de 2013. Este sistema ofrece herramientas sofisticadas para administración remota y diversas ediciones especializadas como *Standard*, *Datacenter* y *Essentials*, adaptadas a diferentes necesidades de servidor. Su modelo de licenciamiento se basa en el número de núcleos o procesadores del servidor [20]. Sin embargo, esta versión de *Windows Server* ha finalizado su soporte principal en octubre de 2018 y el soporte extendido concluyó el 10 de octubre de 2023 [21].

En el laboratorio de la Micro-Red de la Universidad de Cuenca, la distribución *Windows Server 2012 R2* funciona como el servidor que contiene el programa de Laboratory Virtual Instrument Engineering Workbech (LabVIEW) del sistema *Supervisory Control and Data Acquisition* (SCADA).

2.2.3. VMware Workstation Pro

VMware Workstation Pro es una herramienta avanzada de virtualización que permite desarrollar, probar y desplegar software ejecutando múltiples sistemas operativos basados en x86, como *Windows* y *Linux*, simultáneamente en una sola computadora física, como si fueran equipos independientes. Facilita la replicación de entornos de servidor, escritorio y tableta en máquinas virtuales, asignando recursos como núcleos de procesador, memoria y gráficos [22]. Sus funcionalidades incluyen la configuración de memoria, discos duros, unidades de Compact Disc (CD)/Digital Versatile Disc (DVD) y dispositivos Universal Serial Bus (USB) 2.0, con soporte para hasta 8 GigaBytes (GB) de Random Access Memory (RAM) por máquina virtual y la capacidad de clonar máquinas físicas e importar máquinas virtuales de diversos proveedores. Ofrece diversas configuraciones de red y hasta 10 *switches* virtuales para conectar máquinas virtuales, anfitriones y redes públicas. Permite ajustes automáticos de resolución de pantalla, ejecución de máquinas en segundo plano, y facilita la interacción entre el sistema invitado y el anfitrión mediante funciones como carpetas compartidas y arrastrar y soltar. También incluye la posibilidad de capturar la actividad de pantalla y realizar instantáneas del estado de las máquinas virtuales [23].

En el laboratorio de la Micro-Red de la Universidad de Cuenca, el *software VMware Workstation Pro* funciona como un visualizador para el servidor principal, que integra los servidores *Ubuntu 22.04 LTS* y *Windows Server 2012 R2*, accediendo a ellos mediante una conexión remota.

2.2.4. LabVIEW

LabVIEW se presenta como una plataforma que adopta un enfoque de programación gráfica para permitir al usuario visualizar cada faceta de su aplicación, desde la configuración del hardware hasta los datos de medidas y la depuración [24].

Entre los beneficios que ofrece LabVIEW se destacan [24]:

- Programación gráfica intuitiva que permite al usuario programar de acuerdo a su pensamiento.
- Amplio soporte de hardware.
- Capacidad para recopilar y visualizar datos.
- Integración fluida con otras herramientas de software.

LabVIEW incorpora *Historical Trend*, término que se refiere a la capacidad de visualizar y analizar datos históricos adquiridos a lo largo del tiempo mediante esta función. Esta capacidad posibilita el monitoreo continuo y la evaluación retrospectiva de sistemas y procesos. La función de *Historical Trend* permite a los usuarios graficar datos históricos, identificar patrones y tendencias, y realizar análisis comparativos a lo largo de diferentes periodos. Además, esta función facilita la toma de decisiones informadas basadas en el análisis de datos pasados, lo permite la optimización y mejora continua de procesos industriales y experimentales [24].

El módulo Datalogging and Supervisory Control (DSC) de LabVIEW ofrece una *suite* de herramientas para la adquisición de datos, el control supervisado y la generación de informes. Además. permite a los usuarios desarrollar aplicaciones para monitoreo y control en tiempo real, integrando datos de múltiples fuentes en una interfaz centralizada. Entre sus características destacan:

- **Adquisición de datos:** DSC facilita la captura de datos de una amplia gama de sensores y dispositivos, permitiendo la integración de datos en tiempo real en aplicaciones de LabVIEW.
- **Control supervisado:** Ofrece herramientas para la supervisión y control de sistemas distribuidos, permitiendo a los usuarios gestionar y controlar procesos industriales de manera remota y eficiente.
- **Generación de informes:** Permite la creación de informes detallados y personalizados, facilitando la documentación y el análisis de datos para la toma de decisiones.
- **Alarmas y eventos:** Configura alarmas y eventos para notificar a los usuarios sobre condiciones anómalas o críticas, mejorando la capacidad de respuesta ante situaciones imprevistas [24].

En el laboratorio de la Micro-Red de la Universidad de Cuenca, la plataforma LabVIEW implementa el programa del sistema SCADA, permitiendo la supervisión y el control de los procesos en ejecución.

2.2.5. Graylog

Graylog se presenta como una solución para un sistema Gestión de Eventos e Información de Seguridad (SIEM). Además, proporciona una plataforma de análisis de registros que simplifica la recolección, búsqueda, análisis y alertas de diversos datos generados por máquinas. Este sistema está diseñado específicamente para capturar datos de múltiples fuentes, permitiendo centralizar, proteger y monitorear los registros de manera efectiva. *Graylog* ofrece una amplia gama de funciones en seguridad cibernética, incluyendo la agregación de datos, análisis de seguridad (a través de informes y paneles), correlación y monitoreo de eventos de seguridad, análisis forense, detección y respuesta a incidentes, consola de alertas o respuesta en tiempo real y gestión de cumplimiento de Tecnologías de la Información

(TI) [25].

Para recibir los datos enviados al servidor se utiliza Graylog Extended Log Format (GELF), un formato de registro estructurado creado por *Graylog*. Mismo que, proporciona una forma eficiente de enviar registros desde diversas aplicaciones y dispositivos a *Graylog* para su almacenamiento y análisis [25].

En el laboratorio de la Micro-Red de la Universidad de Cuenca, el servidor *Graylog* recopila los registros de los mensajes generados para el seguimiento y la medición de la actividad de los elementos que componen el sistema SCADA.

2.2.6. Scapy

Scapy es una herramienta *Python* que ofrece al usuario la capacidad de manipular paquetes de red de manera interactiva. Permite enviar, rastrear, analizar y falsificar paquetes, lo que facilita la creación de herramientas para explorar, escanear o incluso atacar redes. Con *Scapy*, es posible falsificar o decodificar paquetes de diversos protocolos, enviarlos a través de cables, capturarlos, emparejar solicitudes y respuestas, entre otras funciones avanzadas. Este programa es versátil y puede realizar una amplia gama de tareas, incluyendo escaneo de red, rastreo de rutas, sondeo, pruebas unitarias, ataques y descubrimiento de redes. Además, reemplaza varias herramientas conocidas como *hping*, *arpspoof*, *arp-sk*, *arping* e incluso algunas partes de *Nmap*, *tcpdump* y *tshark* [26].

En el laboratorio de la Micro-Red de la Universidad de Cuenca, para el presente trabajo de titulación, la función *Scapy* de *Python* se utiliza para capturar los mensajes intercambiados en el sistema SCADA.

2.2.7. Citadel

Citadel es una base de datos en tiempo real y alta velocidad diseñada para aplicaciones de LabVIEW y utilizada para almacenar grandes volúmenes de datos. Sus principales características incluyen:

- **Almacenamiento de alta velocidad:** Capaz de manejar grandes flujos de datos en tiempo real, asegurando que los datos se almacenen de manera rápida y eficiente.
- **Recuperación eficiente:** Permite la recuperación rápida de datos históricos para análisis y visualización, facilitando el acceso a la información necesaria para evaluaciones y auditorías.
- **Compatibilidad Open Database Connectivity (ODBC):** National Instruments (NI) proporciona un controlador ODBC para bases de datos *Citadel*, permitiendo que los datos históricos se exporten a software de terceros compatible con ODBC [24].

En el laboratorio de la Micro-Red de la Universidad de Cuenca, la base de datos *Citadel* actúa como un recolector de los datos captados por el sistema SCADA en tiempo real.

2.2.8. Measurement and Automation Explorer (NI MAX)

Measurement and Automation Explorer (MAX) es una herramienta integral proporcionada por NI para la configuración y administración de dispositivos de adquisición de datos y sistemas de prueba. NI MAX permite a los usuarios:

- **Configurar hardware y software de NI:** Facilita la instalación, configuración y gestión de dispositivos de *hardware* NI, así como la configuración de *software* relacionado.
- **Exportar e importar configuraciones:** Permite la exportación e importación de configuraciones del sistema.
- **Crear y editar componentes:** Proporciona herramientas para crear y editar canales, tareas, interfaces, escalas e instrumentos virtuales.
- **Diagnósticos y pruebas del sistema:** Incluye paneles de prueba y herramientas de diagnóstico que permiten a los usuarios ejecutar pruebas y verificar el estado y rendimiento de sus sistemas de adquisición de datos y control.
- **Interfaz intuitiva:** Ofrece una interfaz de usuario amigable que simplifica la configuración y gestión de dispositivos y sistemas [27].

En el laboratorio de la Micro-Red de la Universidad de Cuenca, la herramienta NI MAX se utiliza para que los operarios puedan visualizar gráficos de interés de los datos captados por el sistema SCADA.

2.3. Red de tecnología operativa

Operational Technologies (OT) se refiere a un conjunto de sistemas informáticos y tecnológicos diseñados específicamente para controlar y supervisar procesos industriales y operacionales en entornos físicos. A diferencia de los sistemas TI, que se centran en el procesamiento de datos y la comunicación digital, la OT interactúa directamente con el mundo físico para automatizar y gestionar operaciones en áreas críticas como la manufactura, la energía, la salud, la gestión de edificios y otros sectores similares [28].

Los sistemas de OT incluyen una variedad de dispositivos y tecnologías especializadas, como *Programmable Logic Controller* (PLC), Distributed Control System (DCS), SCADA, entre otros. Estos sistemas son fundamentales para supervisar y controlar procesos físicos, como la producción de bienes, la gestión de infraestructuras críticas como redes eléctricas o sistemas de suministro de agua, y para garantizar un funcionamiento eficiente y seguro de las operaciones industriales y de infraestructura [29].

2.3.1. SCADA

SCADA se centra en la supervisión y adquisición de datos en sistemas de control mediante un paquete de *software* que se superpone al *hardware*, como PLC u otros módulos comerciales [30]. Los sistemas SCADA varían en capacidad, soportando desde unos pocos miles hasta cientos de miles de canales de Input/Output (I/O). La mayoría de los proveedores de SCADA han migrado a plataformas como *NT*, con algunos productos también compatibles con *Linux*. Esta tecnología permite la monitorización en tiempo real de campos industriales y facilita la toma de decisiones críticas mediante la adquisición y presentación de datos, comunicación en red y control de procesos [31].

2.3.2. Servidor Risk Aware Consensual Kink (RACK)

Los servidores RACK, diseñados para instalarse en RACKs de servidores, se utilizan en centros de datos y entornos empresariales de computación por su alta escalabilidad, fiabilidad y eficiencia [32]. Estos servidores maximizan la densidad de equipos en espacios limitados, permitiendo la instalación de múltiples unidades en un solo RACK. Además, contribuyen a la optimización del uso del espacio en centros de datos. También, ofrecen capacidades diversas, desde procesamiento y memoria hasta almacenamiento y redes. Sin mencionar que su eficiencia destaca en la facilidad de montaje seguro de componentes de *hardware* que provee [33].

2.3.3. Interfaz de programación de aplicaciones (API)

Una *Application Programming Interface* (API) es un conjunto de definiciones y protocolos que se usa para diseñar e integrar el software de las aplicaciones. Las API permiten que productos y servicios se comuniquen sin necesidad de conocer los detalles de su implementación, lo que simplifica el desarrollo de aplicaciones y ahorra tiempo y dinero. Ofrecen flexibilidad, simplifican el diseño, la administración y el uso de aplicaciones, y abren oportunidades de innovación, esenciales para diseñar nuevas herramientas o gestionar las existentes. Funcionan como contratos documentados, estableciendo acuerdos sobre cómo estructurar y responder a solicitudes remotas.

Las API son fundamentales para el desarrollo rápido de servicios innovadores, especialmente en aplicaciones nativas de la nube que utilizan una arquitectura de microservicios. También simplifican la conexión de la infraestructura interna y permiten compartir datos con clientes y usuarios externos, con API públicas que amplían conexiones comerciales y rentabilizar datos, como es el caso de la API de *Google Maps* [34].

2.3.4. PLC

Un PLC está diseñado para ejecutar procesos de automatización en el ámbito industrial. Este dispositivo consta de un sistema operativo (*firmware*) y un entorno de programación que soporta uno o varios lenguajes. Utilizados en la automatización de procesos, los PLC han avanzado tecnológicamente, proporcionando hoy en día capacidades destacadas para la industria. La comunicación entre dos o más PLC se realiza a través de una conexión específica basada en reglas que permiten la transferencia de datos o información entre ellos. Estas reglas se conocen como protocolos de comunicación. Entre los protocolos más comunes utilizados por PLC se incluyen *Profibus*, *Fieldbus*, *Modbus*, *Devicenet* e *Interbuss*. [35]

2.3.5. Protocolo *Modbus*

El protocolo *Modbus* es un protocolo de comunicación diseñado con una arquitectura maestro/esclavo o cliente/servidor. Su principal objetivo es facilitar la comunicación rápida y fiable entre dispositivos de automatización y de campo. [36] La definición del protocolo *Modbus*/Transmission Control Protocol (TCP) permite un diseño sencillo de un cliente [37].

En un sistema SCADA, el protocolo *Modbus* permite a los PLC, sensores y actuadores comunicarse con el *software* de supervisión. El sistema SCADA actúa como maestro, enviando consultas a los dispositivos esclavos y procesando las respuestas para monitorear y controlar el sistema en tiempo real. Este flujo de comunicación asegura que los operarios tengan acceso constante a los datos de interés para mantener el control del proceso industrial [38].

Existen dos variantes principales del protocolo: *Modbus* Remote Terminal Unit (RTU) y *Modbus* TCP/Internet Protocol (IP). *Modbus* RTU utiliza comunicación serial (RS-232, RS-485) y los mensajes se envían en un formato binario compacto. Mientras que, *Modbus* TCP/IP utiliza redes *Ethernet* y los mensajes se encapsulan en paquetes TCP/IP. Adicionalmente, todas las solicitudes son enviadas vía TCP sobre el puerto registrado 502 y emplean comunicación *half-duplex* sobre una conexión dada [38]. En el laboratorio de la Micro-Red de la Universidad de Cuenca, el sistema SCADA utiliza el protocolo *Modbus* TCP/IP para la comunicación.

2.3.5.1. Mensajes del Protocolo *Modbus*

Los procesos principales que realiza el sistema SCADA consisten en enviar una solicitud *Modbus* (*Query*) y recibir una respuesta *Modbus* (*Response*).

- Solicitud *Modbus* (*Query*): Un mensaje de consulta [38] es enviado por el maestro e incluye:

- **Dirección del Dispositivo (1 byte)**: Identifica el esclavo al que se dirige la consulta.
- **Código de Función (1 byte)**: Indica la operación a realizar (por ejemplo, leer o escribir).
- **Datos (n bytes)**: Contienen la información necesaria para la operación, como la dirección de inicio y la cantidad de datos.
- **CRC (2 bytes en Modbus RTU) / Verificación de error (TCP/IP)**: Asegura la integridad del mensaje.

- Respuesta *Modbus* (*Response*):

  Un mensaje de respuesta [38] es enviado por el esclavo e incluye:

  - **Dirección del Dispositivo (1 byte)**: La misma dirección que en la consulta.
  - **Código de Función (1 byte)**: El mismo código de función que en la consulta.
  - **Datos (n bytes)**: Contienen los datos solicitados o una confirmación de la operación.
  - **CRC (2 bytes en Modbus RTU) / Verificación de error (TCP/IP)**: Asegura la integridad del mensaje.

2.3.6. Conmutadores Weidmüller

Estos *switches* gestionables proveen una conexión inteligente en red de componentes de automatización de forma sencilla y eficaz. Además, ofrecen amplios mecanismos de control para la distribución de datos y la gestión del ancho de banda para coordinar y hacer frente a los diferentes requisitos de los participantes de la comunicación en una red industrial. La configuración se basa en la web mediante una interfaz de usuario sencilla e intuitiva [39].

2.4. Redes Ópticas

La fibra óptica es un medio físico hecho de vidrio o plástico que utiliza pulsos de luz emitidos por Light Emitting Diode (LED) o láser, con velocidades de gigabits por segundo. Entre las ventajas de la fibra se encuentran la transmisión de datos a altas velocidades, mayor ancho de banda, eliminación de interferencias electromagnéticas, mejor calidad en formatos de video y sonido, alta confiabilidad y bajos niveles de atenuación. Sin embargo, también presenta desventajas, como el costo de instalación, que es más elevado en comparación con el cable de cobre o coaxial, y la necesidad de equipos especializados para su reparación [40].

Por otro lado, los *splitters* ópticos son componentes pasivos que se utilizan para dividir la señal óptica en una Red Óptica Pasiva (PON). Estos *splitters* están compuestos por una o

dos fibras de entrada y N fibras de salida, distribuyendo la potencia de la señal óptica entre ellas de manera equilibrada, lo que los define como *splitters* balanceados [41]. Además, estos componentes tienen la capacidad de ser utilizados para generar un *Bypass Físico*.

2.4.1. Topologías de Red

Una topología de red se refiere a la forma planificada de disposición de la red, que incluye nodos, conexiones y líneas utilizadas para la transmisión y recepción de datos [42]. A continuación se enlistan las topologías más utilizadas en la actualidad.

- Topología de Bus
- Topología de Anillo
- Topología de Estrella
- Topología de Árbol
- Topología de Malla

Las topologías de interés para este trabajo se describen en las Secciones 2.4.1.1 y 2.4.1.2.

2.4.1.1. Topología de Anillo

En una topología de anillo, todos los nodos están interconectados formando un círculo. En este sistema, cada nodo procesa las tramas por turno y, si no son para él, las pasa al siguiente nodo [43]. Además, cada nodo regenera la señal, aunque un fallo en uno afecta a toda la red [44].

2.4.1.2. Topología de Árbol:

La topología de árbol es una combinación de las topologías de bus y estrella, que permite la conexión de varios servidores. Esta red se organiza a partir de un punto de enlace troncal del cual se derivan los nodos. Existen dos subclases: el árbol binario, donde cada nodo se divide en dos enlaces, y el árbol *backbone*. Este último tiene un tronco principal llamado *backbone* que distribuye información a todos los nodos conectados [42].

2.4.2. *Bypass* Físico

Un *bypass* físico en una red se refiere a la configuración de una vía alternativa o un camino redundante que permite el flujo de datos en caso de un fallo en la red principal. Esto implica que, durante el funcionamiento normal, el tráfico de datos sigue su curso a través de la red principal, pero si surge algún problema, como un fallo en un enlace o un dispositivo de red, el tráfico se redirige por la vía alternativa [41].

2.4.3. Protocolos de redundancia de red

En entornos industriales y de automatización, la confiabilidad y disponibilidad de las redes son relevantes para mantener operaciones ininterrumpidas y seguras. Para garantizar esto, se utilizan protocolos y tecnologías especializados como *Spanning Tree Protocol* (STP), *Rapid Spanning Tree Protocol* (RSTP), *Turbo Ring* y *Turbo Chain*. Estos están diseñados para proporcionar redundancia y una rápida recuperación en caso de fallos, asegurando una infraestructura resistente capaz de manejar situaciones adversas sin afectar el servicio de manera significativa.

- **Protocolo *Spanning Tree* (STP)**

  El protocolo STP fue concebido para solucionar el problema de las tormentas de difusión, donde los mensajes de difusión se multiplican, sobrecargando la red. Este algoritmo, definido por el estándar IEEE 802.1D y sus actualizaciones posteriores, gestiona la presencia de bucles en las topologías de red generados por conexiones redundantes. Además, permite que los dispositivos de interconexión activen o desactiven automáticamente los enlaces de conexión para asegurar una topología de red sin bucles [45].

- **Protocolo *Rapid Spanning Tree* (RSTP)**

  Un *Spanning Tree* que opera en modo STP requiere hasta un minuto para reconstruirse luego de un cambio en la topología o la configuración. En contraste, el protocolo RSTP ofrece una recuperación más veloz de la conectividad tras una falla en un puente, puerto de puente o enlace. Para lograrlo, RSTP integra roles de puerto en la determinación de los estados de los puertos, y permite que los puentes adyacentes reconozcan señales en un enlace punto a punto que indiquen que un puerto desea pasar al modo de reenvío [46].

- **Protocolo *Turbo Ring***

  Turbo Ring ofrece redundancia de red mediante una configuración en anillo que conecta todos los *switches* [47]. Esta tecnología resuelve el problema de los bucles infinitos al utilizar *switches* gestionados, donde los puertos *Ethernet* se configuran en modo de desactivación, para activarse en caso de detectar una falla.

Este protocolo detecta y se recupera de fallas en la ruta principal en menos de 20 milisegundos, mucho más rápido que los protocolos convencionales como RSTP. Además, proporciona la capacidad de configurar alarmas para notificar al personal de soporte en caso de fallo de la red, asegurando la continuidad de la transmisión de datos [48].

- **Protocolo *Turbo Chain***

  Turbo Chain es una ampliación de *Turbo Ring*. El protocolo facilita la interconexión de anillos, conexiones dobles y anillos redundantes. La configuración topológica de *Turbo Chain* es similar a la de *Turbo Ring*, pero es capaz de integrarse en una red preexistente, lo que posibilita que redes independientes operen mientras permanecen enlazadas a la red principal. Finalmente, la importancia de *Turbo Ring* y *Turbo Chain* radica en su capacidad para brindar redundancia de red sin requerir hardware adicional ni gastos de cableado [48].

2.5. Métodos de investigación

Para establecer los lineamientos de investigación que guían este trabajo de titulación, se han revisado los principales métodos de investigación considerados relevantes para estudiar los antecedentes de este proyecto y elaborar un plan que cumpla con los objetivos propuestos.

2.5.1. Método de estudio de caso

El método de estudio de caso es un enfoque cualitativo, particularmente adecuado para entender fenómenos complejos y explorar temas donde los conocimientos previos son limitados. La elección de este método de estudio, en la parte inicial del presente trabajo de titulación, no solo permite una mejor comprensión del problema en estudio, sino que también contribuye al desarrollo teórico y nuevas etapas de investigación [49]. En el artículo [50] se subraya la rigurosidad del método de estudio de caso, destacando sus múltiples ventajas.

A continuación, se explica cada una de ellas, resaltando su aplicación en el presente trabajo de titulación:

- El método es idóneo para investigar fenómenos donde el objetivo es entender cómo y por qué ocurren.
- El método permite una investigación profunda sobre un tema concreto.
- El método es adecuado para temas donde las teorías existentes no son suficientes.
- El método permite estudiar fenómenos desde múltiples ángulos.

- El método facilita una exploración más profunda y un conocimiento más amplio sobre cada fenómeno, lo que permite la aparición de nuevas señales y hallazgos sobre los temas emergentes.

2.5.2. Método empírico

El método empírico permite el estudio de fenómenos observables y la confirmación de hipótesis y teorías a través de la observación, la experimentación y/o la medición. Este tipo de métodos son capaces de identificar las características fundamentales y las relaciones del objeto de estudio, facilitando una comprensión directa y objetiva de los fenómenos ([51]).

La observación científica, como uno de los métodos empíricos clave, posibilita la percepción directa del objeto de investigación para comprender el funcionamiento actual del sistema SCADA y los elementos congregados en la red. Este método es útil para diagnosticar problemas potenciales durante los procesos de mantenimiento de los equipos de la Micro-red de la Universidad de Cuenca, así como para validar hipótesis, predecir tendencias y desarrollar estrategias para mejorar la disponibilidad del sistema.

Por otro lado, el método experimental, también considerado dentro de los métodos empíricos, ofrece la oportunidad de crear condiciones controladas para investigar las propiedades y relaciones del objeto de estudio. En esta investigación, el diseño de experimentos se utiliza para evaluar la efectividad de diferentes soluciones propuestas para mejorar la disponibilidad del sistema SCADA durante los procesos de mantenimiento.

2.5.3. Método de diseño de experimentos

El método de diseño de experimentos o Design of experiments (DoE), por sus siglas en inglés, se ha empleado para la realización de varios experimentos importantes. Esta metodología, consiste en realizar una serie de pruebas donde se inducen cambios deliberados en las variables de entrada de un proceso, permitiendo observar e identificar las causas de los cambios en la respuesta de salida [52]. De esta manera, los experimentos se resumen en los siguientes:

- Verificar la configuración de los *switches Weidmüller*.
- Comprobar que la desconexión de un PLC no afecta el funcionamiento de los demás PLC.
- Confirmar que el programa se detiene a pesar de una configuración física correcta.

El DoE [52] ofrece múltiples beneficios que son especialmente relevantes para los experimentos propuestos:

- El método DoE facilita la identificación precisa de las causas de los cambios en la respuesta de salida al inducir cambios deliberados en las variables de entrada.

- El método diseña experimentos que contribuyen a respondes las siguientes preguntas:
  - ¿Cuáles son los objetivos de la experimentación?
  - ¿Qué variables se deben incluir?
  - ¿Cuántas pruebas se deben realizar?
  - ¿Qué datos recoger y cómo recogerlos eficientemente?

En la Sección 3.1 se explica cada experimento, respondiendo las preguntas del método DoE.

3. Metodología

Este capítulo expone la estrategia empleada para alcanzar los objetivos delineados en el Capítulo 1. Los datos preliminares sobre el estado y funcionamiento del sistema *Supervisory Control and Data Acquisition* (SCADA) se han recopilado a través de discusiones con los técnicos del laboratorio de Micro-Red de la Universidad de Cuenca. En este sentido, se detalla la aplicación de los métodos seleccionados en la revisión literaria para identificar posibles soluciones y ejecutarlas.

Para el desarrollo de este trabajo de titulación, se ha optado por una investigación mixta que incorpora tres métodos interrelacionados: el método de estudio de caso (Sección 2.5.1), el método empírico (Sección 2.5.2) y el método de diseño de experimentos (Sección 2.5.3). La elección de estos métodos en conjunto se fundamenta en la complejidad para identificar los posibles problemas que generan la inoperabilidad del sistema SCADA en procesos de mantenimiento y abordarlos utilizando soluciones pertinentes en base definición de experimentos a realizar.

Para abordar la interrupción del sistema SCADA, tema que demanda un análisis detallado debido a la falta de teorías existentes acerca de la causa del problema, y a los limitados conocimientos previos; utilizamos el método de estudio de caso. Mediante este método, se investigaron las posibles causas del problema desde varios enfoques. Al aplicar este método de investigación, surgieron las siguientes hipótesis:

- La carencia de gestión de la redundancia de red a través de los *switches Weidmüller* provoca la falta de disponibilidad del sistema SCADA.
- La desconexión de un equipo en la red física del anillo de fibra óptica es la causa principal de la falta de disponibilidad del sistema SCADA.
- La programación del sistema SCADA en el *software* Laboratory Virtual Instrument Engineering Workbech (LabVIEW) resultan en la falta de disponibilidad del sistema SCADA.

Para validar las hipótesis planteadas, con ayuda del método empírico se exploró a fondo la dinámica de los equipos con el sistema SCADA, y con esto se han diseñado diferentes experimentos en la Sección 3.1, utilizando el método Design of experiments (DoE). La validación de las hipótesis en base a los resultados de los experimentos, detallados a continuación, se presentan en el Capítulo 4.

3.1. Diseño y ejecución de los experimentos

A continuación, se detallan los experimentos diseñados y las actividades planteadas para realizar cada uno de ellos.

3.1.1. Experimento: Verificar la correcta gestión de redundancia de la red en la configuración de los *switches Weidmüller*

En este experimento se busca analizar la configuración actual de la gestión de redundancia de red en los *switches Weidmüller*. Además, se ha planteado la posibilidad de configurar un puerto espejo para analizar el comportamiento del tráfico que recibe el sistema SCADA a través de los *switches Weidmüller* ante la ocurrencia de un evento. Para ello, se utilizará el manual del fabricante disponible en [53] para explorar esta funcionalidad y sus características de redundancia. De esta manera, en caso de que el tráfico no pueda continuar debido a la desconexión de un elemento, la red se restablecerá utilizando la redundancia disponible. Este experimento se fundamenta en la hipótesis de que la inoperabilidad del sistema es provocada por la inhabilitación del anillo de fibra debido a la desconexión de uno de sus equipos.

Las pruebas se enfocarán en mantener la comunicación entre los equipos, incluso durante una desconexión. Se interactuará con el dispositivo para comprender y, si es necesario, configurar la redundancia, permitiendo que los *switches* recuperen el tráfico perdido. Se ha tomado como referencia el levantamiento de direcciones Internet Protocol (IP) del trabajo "Diseño e implementación de una Arquitectura de Ciberseguridad para la Micro-red de la Universidad de Cuenca"[54], para realizar un seguimiento de conexión de los equipos de la red *Operational Technologies* (OT).

Durante las pruebas, se recopilarán datos de tráfico en los *switches* utilizando herramientas de análisis como *Wireshark* y *tcpdump*. Estos datos proporcionarán información sobre los protocolos de comunicación principales utilizados, permitiendo identificar aquellos paquetes útiles para deducir los efectos de la desconexión en la red OT y su impacto en el funcionamiento del sistema SCADA.

El proceso de desarrollo del experimento incluye los siguientes pasos:

1. Ingreso a la interfaz del *switch* utilizando la dirección IP asignada y la contraseña por defecto $\mathrm{Detmold}$ o sugerida en el compendio de claves de acceso del laboratorio.
2. Exploración de prueba del *switch* con IP $192.\mathrm{X}.\mathrm{X}.\mathrm{X}$.

   La Figura 3.1 muestra la configuración de este *switch*, destacando sus características principales.

Figura 3.1: Configuración del *Switch Weidmüller*

3. Verificación del estado de la configuración de redundancia en los *switches* de la red OT.

La Figura 3.2 presenta el estado actual del protocolo de redundancia de los dispositivos que integran la red OT. En la sección *Current Status*, se observa que el protocolo de redundancia activo es el *Turbo Ring*. Además, los puertos redundantes están en estado *Forwarding* tanto para el primer como para el segundo puerto, lo que indica que están transmitiendo datos correctamente. Por lo tanto, la redundancia está activa y los puertos redundantes se encuentran operativos.

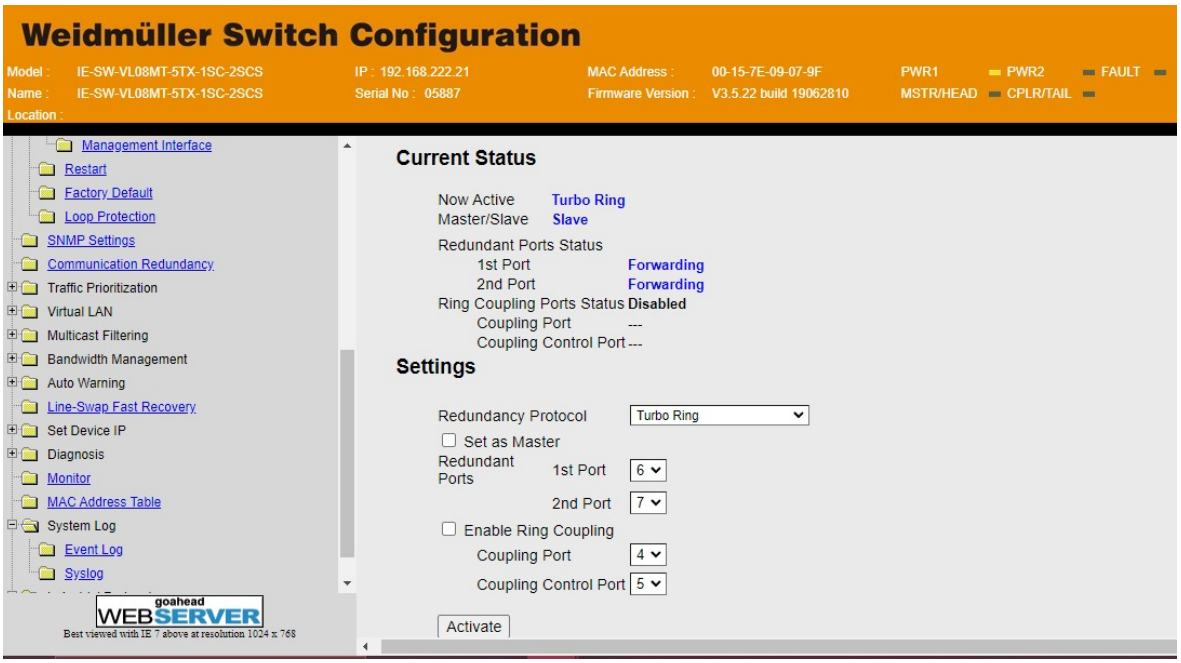


Figura 3.2: Interfaz de Usuario del *Switch Weidmüller*

4. Revisión de factibilidad de configuración del *port mirroring* en el *switch*.

   Al revisar las características configurables del *switch*, se expone la factibilidad de realizar un puerto espejo con la configuración mostrada en la Figura 3.3.

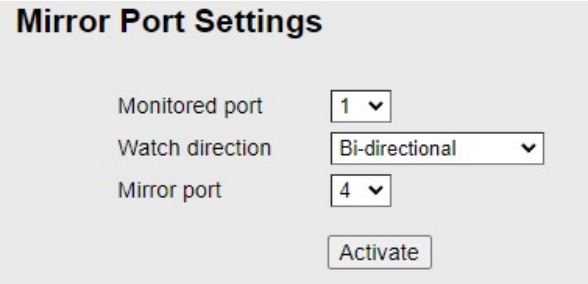


Figura 3.3: Configuración de puerto espejo en el *Switch Weidmüller*

5. Captura del tráfico utilizando el puerto espejo configurado en el *switch*.

   Utilizando el puerto físico 4 del *switch* y una conexión por *Ethernet* para capturar dicha interfaz utilizando el *software Wireshark*.

En este punto, se establece una conexión *Ethernet* hacia el ordenador con el *sniffer* de captura de paquetes. Se utiliza la dirección http://192.X.X.X/ para acceder a la interfaz del *switch*. Con las credenciales disponibles en la recopilación de acceso del laboratorio, se ingresa a la configuración del *switch* y se captura el tráfico de la interfaz conectada.

Los datos capturados, se muestran a continuación.

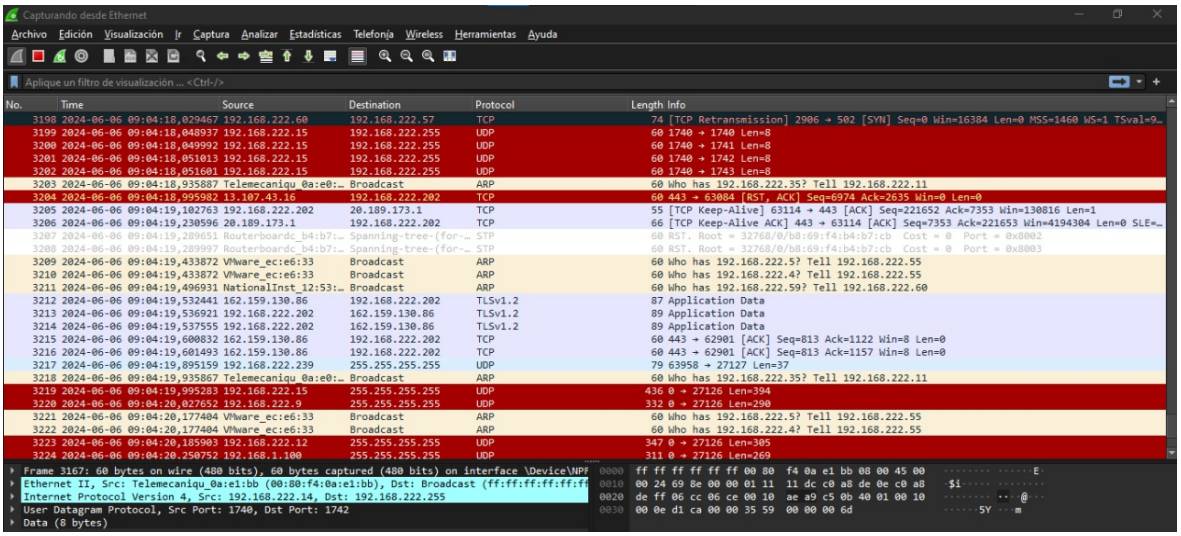


Figura 3.4: Captura de paquetes del *Switch Weidmüller* principal

En la Figura 3.4, se observan principalmente los protocolos Transmission Control Protocol (TCP), User Datagram Protocol (UDP) y Address Resolution Protocol (ARP). Las direcciones IP de origen y destino más frecuentes son 192.X.X.X y 192.X.X.X para el origen, y 192.X.X.X y 192.X.X.X para el destino. Mismas que pertenecen a los equipos congregados al sistema del laboratorio. Además, se detecta tráfico UDP recurrente en el puerto 1740, y en el caso del tráfico TCP, se notan retransmisiones y Acknowledgment (ACK)s, lo que sugiere problemas de red o congestión.

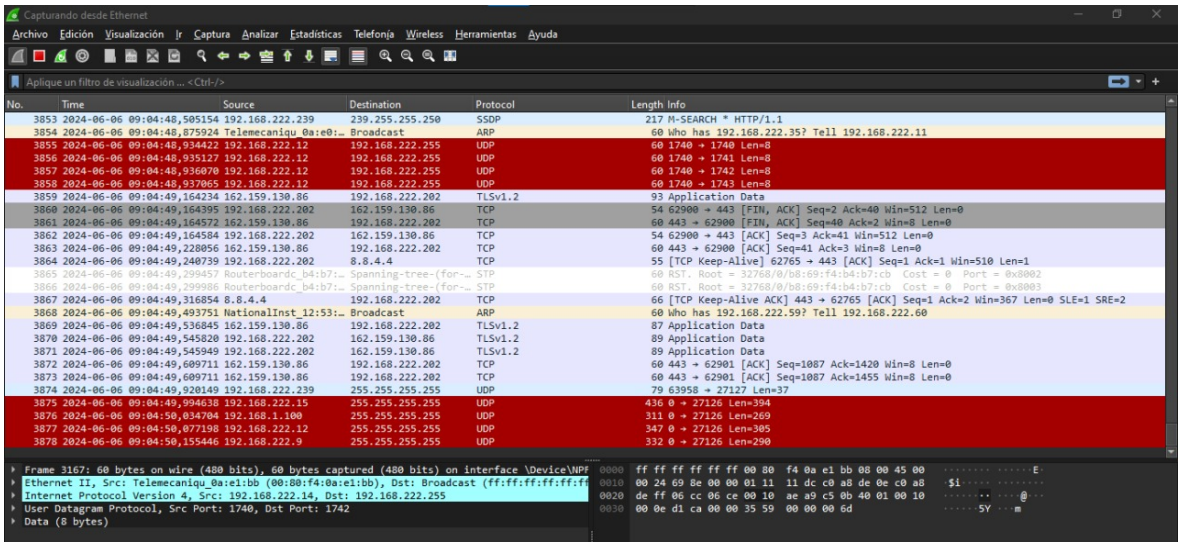


Figura 3.5: Captura de paquetes del *Switch Weidmüller* principal

En tanto que, la Figura 3.5 muestra los protocolos Simple Service Discovery Protocol (SSDP), ARP, UDP, TCP y Transport Layer Security (TLS)v1.2. Las direcciones IP más frecuentes son 192.X.X.X y 192.X.X.X como origen, y 239.X.X.X (*multicast*) y 192.X.X.X como destino. El tráfico UDP sigue siendo notable en el puerto 1740. En el tráfico TCP, se observan ACKs y FIN, lo que indica el cierre de conexiones TCP.

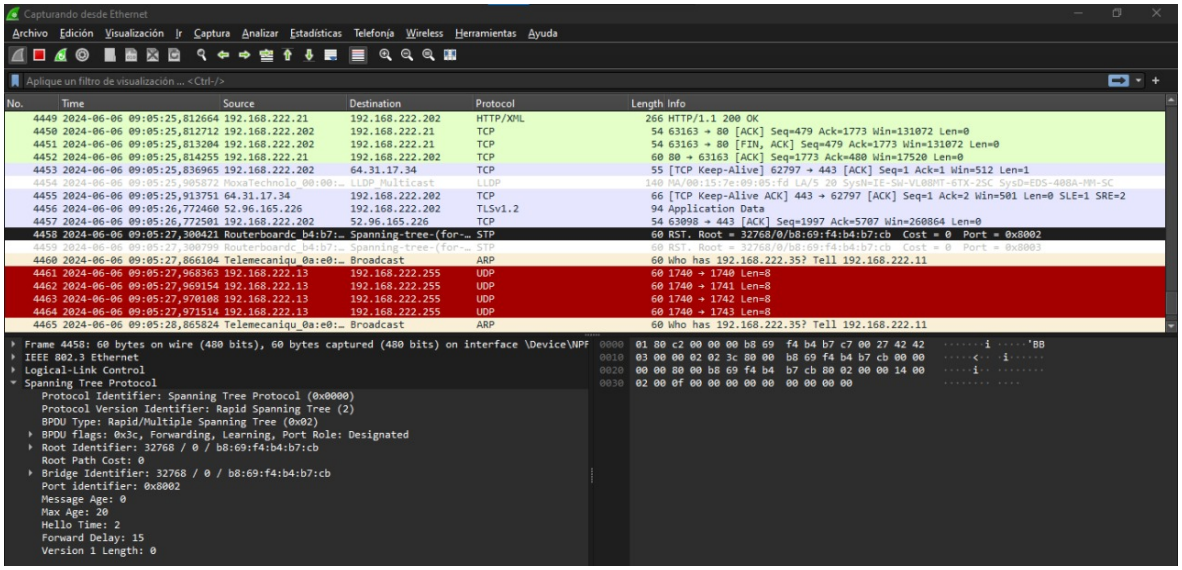


Figura 3.6: Captura de paquetes del *Switch Weidmüller* principal

De la Figura 3.6 presentada, los protocolos coherentes con el objetivo de análisis son ARP, UDP, TCP y TLSv1.2. Las direcciones IP de origen y destino más frecuentes son 192.X.X.X. Aquí, el tráfico TCP notable incluye paquetes *Keep-Alive* y ACKs, lo que es común en las conexiones Hypertext Transfer Protocol (HTTP) prolongadas.

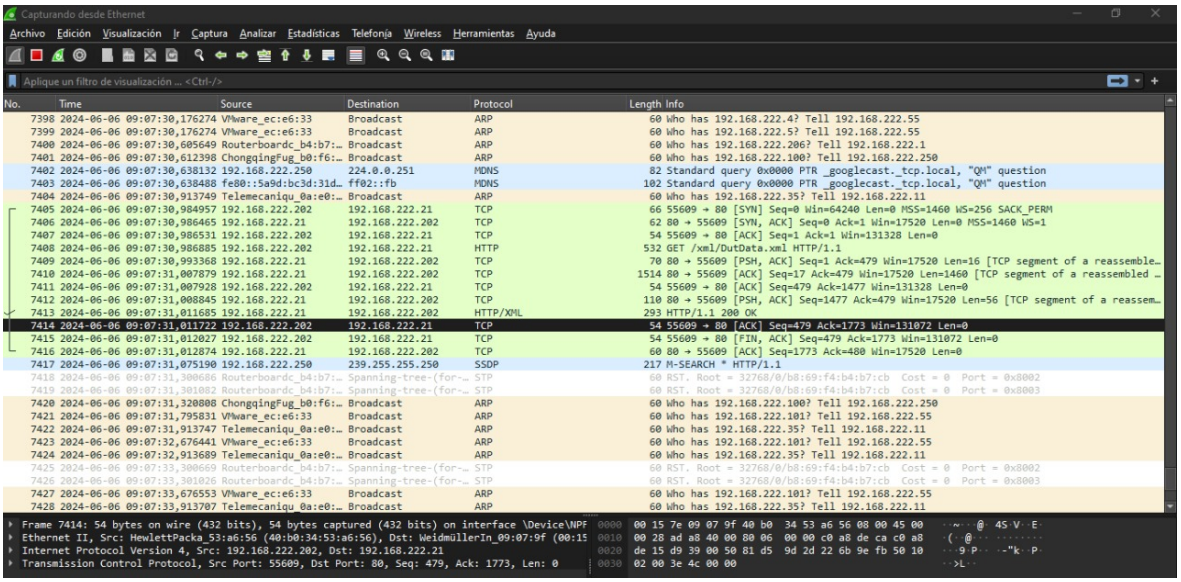


Figura 3.7: Captura de paquetes del *Switch Weidmüller* principal

Por último, la Figura 3.7 incluye los protocolos ARP, TCP y HTTP. Las direcciones IP de origen frecuentemente provienen de dispositivos *VMware* y otras con direcciones 192.X.X.X, mientras que los destinos frecuentes son 192.X.X.X (*broadcast*), 192.X.X.X y 192.X.X.X (SSDP). El tráfico TCP incluye segmentos HTTP, ACKs y Synchronous Idle Mode (SYN), lo cual es indicativo de sesiones activas y establecimiento de conexiones.

3.1.2. Experimento: Verificar si la desconexión de un *Programmable Logic Controller* (PLC) afecta el funcionamiento habitual de los demás PLC

El experimento tiene el objetivo de comprobar si el problema de la falta de disponibilidad está relacionada con la red física de la Micro-red, mediante la evaluación del comportamiento de los PLC al momento de desconectar uno de ellos.

Con las primeras observaciones realizadas en este método, se evidencia lo siguiente: los PLC son los únicos elementos que se comunican con el sistema SCADA a través de mensajes *Modbus*. Esta afirmación motiva el análisis de los paquetes que utilizan el protocolo *Modbus*, específicamente la cantidad de paquetes por minuto que cada PLC envía al sistema SCADA. El planteamiento se basa en que, si se desconecta un PLC de su fuente de energía, la cantidad de paquetes por minuto de dicho PLC será 0, mientras que la cantidad de paquetes por minuto de los demás PLC debe mantener su valor habitual. Estas pruebas se realizan en los PLC de las *Application Programming Interfaces* (APIS) conectadas al sistema SCADA de la Micro-red. Esta evaluación permitirá determinar si la desconexión de un PLC afecta o no el funcionamiento normal de los otros PLC en la red.

Para obtener los datos necesarios para realizar las pruebas, se ha implementado, como máquina virtual en *VMware Workstation*, un servidor *Ubuntu 22.04 LTS* alojado en el servidor *Lenovo System X 3550 M5*. En este servidor se encuentra la máquina virtual del servidor *Windows* que contiene el programa en LabVIEW del sistema SCADA. En la Tabla 3.1, se detallan las características de la máquina que contiene el servidor local *Graylog*.

Nombre	Nombre Servidor	Sistema Operativo	Disco Duro	Memoria RAM
TT-SDN	sdn_tesis	Ubuntu 22.04 LTS	30GB	6GB

Tabla 3.1: Características del Servidor

Esta máquina virtual actuará como servidor de registros del sistema SCADA, cuyos datos serán dedicados a definir la disponibilidad o inactividad de los PLC congregados. Para ello, permitirá el funcionamiento del servidor a través de contenedores *docker*, lo que facilita su adaptación a investigaciones futuras y a las necesidades del laboratorio. El proceso de instalación y configuración del servidor *Ubuntu*, de la plataforma de código abierto *docker*, y del servidor *Graylog* se presentan en los Anexos 6.3, 6.3 y 6.3. El servidor para la gestión de *Logs* obtiene los mensajes *Modbus* en tiempo real, que se requieren para verificar la hipótesis planteada. Es importante mencionar que, se capturan los mensajes *Modbus*: *Query* y *Response*. El proceso para capturar el número de mensajes *Modbus* recibidos por minuto es el siguiente:

1. **Configuración de la IP Estática en el servidor *Ubuntu 22.04 LTS***

   Para empezar, se elige una dirección IP dentro del rango de 192.X.X.X a 192.X.X.X. La IP escogida es 192.X.X.X. Para realizar esta configuración en *Ubuntu*, se usa el archivo de configuración de red ubicado en /etc/netplan/00-installer-config.yaml. En este archivo, se usó el siguiente contenido:

```
network:
  version: 2
  ethernets:
    ens32:
      addresses:
        - 192.168.222.66/24
      nameservers:
        addresses:
          - 8.8.8.8
        search: []
      routes:
        - to: default
          via: 192.168.222.1
    ens36:
      match:
        macaddress: ”00:0c:29:ec:e6:33”
```

Código 1: Código utilizado para la configuración de una IP estática en *Ubuntu*

En el Código 1 se definió la versión de la configuración, y se especificó la configuración para dos interfaces de red: ens32 y ens36. En la interfaz ens32 se asignó la dirección IP estática 192.X.X.X con una máscara de subred /24, se configuró el servidor Domain Name System (DNS) utilizando la dirección IP 8.8.8.8, y se estableció la puerta de enlace predeterminada en 192.X.X.X. Por otro lado, para la interfaz ens36 se estableció la dirección Media Access Control (MAC) como 00:0c:29:X:X:X, que es la misma dirección MAC del adaptador de red del servidor *Windows* donde se encuentra el sistema SCADA, con la finalidad de capturar los paquetes *Modbus* entrantes.

Después de realizar estos cambios en el archivo de configuración, se ejecuta el comando sudo netplan apply para aplicar los cambios de configuración de red. De esta manera, la interfaz ens32 toma una dirección IP estática, y la configuración de red se mantiene persistente incluso después de reiniciar el sistema.

2. **Desarrollo e implementación del programa en *Python***

A continuación, se capturan los mensajes tipo *Modbus*, para clasificarlos en sus dos tipos: *Query* y *Response*. Posteriormente, se cuenta el número de mensajes por minuto para enviarlos como *logs* a *Graylog*. Para esta parte, se usó el siguiente código en *Python*, llamado sniffer.py:

```
"""
Universidad de Cuenca
Facultad de Ingeniería
Trabajo de Titulación para Ingeniero de Telecomunicaciones
Tesistas: Lourdes Gutiérrez, Claudia Padilla
Junio 2024

Documentación: Código de Sniffeo del servidor SCADA, protocolo Modbus.
Para identificar la actividad de los PLCs.
"""
from scapy.all import sniff, TCP, IP
import logging
from graypy import GELFUDPHandler
import threading
import time

# Diccionario de mapeo de direcciones IP a nombres
mapeo_ips = {
    "192.168.222.9": "PLC apis1",
    "192.168.222.11": "PLC1 apis2",
    "192.168.222.12": "PLC2 apis2",
    "192.168.222.13": "PLC3 apis2",
    "192.168.222.14": "PLC4 apis2",
    "192.168.222.15": "PLC apis3",
    "192.168.222.55": "SCADA"
}

# Diccionario para mantener el recuento de Rs y ADURequests para cada PLC
recuento_plc_minuto = {plc: {"R": 0, "Q": 0} for plc in mapeo_ips.values()}

# Función para manejar cada paquete capturado
def manejar_paquete(packet):
    global recuento_plc_minuto
```

```
34
35      tipo_mensaje = None
36
37      if TCP in packet:
38          ipsrc = packet[IP].src
39          ipdest = packet[IP].dst
40
41          if packet[TCP].dport == 502:
42              tipo_mensaje = "Q"
43          elif packet[TCP].sport == 502:
44              tipo_mensaje = "R"
45
46          if tipo_mensaje:
47              plc_src = mapeo_ips.get(ipsrc, "Desconocido")
48              plc_dest = mapeo_ips.get(ipdest, "Desconocido")
49
50              if plc_src in recuento_plc_minuto:
51                  recuento_plc_minuto[plc_src][tipo_mensaje] += 1
52              if plc_dest in recuento_plc_minuto:
53                  recuento_plc_minuto[plc_dest][tipo_mensaje] += 1
54
55  # Función para enviar el recuento por minuto y reiniciar el recuento
56  def enviar_conteo_por_minuto():
57      global recuento_plc_minuto
58      mensaje_por_minuto = "Recuento de queries (Q) y responses (R) para todos los PLCs por minuto:\n"
59      for plc_src, stats in recuento_plc_minuto.items():
60          mensaje_por_minuto += f"Q {plc_src}: {stats['Q']}, R {plc_src}: {stats['R']}\n"
61      logger.debug(mensaje_por_minuto)
62      reiniciar_recuento_por_minuto()
63
64  # Función para reiniciar el recuento por minuto
65  def reiniciar_recuento_por_minuto():
66      global recuento_plc_minuto
67      for plc_stats in recuento_plc_minuto.values():
68          plc_stats["R"] = 0
69          plc_stats["Q"] = 0
70
71  # Set logs
72  logger = logging.getLogger("gelf")
73  logger.setLevel(logging.DEBUG)
74
75  handler = GELFUDPHandler(host="192.168.222.66", port=5514)
76  logger.addHandler(handler)
77
78  # Iniciar el proceso de captura y envío por minuto
79  threading.Thread(target=capturar_y_enviar_por_minuto).start()
```

Código 2: Código utilizado para la captura, clasificación y conteo de los paquetes tipo *Modbus*

En el Código 2 se implementa la captura de paquetes de red utilizando *scapy*. Primero, se define un diccionario para mapear direcciones IP a nombres de PLC y un diccionario para llevar el recuento de *queries* (Q) y *responses* (R) por minuto para cada *PLC*. La función manejar_paquete analiza cada paquete capturado, determina el tipo de mensaje basándose en los puertos *Modbus* (502), y actualiza el recuento correspondiente.

También, se configuró un *logger* para enviar los recuentos cada minuto al servidor *Graylog* utilizando *graypy*, con la función $\mathrm{enviar\_conteo\_por\_minuto}$ que se ejecuta periódicamente en un hilo separado, y la función $\mathrm{reiniciar\_recuento\_por\_minuto}$ que reinicia el recuento para el siguiente minuto. Finalmente, se inicia la captura de paquetes con la función *sniff* de *scapy*, permitiendo monitorear y registrar la actividad de los PLC en el sistema SCADA en tiempo real.

3. **Configuración de reglas en el *Firewall* de *Windows* y *Ubuntu***

   Es fundamental permitir la salida de información a través de los puertos utilizados para recibir mensajes en el servidor *Graylog*, mismos que se listan a continuación.

   - 9000:9000
   - 3514:3514/udp
   - 4514:4514/udp
   - 5514:5514/udp
   - 6514:6514/udp
   - 7514:7514/udp

   Para ello, se han definido las siguientes reglas de salida en *Windows* que incluyen los puertos utilizados.

   - 5514: Envío de mensajes *Modbus* capturados, utilizando el puerto 502.
   - 514:514/udp
   - 4514:4514/udp
   - 5514:5514/udp

   Estas reglas de salida permiten el envío de datos hacia el servidor de registros y su activación incluye los pasos enumerados en seguida.

   a) Abrir el *Firewall* de *Windows*.

   b) Establecer reglas de entrada o salida como sea pertinente.

   En este caso, *Windows* solo requiere el establecimiento de reglas de salida, ya que por ellas se enviarán los datos hacia el servidor de registros.

   c) Nominación y establecimiento de puertos.

   Se debe establecer el tipo de protocolo utilizado y el puerto pertinente. En este trabajo de titulación, todos los datos enviados utilizan el protocolo de comunicación UDP, ya que la seguridad no es el objetivo principal. Las reglas para el envío de datos principales se muestran en la Figura 3.10. A continuación, se presentan ejemplos de configuración para dos puertos utilizados:

Figura 3.8: Regla de salida habilitada para el puerto 514

Figura 3.9: Regla de salida habilitada para el puerto 5514

d) Reglas de salida definidas en *Windows*.

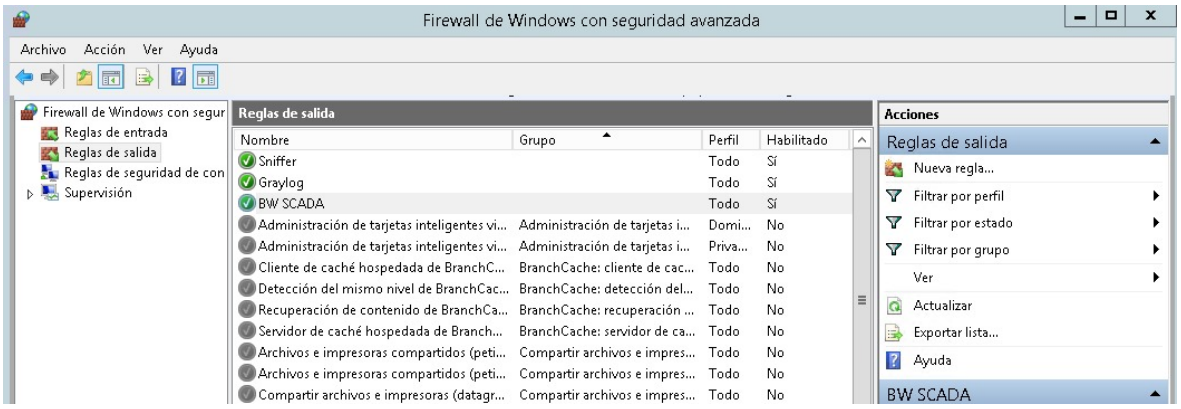


Figura 3.10: Reglas de salida definidas en el servidor SCADA

4. **Configuración de tareas programadas para captura de paquetes en *Windows***

   Para la ejecución del código de envío, se ha creado una nueva tarea en el programador de tareas de *Windows*. A continuación, se explica la configuración de las secciones de cada tarea.

   En la Sección General, se asignó el nombre *Sniffer* a la tarea programada. En cuanto a las opciones de seguridad, se configuró la tarea para que se ejecute tanto si el usuario ha iniciado sesión como si no. Esto asegura que el programa se ejecute independientemente del estado de la sesión del usuario, garantizando una captura de paquetes continua. La configuración de Sección se presenta en la Figura 3.11.

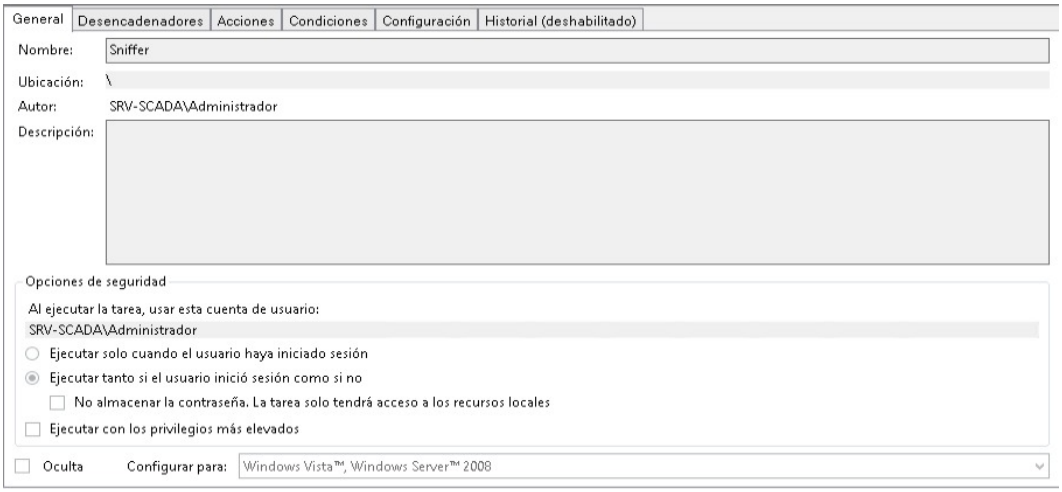


Figura 3.11: Configuración de la Sección General, del programador de tareas

En la Sección Desencadenadores, se configuró la tarea para que se inicie al arrancar el sistema. Esto significa que cada vez que el sistema se reinicia o se enciende, la tarea se activará automáticamente, asegurando que el *sniffer* de paquetes comience a operar sin necesidad de intervención manual. La configuración de Sección se presenta en la Figura 3.12.

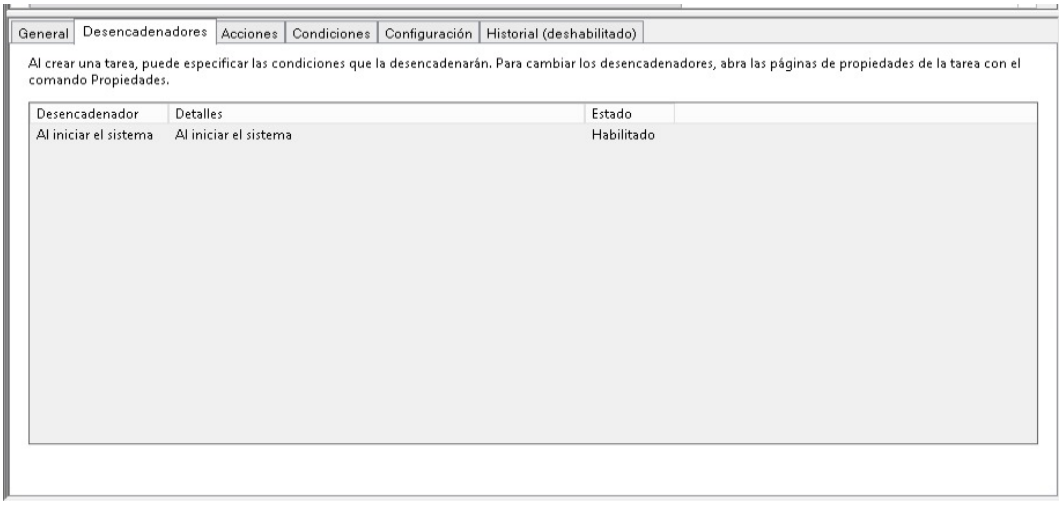


Figura 3.12: Configuración de la Sección Desencadenadores, del programador de tareas

En la Sección Acciones se configura la tarea para Iniciar un programa. Específicamente, se ingresó la ruta del ejecutable de *Python* y la ruta del *script sniffer.py*. Esto garantiza que el programa *Python* que captura y clasifica los mensajes *Modbus* se ejecute automáticamente como parte de la tarea programada. La configuración de Sección se presenta en la Figura 3.13.

Figura 3.13: Configuración de la Sección Acciones, del programador de tareas

En la Sección Condiciones, no se configuró ninguna condición especial. La tarea se ejecutará siempre que sea activada por el desencadenador, sin depender de condiciones adicionales. La configuración de Sección se presenta en la Figura 3.14.

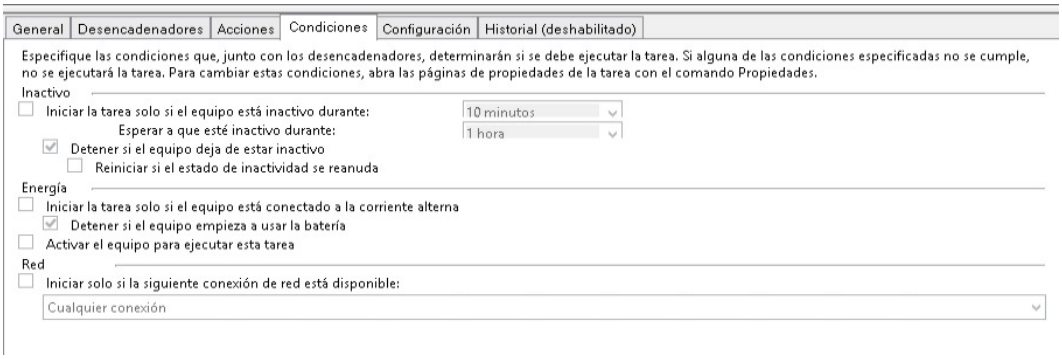


Figura 3.14: Configuración de la Sección Condiciones, del programador de tareas

En la Sección Configuración, se permitió que la tarea se ejecute a petición, proporcionando flexibilidad para ejecutarla manualmente si es necesario. Además, se configuró para que, en caso de no ejecutarse, se reinicie cada minuto, con un máximo de tres intentos de reinicio. Esta configuración mejora la resiliencia de la tarea, asegurando su ejecución fiable incluso ante fallos temporales. La configuración de esta Sección se presenta en la Figura 3.15.

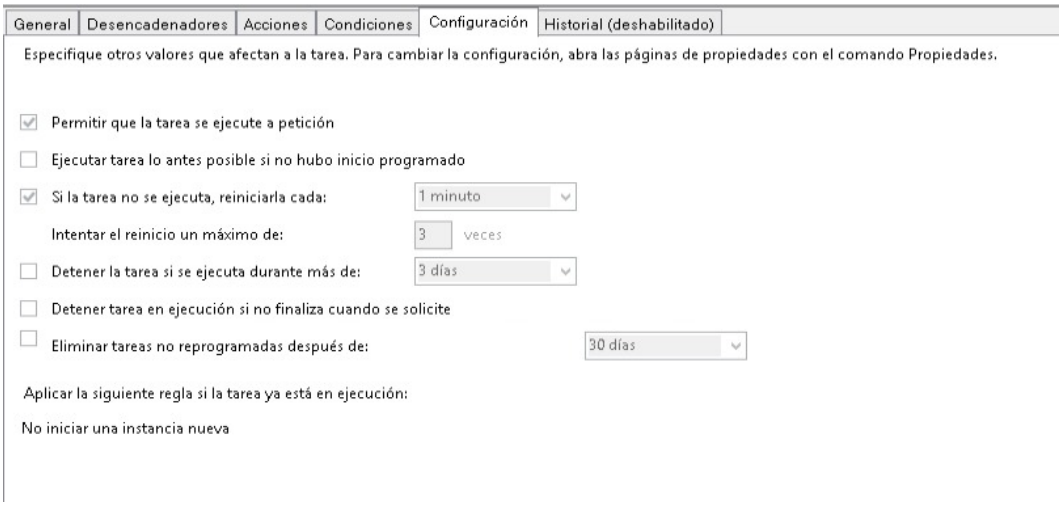


Figura 3.15: Configuración de la Sección Configuración, del programador de tareas

5. **Creación del Dashboard de los Mensajes Modbus en *Graylog***

   Los mensajes *Modbus* permitirán validar la disponibilidad de los PLC mediante la recopilación de registros. Para procesar y analizar estos datos, se han creado varios *widgets* en el servidor alojado en *Ubuntu*. El proceso de creación de *dashboards* y la recepción de mensajes a través de entradas persistentes se detalla en el entregable adjunto a este trabajo de titulación, denominado “Manual de Usuario” [55].

4. Validación de hipótesis

Este capítulo aborda la implementación del plan metodológico diseñado en el Capítulo 3 de este trabajo de titulación. Se detalla el proceso de validación de las hipótesis planteadas y se discuten los resultados obtenidos en cada experimento realizado. Así mismo, se describen los procedimientos, técnicas y herramientas empleadas. Adicionalmente, se incluye la identificación de las soluciones finales para cumplir con los objetivos propuestos.

Primero, utilizando el método de estudio de caso mencionado previamente, se ha comprendido la situación actual del sistema *Supervisory Control and Data Acquisition* (SCADA) del laboratorio de la Micro-red de la Universidad de Cuenca. Entre los hallazgos más importantes se destaca el diagrama de red mostrado en la Figura 4.1, que ilustra la lógica de acceso al sistema SCADA diseñado en Laboratory Virtual Instrument Engineering Workbech (LabVIEW) y alojado en un servidor de acceso remoto.

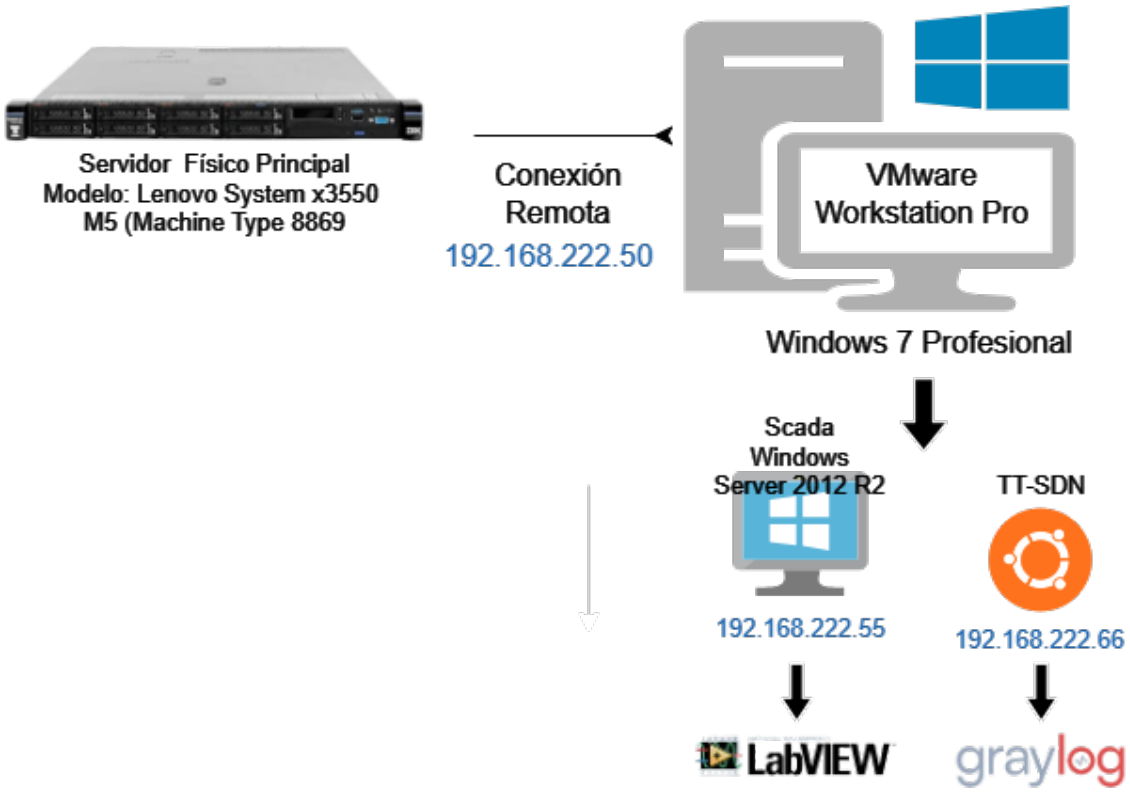


Figura 4.1: Topología Lógica de Acceso a los sistemas utilizados

Las siguientes secciones de este capítulo detallan cada una de las hipótesis formuladas anteriormente mediante el método empírico y de estudio de caso, así como los resultados de cada experimento del método de diseño de experimentos. Las teorías planteadas ante la falta de disponibilidad del sistema SCADA se resumen a continuación:

- **Hipótesis 1**: La carencia de gestión de la redundancia de red a través de los *switches*

Weidmüller provoca la falta de disponibilidad del sistema SCADA.

- **Hipótesis 2**: La desconexión de un equipo en la red física del anillo de fibra óptica es la causa principal de la falta de disponibilidad del sistema SCADA.
- **Hipótesis 3**: La programación del sistema SCADA en el *software* LabVIEW resultan en la falta de disponibilidad del sistema SCADA.

En las Secciones 4.1, 4.2 y 4.3 se analiza cada una de las hipótesis, con la finalidad de demostrar o refutar su validez, de acuerdo con los datos recopilados en los experimentos de la Sección 3.1.

4.1. Hipótesis 1

Esta hipótesis plantea que la desconexión observada y la consecuente inhabilitación del *software* se deben a la pérdida de comunicación entre *switches*. Esta situación se atribuye a la configuración en anillo de fibra óptica que interconecta los dispositivos de la red *Operational Technologies* (OT).

En este contexto, las pruebas detalladas en la Sección 3.1.1 se realizaron para demostrar que, al desconectar algunos de los *switches* de la red, el equipo que captura el tráfico no debería recibir más datos de los dispositivos conectados al sistema SCADA. Estos dispositivos se mencionan en el listado de direcciones Internet Protocol (IP) del manual de programación del SCADA, disponible en [56].

Al momento de capturar el tráfico según lo planificado en el diseño del experimento, se observó que, a pesar de la desconexión, los datos de las direcciones IP de la red OT persisten. Esto es evidente en los resultados de las Figuras 3.7 y 3.10, comparables a la captura de paquetes previa a la desconexión, como se muestra en las Figuras 3.4, 3.5 y 3.6.

Validación:

Al observar los resultados del experimento, se rechaza la hipótesis inicial, ya que se verifica que, a pesar de interrumpir una parte del anillo de fibra, los datos del resto de equipos se siguen recibiendo. Esto se debe a la gestión de redundancia mediante conexiones *Ethernet* directas entre todos los *switches* y el *switch* principal.

Razón:

Los resultados obtenidos en el primer experimento revelan que la inoperabilidad del sistema no es causada por la desconexión de uno de los equipos en el anillo de fibra. En cambio, se observa que la gestión de redundancia mediante conexiones *Ethernet* asegura la continuidad del tráfico de datos. Esto implica que la causa de la inhabilitación del *software* debe buscarse en otros factores.

4.2. Hipótesis 2

La hipótesis plantea que la falta de disponibilidad del sistema SCADA se debe a la interrupción en la red física del anillo de fibra óptica. Entonces, para validar o invalidar esto, se toman los datos de la cantidad de mensajes *Modbus* de cada *Programmable Logic Controller* (PLC), que llegan al servidor *Graylog*.

La prueba consiste en desconectar un PLC de su fuente de energía. En este caso, se ha tomado como ejemplo el PLC de la *Application Programming Interface* (API) 3, dado que es la API más sencilla de desconectar, según el personal de la Micro-Red. En la Figura 4.2 se muestra en un diagrama de barras, para cada PLC de las diferentes *Application Programming Interfaces* (APIS), el conteo de *Queries* y *Responses* por minuto, antes de la desconexión del equipo.

Figura 4.2: Datos recolectados de los PLC, previo a la desconexión de un equipo

Ahora, en la Figura 4.3 se muestran los mismos diagramas de barras, correspondientes al conteo de mensajes *Modbus* por minuto, después de la desconexión del equipo. Entonces, se observar que, el conteo de *Queries* y *Responses* del PLC de la APIS 3, cambió a 0. Mientras que, el conteo de *Queries* y *Responses* de los demás PLC se mantuvo en su valor habitual.

Figura 4.3: Datos recolectados de los PLC, posterior a la desconexión de un equipo

En la Figura 4.4 se observa el conteo de *Queries* y *Responses* a lo largo del tiempo, para cada PLC. Entonces, en el instante en que se desconecta el PLC de la APIS 3, se observa que el punto de la recta en dicho instante se vuelve cero.

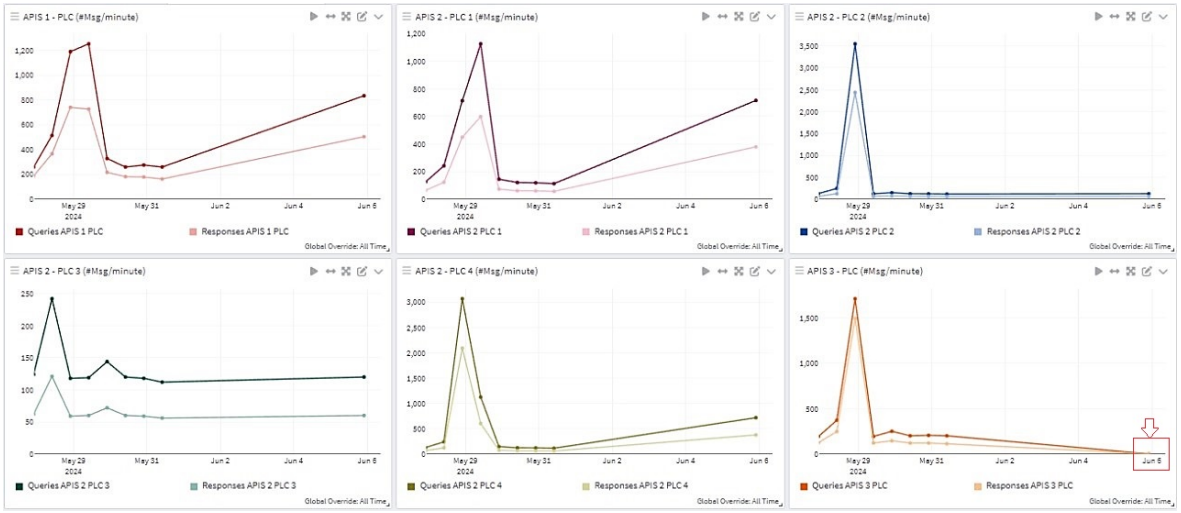
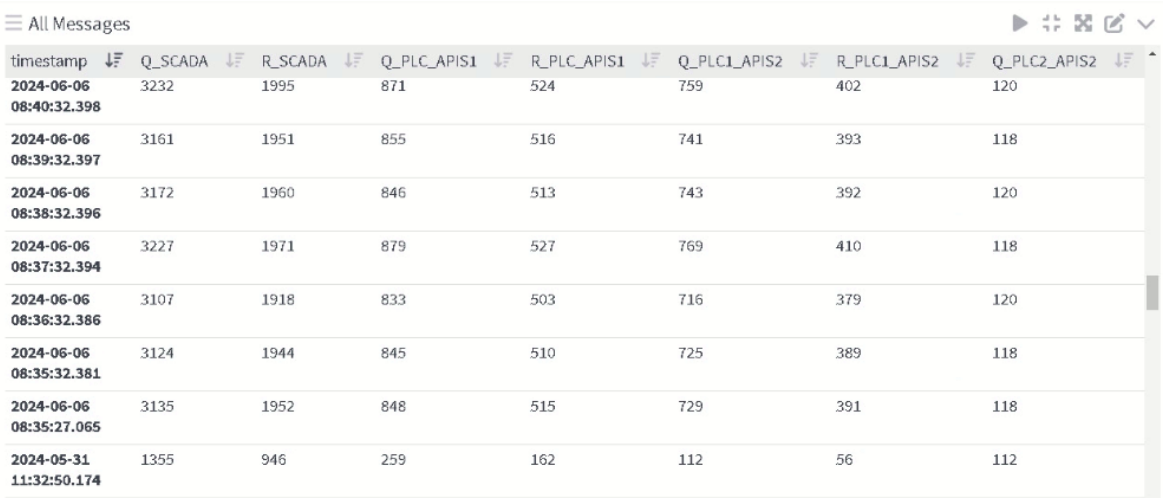


Figura 4.4: Conteo de *Queries* y *Responses* a lo largo del tiempo, posterior a la desconexión de un equipo

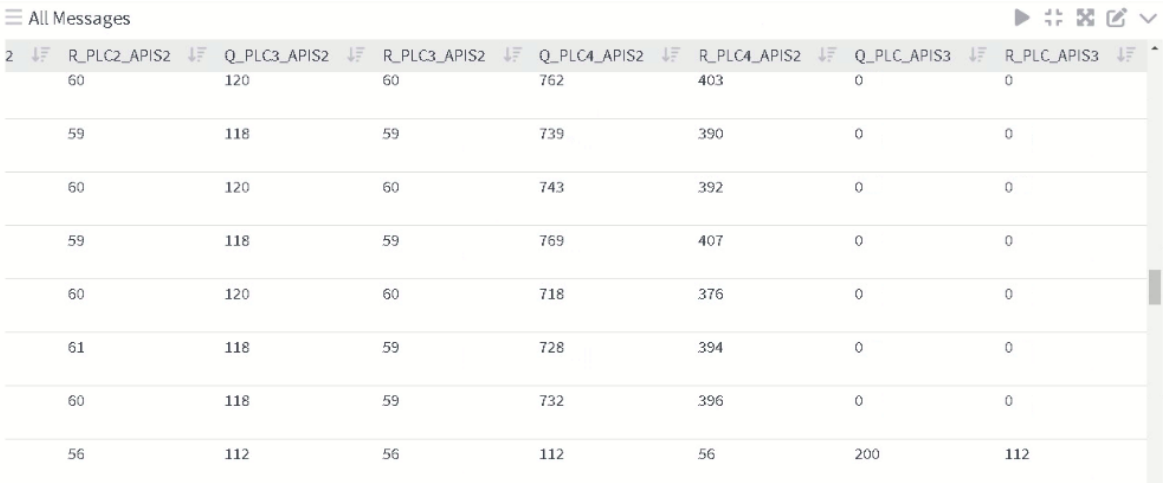
Por último, para obtener una visión más precisa del valor del conteo para cada PLC, en la Figura 4.5 se presenta una tabla proporcionada por *Graylog*, donde se guarda, por cada minuto, dicho valor para el SCADA, y el PLC de la APIS 1. En la Figura 4.6 se observa dicho valor para los demás PLC, incluyendo el que se desconectó. Del mismo modo, se observa que el valor para el PLC 3 se vuelve 0 al momento de la desconexión. Mientras que, el conteo de los demás PLC mantiene un valor diferente de cero.



All Messages

timestamp	Q_SCADA	R_SCADA	Q_PLC_APIS1	R_PLC_APIS1	Q_PLC1_APIS2	R_PLC1_APIS2	Q_PLC2_APIS2
2024-06-06 08:40:32.398	3232	1995	871	524	759	402	120
2024-06-06 08:39:32.397	3161	1951	855	516	741	393	118
2024-06-06 08:38:32.396	3172	1960	846	513	743	392	120
2024-06-06 08:37:32.394	3227	1971	879	527	769	410	118
2024-06-06 08:36:32.386	3107	1918	833	503	716	379	120
2024-06-06 08:35:32.381	3124	1944	845	510	725	389	118
2024-06-06 08:35:27.065	3135	1952	848	515	729	391	118
2024-05-31 11:32:50.174	1355	946	259	162	112	56	112

Figura 4.5: Valor del conteo de *Queries* y *Responses* para el SCADA y los PLC



All Messages

2	R_PLC2_APIS2	Q_PLC3_APIS2	R_PLC3_APIS2	Q_PLC4_APIS2	R_PLC4_APIS2	Q_PLC_APIS3	R_PLC_APIS3
	60	120	60	762	403	0	0
	59	118	59	739	390	0	0
	60	120	60	743	392	0	0
	59	118	59	769	407	0	0
	60	120	60	718	376	0	0
	61	118	59	728	394	0	0
	60	118	59	732	396	0	0
	56	112	56	112	56	200	112

Figura 4.6: Valor del conteo de *Queries* y *Responses* de cada PLC

Validación: La hipótesis correspondiente a la interrupción en la red física del anillo de fibra óptica no es válida.

Razón: La investigación reveló que la desconexión de un equipo PLC no repercute en los demás dispositivos dentro de la red de fibra óptica. Por ende, la suposición inicial sobre una posible interrupción en la red física de tipo anillo, en la que se basaba la hipótesis, no se confirma en la práctica.

4.3. Hipótesis 3

Al refutar las hipótesis planteadas en los experimentos realizados, se confirma que la interconexión física del sistema no se ve afectada por la desconexión de algún equipo. La red OT mantiene su operación normal gracias a las conexiones *Ethernet*, que restablecen el funcionamiento del anillo de fibra. En este contexto, surge la idea de que si no es un problema

físico de la red, el problema probablemente radica en la programación en LabVIEW, como se detalla brevemente en el Manual disponible en [56].

Tras revisar la documentación disponible y el programa de bloques del sistema, se realizan las pruebas pertinentes para analizar el comportamiento del *software* y validar esta nueva hipótesis. Al iniciar el sistema SCADA, se observa que el tiempo estimado de inicialización oscila entre las 6 y 8 horas, lo cual representa uno de los problemas principales identificados hasta este punto de la investigación.

Posteriormente, se han generado escenarios de prueba junto con los operarios del laboratorio, simulando procesos de mantenimiento que provocan errores a los que normalmente se enfrenta el sistema. A continuación, se detallan los errores identificados y las soluciones implementadas. Primero, se identificó que ciertos procesos de mantenimiento causaban la pérdida de comunicación temporal entre los PLC y el sistema SCADA, lo que derivaba en la inhabilitación temporal del sistema. El error específico despliega la ventana mostrada en la Figura 4.7.

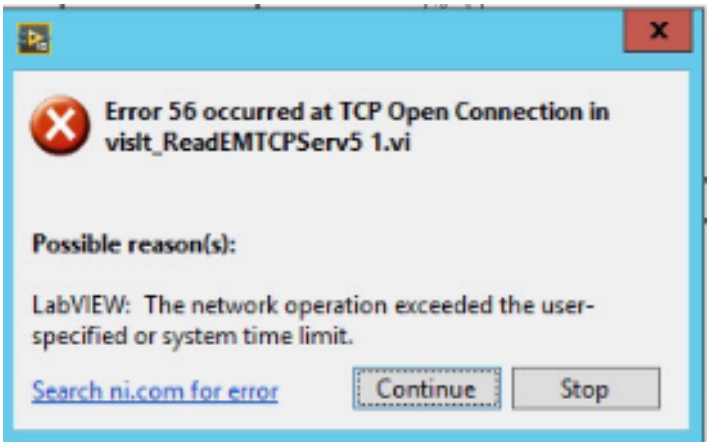


Figura 4.7: Error en la conexión Transmission Control Protocol (TCP)

Al explorar el código implementado en el archivo principal con extensión .vi, se observó que el error provenía de un *bucle* de configuración realizado en una *Case Structure*. Este *bucle* dependía de la variable *status*, la cual verifica que no se haya generado un error de comunicación para que el sistema funcione correctamente.

La solución a este problema implicó la modificación del código para que el error sólo remita una advertencia en el *subVI* la implementación de un protocolo de reconexión automática en LabVIEW, permitiendo al sistema reanudar la comunicación sin requerir la restauración de la conexión física. La parte del código que se ha modificado, se muestra en la Figura 4.8.

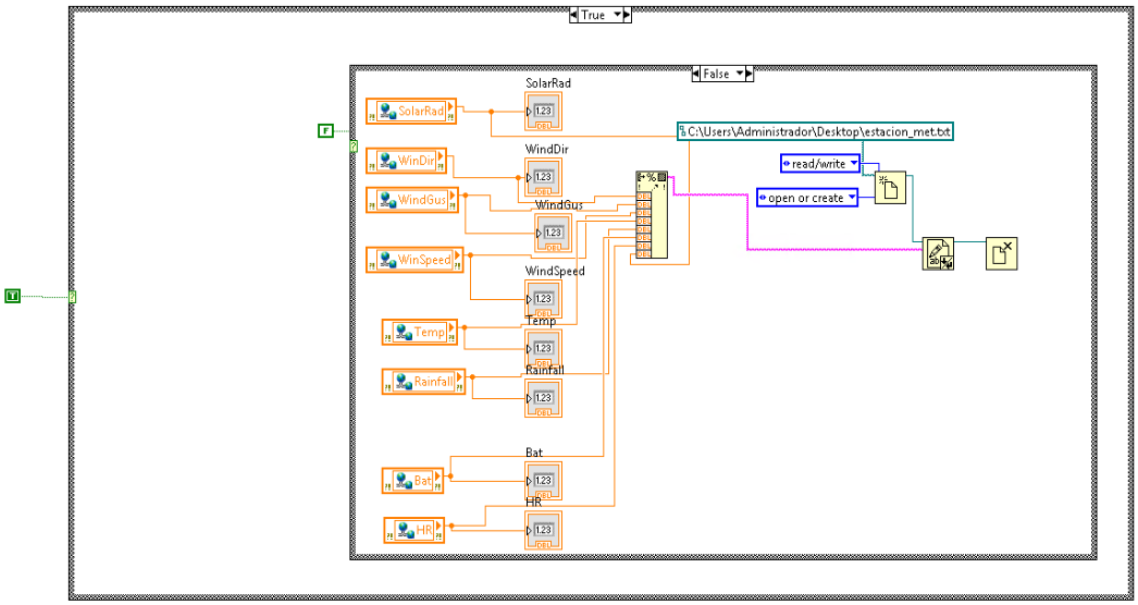


Figura 4.8: Modificación en el código para resolver el error en la conexión TCP

En la imagen adjunta (Figura 4.8), se observa el bloque de programación en LabVIEW que gestiona diversas variables, incluyendo SolarRad, WindDir, WindGus, WinSpeed, Temp, Rainfall, Bat y HR. Estas variables se procesan y se almacenan en un archivo de texto en el escritorio, lo que permite su monitorización y análisis. El error se relaciona con un límite de tiempo en una operación de red en LabVIEW. En este sentido indica que la ejecución del bucle esperando la confirmación de que no hay errores de conexión TCP excedió el límite de tiempo especificado por el sistema.

La exploración del código permitió identificar que el bucle fallaba cuando la variable *status* no se actualizaba correctamente, lo que causaba un error de comunicación y, consecuentemente, la inhabilitación del *software*. Se realizaron ajustes en el código para asegurar que la variable *status* se actualizara adecuadamente y se añadieron controles adicionales para manejar posibles errores de comunicación. La corrección consistió en inicializar una variable constante con el valor *false* para que siempre ingrese al *case* y realice el funcionamiento esperado. Una vez corregido el problema en el código, se procedió a realizar nuevas pruebas para asegurar la correcta operación del sistema.

Adicionalmente, se detectaron errores en la programación del SCADA que impedían la correcta inicialización de ciertos módulos de monitoreo. Estos errores obedecen a la inicialización de todos los datos históricos del sistema SCADA. Esto se visualiza en la Figura 4.9.

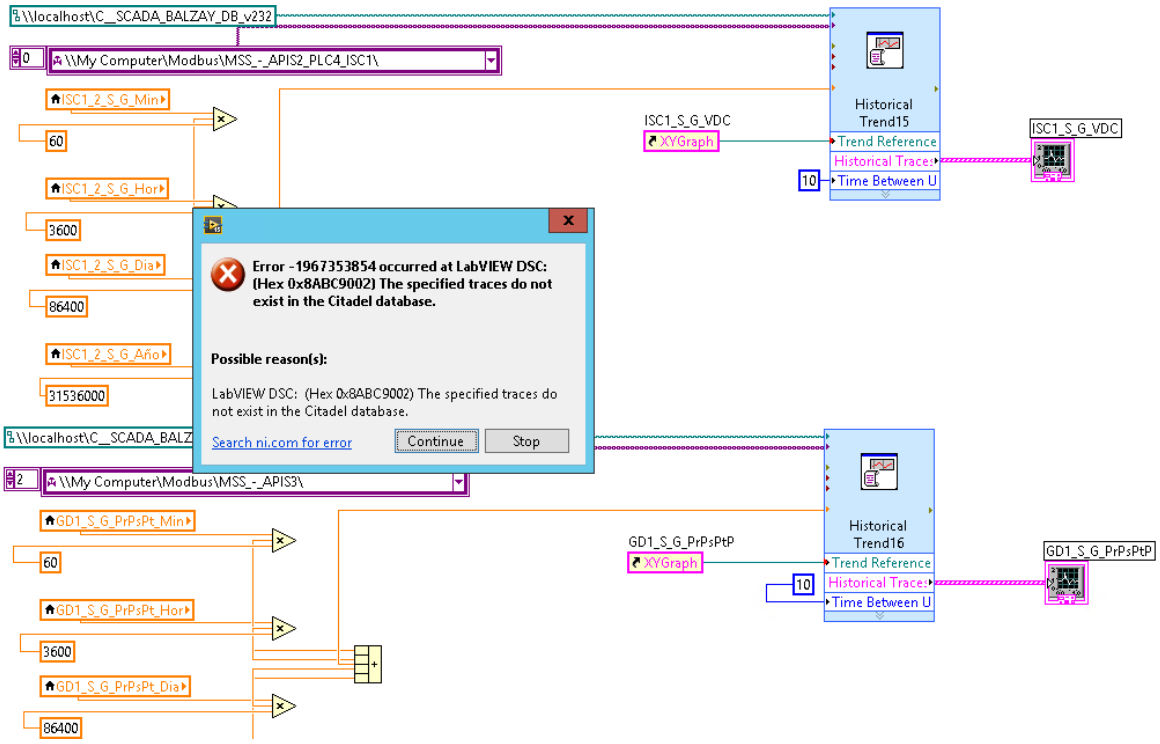


Figura 4.9: Error en la función *historical Trend*

El error observado se relaciona con LabVIEW Datalogging and Supervisory Control (DSC) y la base de datos Citadel. El código de error -1967353854 indica que las trazas especificadas no existen en la base de datos Citadel. La información proporcionada sugiere que se está intentando acceder a trazas históricas, posiblemente para visualización o análisis, pero las trazas no se encuentran en la base de datos especificada. Esto se ha verificado al realizar una depuración paso a paso del código implementado, para encontrar la razón del retardo existente al momento de inicializar este *bucle*.

Para solucionar este problema se ha constatado que la configuración de la base de datos Citadel sea correcta y que las trazas existen. No obstante al explorar esta base de datos a través de un gestor de bases de datos Structured Query Language (SQL), se verificó que el retardo mencionado obedece a que al momento de inicialización el sistema está cargando todos los datos alojados en el servidor de esta base de datos desde la creación del sistema SCADA.

Para solventar este error se procede con la eliminación de la función *Historical Trend*, pues los operarios del sistema utilizan otra herramienta de LabVIEW para visualizar los datos adquiridos por el sistema en tiempo real: NI Measurement & Automation Explorer (NI MAX). Sin embargo, se consideró oportuno solventar la carga de los datos para un rango de datos útiles considerados por los directivos del laboratorio de la Micro-red. Para ello, se utilizó la herramienta SQL Server Management Studio (SSMS). El proceso de eliminación de datos se explican en el video que forma parte de los entregables de este trabajo de titulación y cuyo enlace de adjunta debajo.

Video sobre manejo de la base de datos Citadel disponible en https://drive.google.com/file/d/19zf3hQ_ftT5G7Qe-q2UyIbQvHaqWLzff/view?usp=drive_link

Además, se observó que la versión preliminar del código implementado en LabVIEW para el funcionamiento del sistema SCADA incluía una etapa de inicio que solicitaba el ingreso de credenciales para inicializar el *sotfware*. Esta etapa, mostrada en la Figura 4.10, se considera innecesaria ya que los operarios del sistema no disponen de estas credenciales, por ello fue eliminada del código en la versión final de este trabajo para evitar procesos innecesarios.

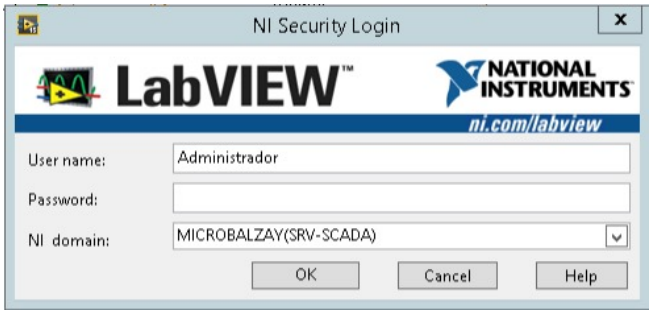


Figura 4.10: Proceso de inicio de sesión

Con las modificaciones del código realizadas, se simularon nuevamente procesos de mantenimiento provocados y se verificó que el sistema permanecía operativo y sin errores de comunicación, confirmando así que la solución implementada era efectiva. De esta manera, se logró identificar y resolver el problema de programación en LabVIEW, mejorando la disponibilidad del sistema SCADA del laboratorio de la Micro-red de la Universidad de Cuenca.

4.4. Síntesis de validación

Al concluir este capítulo, se ha completado la validación de cada una de las hipótesis planteadas mediante los métodos de investigación empleados. Las conclusiones derivadas de la ejecución de los experimentos se han condensado en la Tabla 4.1.

Tabla 4.1: Resumen de la Validación de Hipótesis

Hipótesis	**Estado**	**Inclusión en la Solución**
La carencia de gestión de la redundancia de red a través de los *switches Weidmüller* provoca la falta de disponibilidad del sistema SCADA	Rechazada	No
La desconexión de un equipo en la red física del anillo de fibra óptica es la causa principal de la falta de disponibilidad del sistema SCADA	Rechazada	No
La programación del sistema SCADA en el *software* LabVIEW resultan en la falta de disponibilidad del sistema SCADA.	Aceptada	Sí

5. Resultados

En este capítulo se presentan los resultados de este trabajo de titulación, los cuales se basan en el análisis y la ejecución de las soluciones propuestas conforme a la metodología empleada y las hipótesis aceptadas.

Una vez analizadas las distintas hipótesis planteadas inicialmente y comprobado que la falta de disponibilidad del sistema *Supervisory Control and Data Acquisition* (SCADA) se debe a problemas en la programación del sistema SCADA en el *software* Laboratory Virtual Instrument Engineering Workbech (LabVIEW), se realizaron pruebas de desconexión de equipos con ambas versiones del programa: la anterior y la nueva. En la versión nueva del programa, se han depurado los errores provenientes de la desconexión de cualquier *Programmable Logic Controller* (PLC).

A continuación, se explica como se hizo la desconexión del PLC de la *Application Programming Interfaces* (APIS) 3 en la Sección 5.1. Además, se presentan y analizan los resultados de la desconexión de la fuente de energía del PLC de la APIS 3, con ambas versiones, la antigua en la Sección 5.2 y la nueva en la Sección 5.3.

5.1. Proceso de desconexión del PLC de la APIS 3

Para desconectar la fuente de energía del PLC de la APIS 3, se ubica físicamente la APIS en el laboratorio de la Micro-Red de la Universidad de Cuenca. Entonces, en la Figura 5.1 se observa la APIS mencionada.

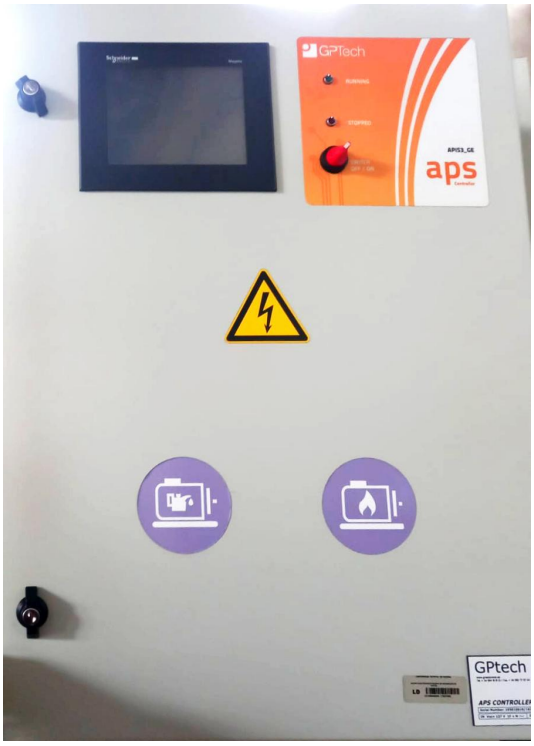


Figura 5.1: APIS 3

Posteriormente, se abre la compuerta de la APIS 3 para ubicar el PLC. En la Figura 5.2 se observa que dicho PLC se encuentra ubicado en la parte superior izquierda. Una vez aquí, se desconecta la alimentación respectiva, cuyos cables se encuentran en la parte inferior del PLC, con una etiqueta de +24V.

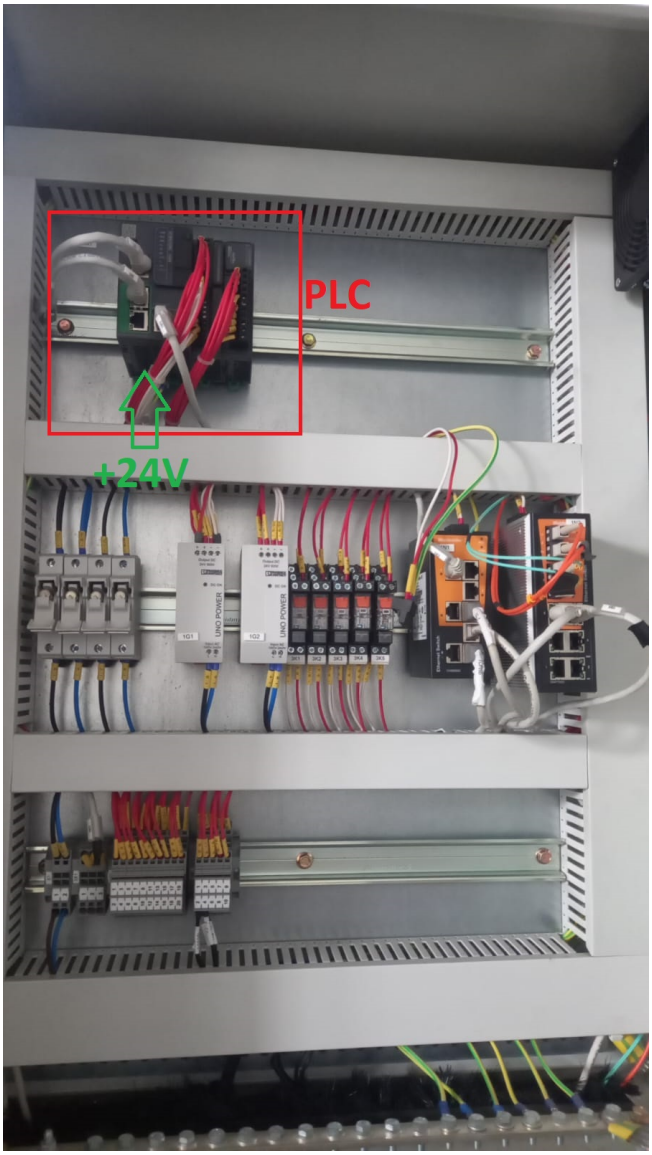


Figura 5.2: Ubicación del PLC y su fuente de alimentación en la APIS 3

Para confirmar que el PLC se ha desconectado de manera correcta, se revisa el *Dashboard* de Mensajes *Modbus* en *Graylog*. En este panel, se observa que, el número de *Queries* y *Responses* del PLC de la APIS 3 se encuentra en 0, lo cual indica que el equipo no está enviando ni recibiendo mensajes. Esta situación se ilustra en la Figura 5.3, donde se muestran los datos de comunicación del PLC después de la desconexión.

Figura 5.3: Número de *Queries* y *Responses* de los PLCs al momento de desconectar el PLC de la APIS 3

5.2. Resultados para la versión antigua del programa del sistema SCADA

Después de realizar la desconexión, en la sección de Comunicaciones del sistema SCADA en LabVIEW, se genera una alerta indicando un error de comunicación con la APIS 3. Esta alerta se presenta en un cuadro, con una alerta visual en ERROR DE COM, informando sobre la pérdida de conectividad con dicho equipo. Esta situación es ilustrada en la Figura 5.4, donde se observa la correspondiente alerta de error.

Figura 5.4: Alerta visual del error de comunicación con el PLC de la APIS 3

Luego de unos segundos, se muestra el cuadro de texto de la Figura 5.5, que indica el siguiente error: Error -1967353902 occurred at Shared Variable in HMI Centro_de_Control.vi. El código de error -1967353902 que surge en el archivo HMI Centro_de_Control.vi de LabVIEW señala que el servidor Input/Output (I/O) de *Modbus* no ha recibido respuesta alguna del dispositivo esclavo *Modbus*.

Este problema se origina por un fallo en el funcionamiento del dispositivo esclavo o por una configuración inadecuada de la conexión entre los dispositivos maestro y esclavo *Modbus*. Además, la aparición de este error está asociada con el intento de leer una variable

compartida específica en el sistema. Por lo tanto, se origina a partir de la desconexión del PLC de la APIS 3.

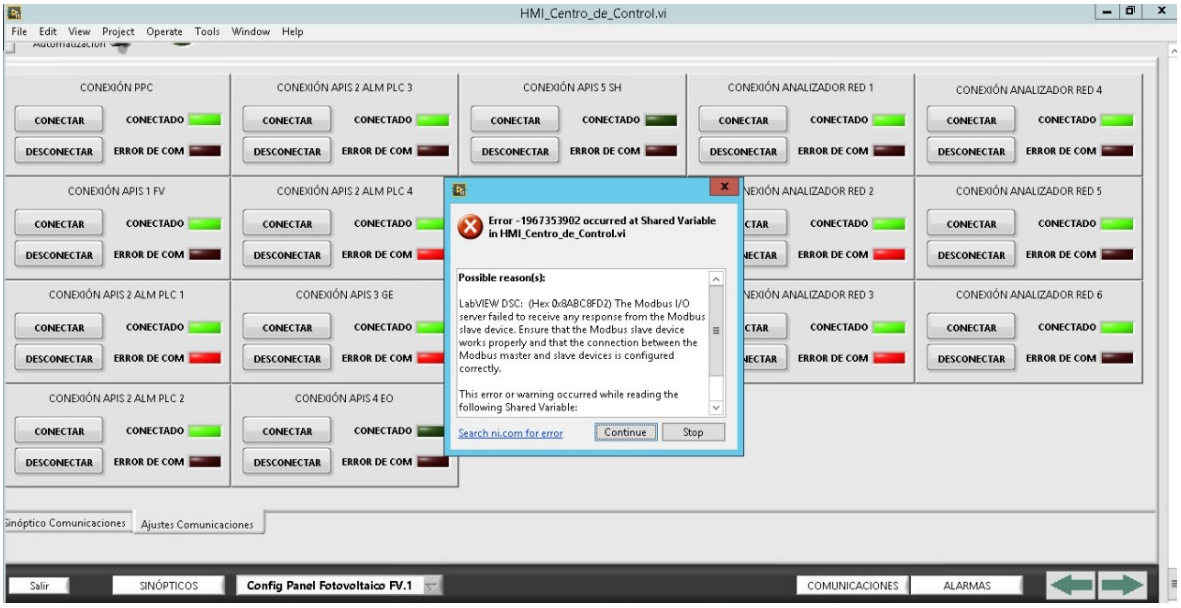


Figura 5.5: Error originado a partir de la desconexión del PLC de la APIS 3

Este error persiste mientras el PLC de la APIS 3 se encuentra desconectado, incluso si se cierra dicho cuadro de texto o se eligen las opciones de *Continue* o *Stop*, lo que impide la navegación e interacción con la interfaz del sistema SCADA. Esta situación genera una interrupción significativa en la operación del sistema, afectando la capacidad de monitoreo y control de los procesos asociados a los equipos conectados al sistema SCADA.

Al reconectar el PLC de la APIS 3 y confirmar que este comienza nuevamente a enviar *Queries* y recibir *Responses*, como se muestra en la Figura 5.6 (*Dashboard* en *Graylog*), se observa que el sistema SCADA en LabVIEW no se recupera automáticamente. A pesar de que la comunicación entre el PLC y el servidor se restablece, el sistema SCADA permanece en estado de error, indicando que requiere intervención manual para reanudar su funcionamiento normal.

Figura 5.6: Datos recibidos luego de conectar el PLC de la APIS 3

Para restaurar el funcionamiento normal del sistema SCADA, es necesario cerrar el programa, volver a abrirlo e iniciarlo nuevamente. El tiempo requerido para este proceso de inicialización, representado en la Figura 5.7, varía considerablemente. Dependiendo de la cantidad de datos que el programa debe procesar, este procedimiento dura desde pocos minutos hasta 8 horas (con los datos desde el año 2018). La variabilidad en el tiempo de reinicio subraya la importancia de una correcta gestión de datos y la eficiencia del sistema en situaciones de recuperación.

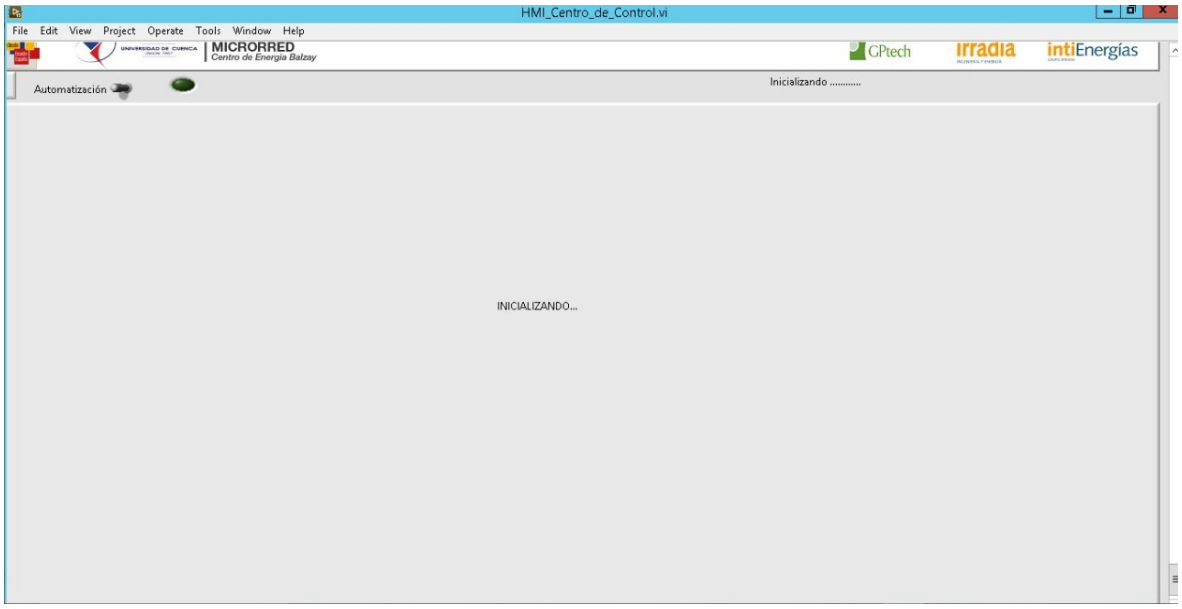


Figura 5.7: Proceso de inicialización del sistema SCADA en LabVIEW

5.3. Resultados para la versión nueva del programa del sistema SCADA

A continuación, se presentan los resultados obtenidos al implementar la nueva versión del programa del sistema SCADA en LabVIEW. Es importante destacar que esta versión constituye la solución desarrollada en el presente trabajo de titulación, basada en la Hipótesis 3: Problemas en la programación del sistema SCADA en el software LabVIEW. Esta actualización del programa ha sido diseñada y probada para resolver las deficiencias identificadas, y se explicarán los efectos y mejoras logrados con esta implementación.

Respecto a la situación después de la desconexión del PLC de la APIS 3, en la Figura 5.8 se evidencia que, en la sección de Comunicaciones, aparece una alarma visual indicando un error de comunicación con dicho equipo. No obstante, a diferencia del programa anterior, no se muestra el error en el cuadro de texto que se ve en la Figura 5.5. Esto significa que la navegación e interacción con la interfaz gráfica del programa no se ven afectadas, permitiendo a los técnicos del laboratorio utilizarla en cualquier momento, incluso durante casos de desconexión.

Figura 5.8: Alerta visual del error de comunicación con el PLC de la APIS 3

Del mismo modo, en la Figura 5.9 se observa un diagrama del anillo de fibra óptica que interconecta los equipos de la Micro-red en el sistema SCADA. Durante la desconexión del PLC de la APIS 3, se genera una alerta visual específica para este equipo, pero el resto del sistema permanece operativo. Esto significa que, a pesar de la desconexión, el funcionamiento global del programa sigue con normalidad, permitiendo que el resto de los dispositivos continúen comunicándose y operando sin interrupciones. Esta resiliencia asegura que el sistema SCADA mantenga su funcionalidad integral, minimizando cualquier impacto negativo en el control y monitoreo de la Micro-red.

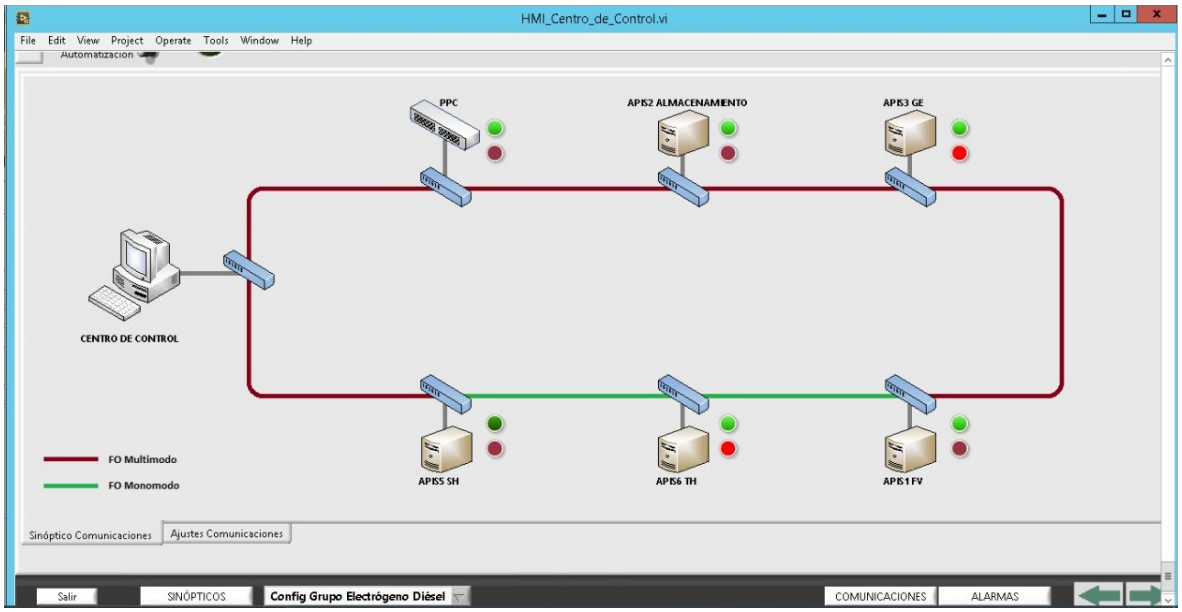


Figura 5.9: Diagrama del anillo de fibra óptica que interconecta los equipos de la Micro-red en el sistema SCADA

Del mismo modo, después de reconectar el equipo, no es necesario reiniciar y volver a iniciar el programa. La falta de comunicación temporal con el PLC no afecta el funcionamiento habitual del sistema SCADA. Este diseño garantiza que el sistema continúe operando sin interrupciones, incluso después de reconectar el dispositivo previamente desconectado.

Además, en el *Dashboard* de *Graylog*, como se muestra en la Figura 5.10, se visualiza que, tras la reconexión del PLC de la APIS 3, este retoma sus valores habituales de *Queries* y *Responses*. Este restablecimiento confirma que la comunicación entre el PLC y el sistema SCADA se ha reanudado correctamente y que el equipo vuelve a integrarse a la red sin necesidad de acciones adicionales.

Figura 5.10: Conteo de *Queries* y *Responses* de los PLCs luego de reconectar el PLC de la APIS 3

Por último, es importante destacar que en esta nueva versión del programa, el tiempo requerido para iniciar el sistema SCADA se ha reducido significativamente a solo unos pocos minutos, en marcado contraste con las 6-8 horas que tomaba con la versión anterior del

software. Este cambio representa una mejora notable en la disponibilidad del sistema, lo que beneficia considerablemente la operatividad y la productividad general del entorno de monitoreo y control.

6. Conclusiones y recomendaciones

6.1. Conclusiones

Al concluir el desarrollo del presente trabajo de titulación, se extraen conclusiones importantes que ratifican el cumplimiento de cada uno de los objetivos planteados. En primer lugar, se identificó que la redundancia en la red *Operational Technologies* (OT) se encontraba correctamente gestionada en la configuración de los *switches Weidmüller*. De esta manera, la identificación y evaluación de las características configurables de los equipos, en este caso de los *switches*, garantizó la continuidad de la comunicación y el control del sistema. Este paso resultó primordial para mantener la operatividad del sistema, incluso ante desconexiones de equipos durante el mantenimiento, descartando así posibles problemas de infraestructura.

Además, la combinación de métodos de investigación mixta, como el estudio de caso, el método empírico y el diseño de experimentos, permitió identificar las causas subyacentes de los problemas en el sistema *Supervisory Control and Data Acquisition* (SCADA) de la Micro-Red. Durante el transcurso de la investigación, se incluyeron problemas de falta de disponibilidad del sistema en momentos críticos, errores de comunicación recurrentes con los dispositivos *Programmable Logic Controller* (PLC) y tiempos prolongados de inicio del sistema. Para abordar estas problemáticas, se formularon hipótesis iniciales y se diseñaron experimentos específicos.

Consecuentemente, el proceso de validación se llevó a cabo mediante pruebas de desconexión de equipos y monitoreo continuo de las comunicaciones entre el sistema SCADA y los dispositivos PLC. Los resultados confirmaron que los fallos en la operatividad del sistema SCADA estaban relacionados con la programación del software Laboratory Virtual Instrument Engineering Workbech (LabVIEW), y no con la infraestructura de red física. Por tanto, la implementación de soluciones específicas, como la actualización del código y la reconfiguración de la base de datos Citadel, tuvo efectos significativos en la funcionalidad y el rendimiento del sistema. Con ello, se observó una reducción drástica en los tiempos de inicio del sistema, así como una mejora general en la estabilidad y la disponibilidad del sistema, especialmente en procesos donde se requiere la desconexión de PLCs.

Finalmente, se identificaron los principales factores que afectan la disponibilidad del sistema SCADA, que conlleva principalmente problemas en la programación del sistema en el *software* LabVIEW. Al abordar estos problemas, se demostró que es posible mejorar significativamente la disponibilidad del sistema SCADA. Las mejoras incluyen la depuración de mensajes de error al momento de desconectar un equipo, para que no se congele el programa; el seguimiento de la desconexión de estos equipos, a través de Graylog; y la reducción del tiempo de inicialización del sistema SCADA en el software LabVIEW.

Estos cambios permiten la continuidad operativa durante los procesos que conlleven la desconexión de equipos, lo que asegura una operatividad continua del sistema SCADA.

Así, se concluye que la mejora de la disponibilidad del sistema SCADA es factible y puede lograrse mediante la mejora de la programación en el software LabVIEW.

6.2. Recomendaciones

Se recomienda llevar a cabo pruebas y validaciones continuas a lo largo del ciclo de desarrollo del sistema SCADA. Estas pruebas deben abarcar desde el funcionamiento básico hasta el rendimiento y la integración con dispositivos y sistemas externos. Al realizar pruebas de manera regular y sistemática, es posible identificar y corregir problemas en etapas tempranas del desarrollo, lo que contribuye a la calidad y confiabilidad del sistema final.

Además, la implementación de sistemas de monitoreo continuo y análisis de datos contribuyen a la detección proactiva de problemas en el sistema SCADA. Estos sistemas permiten identificar patrones de comportamiento anómalos y tomar medidas correctivas antes de que los problemas afecten la operatividad del sistema. Al monitorear y analizar constantemente los datos del sistema, se evitan interrupciones no planificadas y maximizar la disponibilidad y eficiencia del sistema SCADA.

Asimismo, es importante establecer un programa de mantenimiento preventivo para garantizar el funcionamiento óptimo y la disponibilidad continua del sistema SCADA. Este programa incluye actividades como inspecciones regulares de equipos, actualizaciones de software y optimización de la configuración de red. Además, es importante que se actualice el sistema operativo de alojamiento del programa del sistema SCADA en LabVIEW, que actualmente es *Windows Server 2012 R2*, dado que su soporte principal finalizó en 2018, y su soporte extendido en 2023.

Por lo tanto, el sistema operativo ya no garantiza que se tenga seguridad, compatibilidad, acceso a soporte técnico y mejoras con las actualizaciones. Por otro lado, proporcionar capacitación adecuada al personal encargado del mantenimiento y la operación del sistema SCADA para garantizar su funcionamiento adecuado. El personal debe estar familiarizado con las herramientas y tecnologías utilizadas, así como con los procedimientos de operación y respuesta ante situaciones de emergencia. La capacitación contribuiría a mejorar la eficiencia operativa y la seguridad del sistema SCADA.

Finalmente, se recomienda mantener una documentación completa y actualizada del sistema SCADA para su operación y mantenimiento a largo plazo. Esta documentación incluye manuales de usuario, diagramas de red, configuraciones de equipos y procedimientos operativos y de mantenimiento. Una documentación completa facilita la resolución de problemas, la capacitación del personal y la transferencia de conocimientos, asegurando la continuidad operativa del sistema SCADA.

6.3. Trabajos futuros

A futuro hay que considerar la actualización del *hardware* de la Micro-red como una parte integral de la optimización del sistema. Esto incluye, por ejemplo, RAM y almacenamiento rápido como SSDs NVMe. La actualización del *hardware* de visualización y operación del sistema resultará en una mayor velocidad de procesamiento y menor latencia en las operaciones críticas del sistema.

Otro aspecto a considerar es la implementación de un sistema de respaldo automatizado para los datos de la base de datos del sistema SCADA. Este enfoque permitiría la creación regular de copias de seguridad de los datos almacenados, reduciendo la carga sobre el sistema principal y evitando posibles problemas de rendimiento debido a la acumulación de datos. Al tener copias de seguridad disponibles, el sistema SCADA tiene acceso a datos históricos cuando sea necesario, sin comprometer su velocidad y eficiencia operativa del sistema SCADA en general.

Adicionalmente, un trabajo futuro se motiva en el uso de un software externo, como *Modbus Poll*, para analizar detalladamente el intercambio de mensajes *Modbus*. Esta herramienta permite simular un maestro *Modbus*, facilitando la observación en tiempo real de las tramas de consulta y respuesta. Al proporcionar una vista detallada de los datos intercambiados, *Modbus Poll* diagnostica y depurar errores de comunicación. De esta forma, se optimizaría la implementación del protocolo y la eficiencia del sistema SCADA.

Por otro lado, es necesario mejorar el registro efectivo de los procesos de mantenimiento llevados a cabo en el laboratorio de la Micro-red. Por ello, un tema interesante de estudio considera la configuración de cada PLC en el registro, ya que estos podrían utilizarse para gestionar de manera eficiente la red OT.

Finalmente, otro trabajo importante es la integración de tecnologías emergentes, como la inteligencia artificial y el Internet of Things (IoT), para potenciar las capacidades del sistema SCADA. Estas tecnologías generan oportunidades para la automatización inteligente, el análisis avanzado de datos y la toma de decisiones predictivas, lo que genera una mejora considerable en el sistema.

Referencias

[1] E. Wood, “What is a microgrid?” Microgrid Knowledge, March 2023. [En línea]. Disponible: https://www.microgridknowledge.com/about-microgrids/article/11429017/what-is-a-microgrid

[2] “Qué es un sistema scada? información completa - autexopen,” https://www.autex-open.com/automatizacion-industrial/que-es-un-sistema-scada-informacion-completa/, consultado el 25 de octubre de 2023.

[3] D. L. De y et al., “”modelo de estado estacionario de la microrred cuenca-ecuador 2017 trabajo de titulación previo a la obtención del título de ingeniero eléctrico”,” 2017.

[4] D. Ismael, M. Chuva, D. Efrain, B. Moncayo, M. Ángel, y Z. Prieto, “Cuenca-ecuador,” 2021.

[5] “LABORATORIO DE MICRORRED,” https://www2.ucuenca.edu.ec/ingenieria/laboratorios/lab-microrred.

[6] D. Ismael, M. Chuva, D. Efrain, B. Moncayo, M. Ángel, y Z. Prieto, “Creación de sistemas scada para el laboratorio de micro-red de la universidad de cuenca bajo el enfoque de desarrollo dirigido por modelos,” http://dspace.ucuenca.edu.ec/handle/123456789/37333, 2021, consultado el 14 de octubre de 2023.

[7] A. Ujvarosi, “Evolution of scada systems,” *Bulletin of the Transilvania University of Braşov*, vol. 9, num. 58, 2016.

[8] “Equipamiento universidad de cuenca - eductrade,” https://www.eductrade.com/ecuador-en-ejecucion/, consultado el 26 de octubre de 2023.

[9] Weidmüller, *Industrial Ethernet managed Switches Manual for Weidmüller managed switches of series ValueLine and PremiumLine*, https://manualzz.com/doc/6766188/manual-managed-weidm%C3%BCller-switches?__cf_chl_rt_tk=AwBNH07eK8by6v2JzAuM27a1NwF2gDaLuEHm1oBSUk8-1697373870-0-gaNycGzNDdA, Germany, Jun. 2014, consultado el 14 de octubre de 2023.

[10] A. S. Glista, “Una topología de red de fibra óptica en anillo derivado que proporciona detección, aislamiento y elusión de fallas.”

[11] R. Sankar y Y. Y. Yang, “An automatic failure isolation and reconfiguration methodology for fiber distributed data interface (fddi),” 1992.

[12] J. Zhou, D. Liu, X. Ma, y C. Ye, “Application of industry ethernet and configuration software in heating network monitoring system,” in *2009 WRI World Congress on Computer Science and Information Engineering, CSIE 2009*, 2009, pp. 131–134.

[13] E. G. Loayza, “Desarrollo de una guÍa prÁctica para la mediciÓn del trÁfico de red ip y monitoreo de dispositivos en tiempo real mediante herramientas mrtg y prtg,” http://repositorio.puce.edu.ec/bitstream/handle/22000/3421/T-PUCE-3575.pdf?sequence=1&isAllowed=y, 2010, consultado el 14 de octubre de 2023.

[14] □. Culebras, “Desarrollo de un sistema de monitorizaciÓn de redes scada para la detecciÓn de trÁfico anÓmalo,” 2016.

[15] Energy Pool, “What is a microgrid? benefits, types, and applications,” https://www.energy-pool.eu/en/what-is-a-microgrid/.

[16] D. Meléndez, *SISTEMA OPERATIVO LINUX*, Fundación Colegio Aplicación, Toico Palo Gordo, Municipio Cárdenas, 2024, cátedra: Informática, Segundo Año, Secciones: A y B. [En línea]. Disponible: https://uecaplicacion.weebly.com/uploads/6/0/4/0/60407417/guia_dos_2do_informatica.pdf

[17] Ubuntu Fácil, “Ubuntu server,” 04 2013. [En línea]. Disponible: http://www.ubuntufacil.com/2013/04/ubuntu-server/

[18] C. Ltd., “Ubuntu releases,” 2022. [En línea]. Disponible: https://ubuntu.com/about/release-cycle

[19] Microsoft, “Windows server 2012 r2,” 2024. [En línea]. Disponible: https://www.microsoft.com/es-es/evalcenter/evaluate-windows-server-2012-r2

[20] Precitool, “¿qué es windows server? características y ventajas,” 2024, av. 25 Oriente 1823 Col. Bella Vista. Puebla, Pue. México. [En línea]. Disponible: https://www.precitool.com/windows-server/

[21] Microsoft, “Windows server 2012 r2 lifecycle,” 2023. [En línea]. Disponible: https://learn.microsoft.com/en-us/lifecycle/products/windows-server-2012-r2

[22] *VMware Workstation 16.2.3 Pro Release Notes*, VMware Workstation Pro, 2023. [En línea]. Disponible: https://docs.vmware.com/en/VMware-Workstation-Pro/16.2.3/rn/vmware-workstation-1623-pro-release-notes.pdf

[23] M. S. Maldonado, “Guía completa de vmware workstation,” Scribd, 2011. [En línea]. Disponible: https://es.scribd.com/document/79514947/Guia-Completa-de-VMware-Workstation

[24] “Edición de la comunidad de labview - gratis para uso no comercial - ni.” [En línea]. Disponible: https://www.ni.com/es/shop/labview/select-edition/labview-community-edition.html

[25] “What is graylog.” [En línea]. Disponible: https://go2docs.graylog.org/current/what_is_graylog/what_is_graylog.htm

[26] “Introduction — scapy 2.6.0 documentation.” [En línea]. Disponible: https://scapy.readthedocs.io/en/latest/introduction.html#about-scapy

[27] "What is ni measurement & automation explorer (ni max) - ni." [En línea]. Disponible: https://www.ni.com/en/support/documentation/supplemental/21/what-is-ni-measurement---automation-explorer--ni-max--.html

[28] Check Point, "What is operational technology (ot) security?" https://www.checkpoint.com/cyber-hub/network-security/what-is-operational-technology-ot-security/, Sin fecha.

[29] Red Hat, "What is operational technology (ot)?" https://www.redhat.com/en/topics/edge-computing/what-is-ot, August 2022.

[30] A. Daneels y W. Salter, "What is scada?" in *International Conference on Accelerator and Large Experimental Physics Control Systems*, Trieste, Italy, 1999. [En línea]. Disponible: https://cds.cern.ch/record/532624/files/mc1i01.pdf

[31] A. Sami, "Scada (supervisory control and data acquisition)," *Technology Times*, 2019. [En línea]. Disponible: https://www.technologytimes.pk/wp-content/uploads/wp-advanced-pdf/1/scada-supervisory-data-acquisition.pdf

[32] Hewlett Packard Enterprise, "¿qué son los servidores para rack?" https://www.hpe.com/lamerica/es/what-is/rack-servers.html.

[33] Digital Guide IONOS, "¿cómo funcionan los servidores rack?" https://www.ionos.com/es-us/digitalguide/servidores/know-how/rack-server/, 01 2023.

[34] Red Hat, "What are application programming interfaces?" 2024, accessed: 2024-06-08. [En línea]. Disponible: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

[35] "Programación en plc: controladores programables – electrónica edimar." [En línea]. Disponible: https://www.edimar.com/programacion-en-plc/

[36] "Modbus, protocolo de comunicación industrial | wago es." [En línea]. Disponible: https://www.wago.com/es/modbus

[37] Modbusorg, "Modbus messaging on tcp/ip implementation guide v1.0b modbus organization," 2006. [En línea]. Disponible: http://www.Modbus.org

[38] Anonymous, "Modbus protocol and scada systems," 2023, accessed: 2023-07-23. [En línea]. Disponible: https://modbus.org/specs.php

[39] "Managed switches." [En línea]. Disponible: https://www.weidmueller.com/int/products/automation_software/industrial_ethernet/media_converter_protocol_gateways/index.jsp

[40] J. Torres y C. Vásquez, "Design of a main ring and redundante of optical fibre using technology 10gpon to optimise the traffic of the network in the north technical university."

[41] "Et02375 v4 especificación técnica," https://www.furukawalatam.com/es/versao-et-pdf/divisor-optico-modular, 2021, consultado el 26 de octubre de 2023.

[42] "Topología de red: qué es y cuáles son los tipos más habituales | unir ecuador." [En línea]. Disponible: https://ecuador.unir.net/actualidad-unir/topologia-red/

[43] J. Caldera y W. Suazo, "Módulo ii: Redes de datos." https://core.ac.uk/download/pdf/250142648.pdf, consultado el 26 de octubre de 2023.

[44] Black Box, "Ventajas de la topología de anillo en networking," https://www.blackbox.com.mx/mx-mx/page/41987/Recursos/Technical/black-box-explica/LAN/Ring-Topologies-in-Networking#:~:text=Las%20topolog%C3%ADas%20en%20anillo%20conectan,regresan%20al%20centro%20de%20operaciones, consultado el 7 de febrero de 2024.

[45] "Spanning tree procolol," http://www.redtauros.com/Clases/Redes_II/07_Spanning_Tree.pdf, consultado el 26 de octubre de 2023.

[46] A. Telesis y I. All rights reserved, "Stp feature overview and configuration guide."

[47] "The right redundancy technology for your application | moxa," https://www.moxa.com/en/spotlight/industrial-ethernet/redundancy-technology/technologies, consultado el 26 de octubre de 2023.

[48] M. Cuaresma y Q. Automation, "Turbo ring and turbo chain," *TECH CORNER*, May 2014, consultado el 26 de octubre de 2023.

[49] J. L. C. Honores y J. Q. Llanto, "El uso del enfoque del estudio de caso: Una revisión de la literatura," *Horizontes. Revista de Investigación en Ciencias de la Educación*, vol. 5, num. 19, pp. 775–786, 2021. [En línea]. Disponible: http://www.scielo.org.bo/pdf/hrce/v5n19/2616-7964-hrce-5-19-775.pdf

[50] S. Chetty, "The case study method for research in small- and medium-sized firms," *International Small Business Journal*, vol. 5, October–December 1996.

[51] E. GestioPolis, *Métodos y técnicas de investigación*. GestioPolis, sin fecha. [En línea]. Disponible: https://acortar.link/R1DNyK

[52] P. M. T. Rainusso, "Metodología para la aplicación del diseño de experimentos (doe) en la industria," Ph.D. dissertation, Universidad de Navarra, San Sebastián, May 2008.

[53] Weidmüller, "Industrial ethernet managed switches manual for weidmüller managed switches of series valueline and premiumline," Jun. 2014. [En línea]. Disponible: https://manualzz.com/doc/6766188/manual-managed-weidm%C3%BCller-switches?__cf_chl_rt_tk=AwBNH07eK8by6v2JzAuM27a1NwF2gDaLuEHm1oBSUk8-1697373870-0-gaNycGzNDdA

[54] B. A. Guachichullca Guamán, "Diseño e implementación de una Arquitectura de Ciberseguridad para la Micro-red de la Universidad de Cuenca." Feb. 2024.

[55] L. Gutiérrez y C. Padilla, “Manual de usuario ttsdn graylog server,” Jun. 2024.

[56] GPTech, “Scada procedimiento programación.”

[57] “Instalación y configuración básica de ubuntu server - jnoptik intrabach.” [En línea]. Disponible: http://intrabach.org/instalacion-y-configuracion-basica-de-ubuntu-server/

Anexos

Anexo A: Guía para la instalación y configuración del servidor Ubuntu 22.04 LTS

La instalación de *Ubuntu* se realizó dentro del servidor físico principal, cuya Internet Protocol (IP) es 192.X.X.X. Para crear esta máquina virtual, se utilizó el visualizador *VMware Workstation Pro* y se estableció una conexión remota con el servidor. En seguida, se procedió con la instalación de una máquina virtual. Los requisitos mínimos a considerar se enumeran a continuación [57]:

- 30 GigaBytes (GB) de espacio libre en disco.
- 6 GB de memoria Random Access Memory (RAM).

Los pasos a seguir para la instalación del servidor, se detallan a continuación:

1. Descargar la imagen International Organization for Standardization (ISO) de la versión de *Ubuntu Server* que se desea instalar, en este caso *Ubuntu 22.04 Long Term Support (LTS)*.
2. En *VMware* se selecciona la opción Nueva máquina y establecemos las características principales, a la vez que cargamos la imagen *ISO* para su instalación. En la Figura 1 de observan las opciones para realizarla instalación.

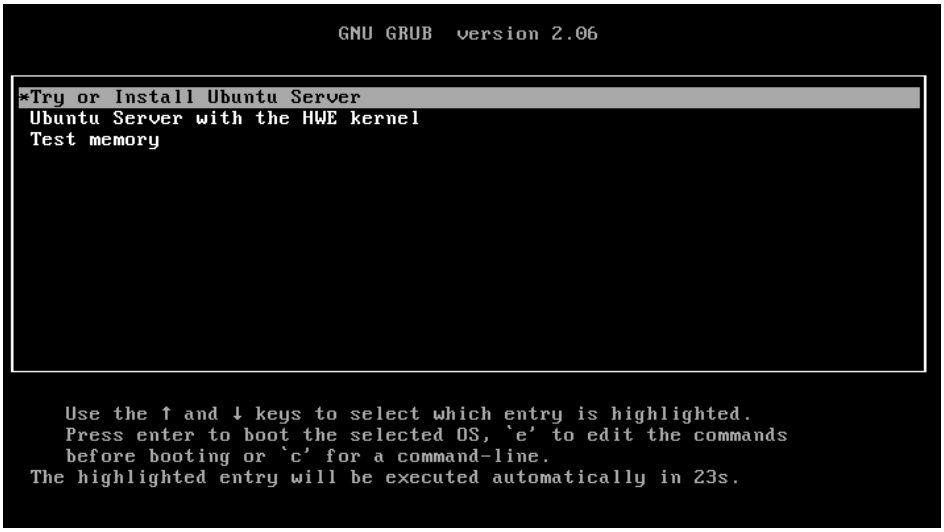


Figura 1: Instalación del sistema operativo *Ubuntu 22.04 LTS*

3. Configuración del perfil de usuario de ingreso al servidor.

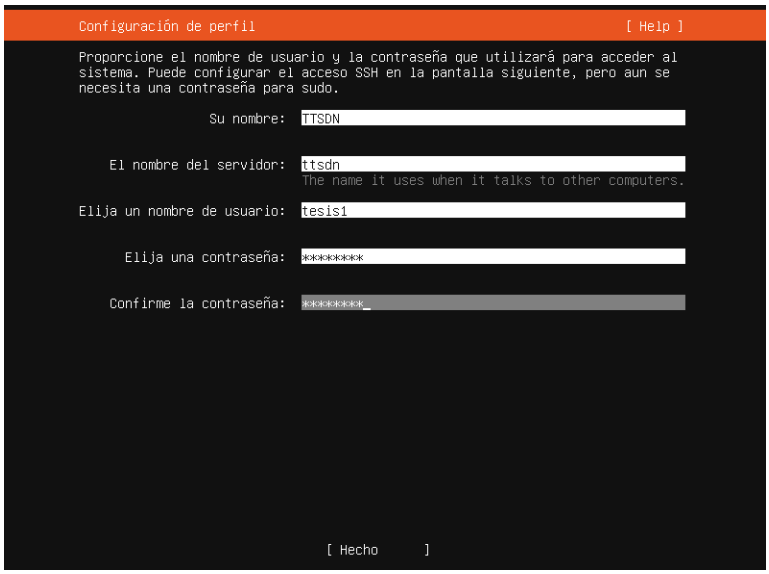


Figura 2: Configuración del perfil de usuario

4. Revisión de las funcionalidades adicionales que se requieran configurar en cada ventana emergente. La Figura 3 evidencia los archivos de instalación utilizados.

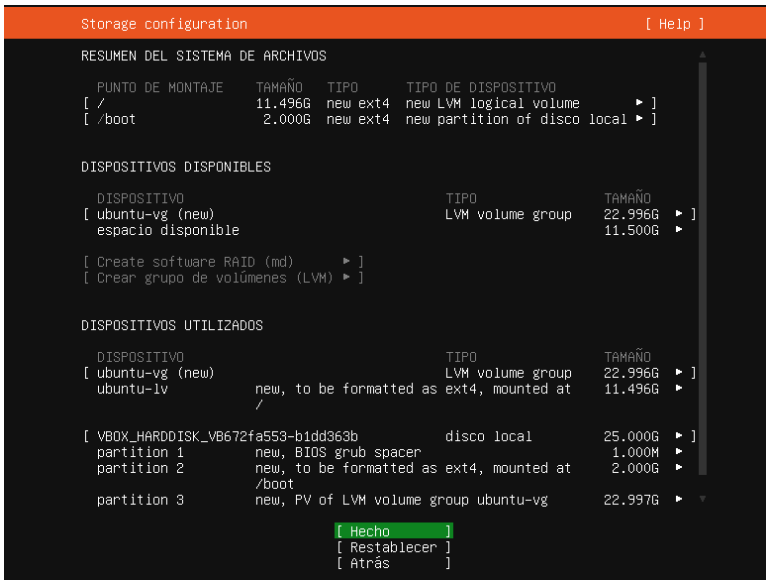


Figura 3: Configuración de instalación del servidor *Ubuntu*

5. En este punto el sistema se instala correctamente y luego de ingresar con las credenciales configuradas, se ejecutan los comandos del Extracto de código 1 de actualización de repositorios.

Extracto de código 1: Actualización del sistema en Ubuntu

```
1 sudo apt-get update
2 sudo apt-get upgrade
```

Anexo B: Guía para la instalación y configuración de la plataforma de código abierto, *Docker*

Para comenzar, se abre una terminal en el servidor de *Ubuntu*, donde se ingresan los comandos que se explican a continuación:

1. El comando del Extracto de código 2 instala utilidades esenciales en sistemas *Ubuntu*. Además, facilita descargas de paquetes a través de Hypertext Transfer Protocol Secure (HTTPS), proporciona certificados para validar conexiones seguras, permite transferir datos usando varios protocolos y simplifica la gestión de repositorios de software.

Extracto de código 2: Instalación de requerimientos

```
sudo apt install apt-transport-https ca-
    certificates curl software-properties-common
```

2. El comando del Extracto de código 3 descarga y añade la clave GNU Privacy Guard (GPG) del repositorio de *Docker* para *Ubuntu*, lo que permite verificar la autenticidad de los paquetes descargados desde dicho repositorio.

Extracto de código 3: Verificación de repositorios

```
curl -fSSL https://download.docker.com/linux/
    ubuntu/gpg | sudo apt-key add -
```

3. El comando del Extracto de código 4 agrega el repositorio de *Docker* para *Ubuntu* al sistema, facilitando la instalación y actualización del *software Docker*.

Extracto de código 4: Adición del repositorio Docker

```
sudo add-apt-repository "deb␣[arch=amd64]␣https://
    download.docker.com/linux/ubuntu␣focal␣stable"
```

4. El comando `sudo apt- install docker-ce` instala *Docker Community Edition* en el sistema, permitiendo la creación y gestión de contenedores.

De este modo, se instala *Docker* en el servidor *Ubuntu*. Una manera de verificar si la instalación fue correcta es ingresar el comando `sudo systemctl status docker`. Este comando verifica el estado del servicio *Docker* en activo, como se presenta en la Figura 4.

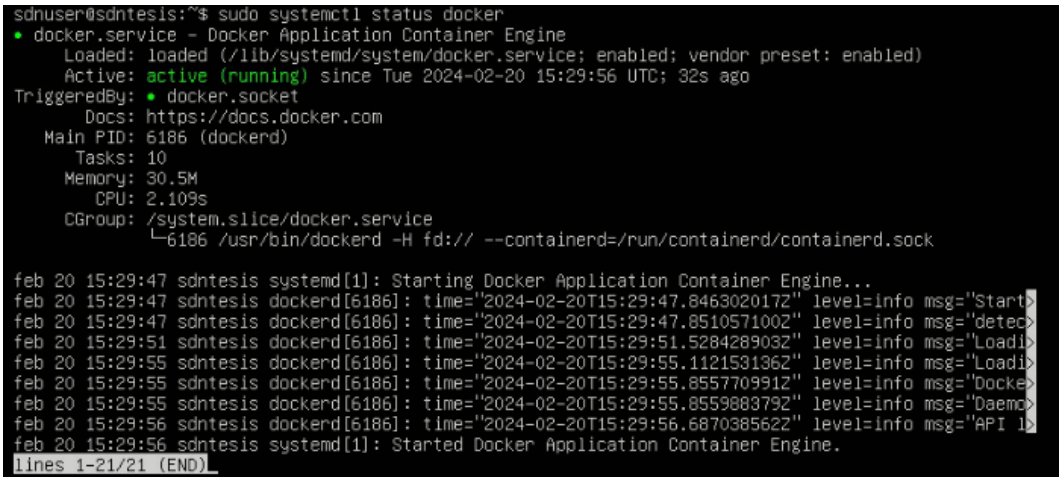


Figura 4: Estado del servicio *docker*

Anexo C: Guía para la instalación y configuración del servidor de *Logs* centralizado, *Graylog*

1. Primero, se crea el archivo de configuración de *Graylog*. Para ello, en una carpeta creada específicamente para *Graylog*, se genera un archivo llamado docker-compose.yml, mediante el comando sudo nano docker-compose.yml.
2. El archivo de configuración docker-compose.yml define un entorno de múltiples servicios utilizando *Docker Compose*. Se establecen tres servicios: *mongodb*, *opensearch*, y *graylog*.
   - El servicio *mongodb* utiliza la imagen más reciente de *MongoDB*, expone el puerto 27017, y almacena sus datos en un volumen llamado graylogd_mongodb_data.
   - El servicio *opensearch* utiliza la imagen más reciente de *OpenSearch*. Se le asigna el nombre de contenedor *opensearch* y expone el puerto 9200, estándar para las comunicaciones Hypertext Transfer Protocol (HTTP) con *OpenSearch*. También, se definen varias variables de entorno para su configuración:
     - discovery.type=single-node configura *OpenSearch* para funcionar como un solo nodo.
     - ES_JAVA_OPTS=-Xms512m -Xmx1g establece las opciones de memoria *Java*, asignando un mínimo de 512 MegaBytes (MB) y un máximo de 1 GB de RAM.
     - plugins.security.ssl.http.enabled=false y plugins.security.disabled=true deshabilitan la seguridad y el Secure Sockets Layer (SSL) para las conexiones HTTP.
     - OPENSEARCH_INITIAL_ADMIN_PASSWORD=SomePasswordPepper123 \# define una contraseña inicial para el administrador de *OpenSearch*.

El servicio *opensearch* está conectado a la red *graylog-net* y utiliza el volumen graylogd_opensearch_data para almacenar sus datos persistentes en /usr/share/opensearch/data.

- El servicio *graylog* utiliza la imagen graylog/graylog:5.2, que indica el uso de la versión 5.2 de *Graylog*. Se le asigna el nombre de contenedor *graylog* y expone una serie de puertos necesarios para su operación: 9000 para la interfaz World Wide Web (Web) de *Graylog*, 514, 4514, 5514, 6514 y 7514 para recibir *logs* en formato System Logging Protocol (Syslog). Del mismo modo, se definen varias variables de entorno para su configuración:
  - GRAYLOG_PASSWORD_SECRET=somepasswordpepper es una clave secreta usada para encriptar contraseñas y otros datos sensibles.
  - GRAYLOG_HTTP_EXTERNAL_URI=http://0.0.0.0:9000/ es la Uniform Resource Locator (URL) externa para acceder a la interfaz Web de *Graylog*.
  - GRAYLOG_WEB_ENDPOINT_URI=http://127.0.0.1:9000/api es la URL del *endpoint* de la *Application Programming Interface* (API) Web de *Graylog*.
  - GRAYLOG_ROOT_PASSWORD_SHA2=********** es la contraseña del usuario administrador de *Graylog* en formato Secure Hash Algorithm (SHA) (SHA-256).
  - GRAYLOG_MONGODB_URI=mongodb://mongodb:27017/graylog es la Uniform Resource Identifier (URI) de conexión a la base de datos *MongoDB*.
  - GRAYLOG_ELASTICSEARCH_HOSTS=http://opensearch:9200 es la URL para conectar *Graylog* a *OpenSearch*.

  El servicio *graylog* depende de que los servicios *mongodb* y *opensearch* estén en funcionamiento antes de iniciarse. Este servicio está conectado a la red *graylog-net* y utiliza dos volúmenes para el almacenamiento persistente, que son los siguientes: graylogd_graylog_data para almacenar los datos de *Graylog* en /usr/share/graylog/data y graylogd_graylog_logs para almacenar los *logs* de *Graylog* en /var/log/graylog. Esta configuración permite que *graylog*, *opensearch*, y *mongodb* funcionen juntos como una pila completa de registro y análisis de datos.

Todos los servicios están conectados a una red personalizada *graylog-net* que utiliza el *driver bridge* para facilitar la comunicación entre los contenedores.

```
# Se especifica la versión de Docker Compose que se está utilizando.
version: '3'

services:
  # Definición del servicio MongoDB.
  mongodb:
    # Imagen de Docker a utilizar para MongoDB, se usa la última versión disponible.
    image: mongo:latest
    # Nombre del contenedor.
```

```
    container_name: mongodb
    # Mapeo del puerto 27017 (MongoDB) del contenedor al host.
    ports:
      - "27017:27017"
    # Conexión del contenedor a la red personalizada 'graylog-net'.
    networks:
      - graylog-net
    # Montaje de volumen para persistencia de datos en MongoDB.
    volumes:
      - /opt/graylog/mongo_data:/data/db

  # Definición del servicio OpenSearch.
  opensearch:
    # Imagen de Docker para OpenSearch, se usa la última versión disponible.
    image: opensearchproject/opensearch:latest
    # Nombre del contenedor.
    container_name: opensearch
    # Mapeo del puerto 9200 (OpenSearch) del contenedor al host.
    ports:
      - "9200:9200"
    # Variables de entorno para la configuración de OpenSearch.
    environment:
      # Configuración para que OpenSearch funcione en modo de un solo nodo.
      - discovery.type=single-node
      # Configuración de memoria para Java en OpenSearch.
      - ES_JAVA_OPTS=-Xms512m -Xmx1g
      # Deshabilitar SSL para HTTP.
      - plugins.security.ssl.http.enabled=false
      # Deshabilitar la seguridad en OpenSearch.
      - plugins.security.disabled=true
      # Contraseña inicial para el administrador de OpenSearch.
      - OPENSEARCH_INITIAL_ADMIN_PASSWORD=SomePasswordPepper123#
    # Conexión del contenedor a la red personalizada 'graylog-net'.
    networks:
      - graylog-net
    # Montaje de volumen para persistencia de datos en OpenSearch.
    volumes:
      - /opt/graylog/opensearch_data:/usr/share/opensearch/data

  # Definición del servicio Graylog.
  graylog:
    # Imagen de Docker para Graylog, se especifica la versión 5.2.
    image: graylog/graylog:5.2
    # Nombre del contenedor.
    container_name: graylog
    # Variables de entorno para la configuración de Graylog.
    environment:
      # Clave secreta para encriptar contraseñas.
      - GRAYLOG_PASSWORD_SECRET=somepasswordpepper
      # URL externa para acceder a la interfaz web de Graylog.
      - GRAYLOG_HTTP_EXTERNAL_URI=http://0.0.0.0:9000/
      # URL del endpoint de la API web de Graylog.
      - GRAYLOG_WEB_ENDPOINT_URI=http://127.0.0.1:9000/api
      # Contraseña del usuario administrador en formato SHA-256.
      - GRAYLOG_ROOT_PASSWORD_SHA2=aea3cc31f52ba4b24b56d2d9f3652b17818
      # URI de conexión a la base de datos MongoDB.
      - GRAYLOG_MONGODB_URI=mongodb://mongodb:27017/graylog
      # URL para conectar Graylog a OpenSearch.
      - GRAYLOG_ELASTICSEARCH_HOSTS=http://opensearch:9200
```

```
      # Configuración de la zona horaria para Graylog.
      - GRAYLOG_ROOT_TIMEZONE=America/Guayaquil
    # Mapeo de puertos para Graylog.
    ports:
      # Puerto para la interfaz web de Graylog.
      - 9000:9000
      # Puertos para recibir logs en formato Syslog.
      - 3514:3514
      - 3514:3514/udp
      - 4514:4514
      - 4514:4514/udp
      - 5514:5514
      - 5514:5514/udp
      - 6514:6514
      - 6514:6514/udp
      - 7514:7514
      - 7514:7514/udp
    # Conexión del contenedor a la red personalizada 'graylog-net'.
    networks:
      - graylog-net
    # Dependencias del servicio Graylog, asegura que estos servicios se inicien antes.
    depends_on:
      - mongodb
      - opensearch
    # Montaje de volumen para el journal de Graylog.
    volumes:
      - /opt/graylog/journal:/usr/share/graylog/data/journal
      # Montaje de volumen para la configuración de Graylog.
      - /var/lib/docker/volume-
      ↪ s/bc43294758fdbdf0ea7c60c5dea534a8272f89d6f618/_data/config:/usr/share/graylog/data/config

# Definición de una red personalizada para los servicios.
networks:
  # Tipo de driver para la red, en este caso 'bridge'.
  graylog-net:
    driver: bridge
```

Código 3: Código utilizado para la configuración del servidor de *Logs* centralizado, *Graylog*

3. Luego de guardar el archivo de configuración *docker-compose.yml*, se ejecuta el comando docker-compose up, con la finalidad de iniciar y ejecutar todos los contenedores definidos en el archivo, en el directorio actual.

4. Para asegurarse de que todos los contenedores se han iniciado de manera correcta, ejecutar el siguiente comando del Extracto de código 5.

Extracto de código 5: Contenedores en ejecución

```
docker ps
```

. La salida e información de los contenedores iniciados se muestra en la Figura 5.

Figura 5: Información de los contenedores iniciados

El comando $\mathrm{docker\ ps}$ proporciona una lista de los contenedores *Docker* que están actualmente en ejecución en el sistema. La información que muestra es la siguiente:

- Identificador único del contenedor (*CONTAINER ID*).
- Imagen *Docker* desde la que se creó el contenedor (*IMAGE*).
- Comando que se está ejecutando dentro del contenedor (*COMMAND*).
- Tiempo desde que fue creado el contenedor (*CREATED*).
- Estado actual del contenedor (*STATUS*).
- Puertos mapeados del contenedor al *host* (*PORTS*).
- Nombre asignado al contenedor (*NAMES*).