Sistema de automatización de pruebas para testeo de sistemas de monitoreo de subestaciones eléctricas.

Nicolas Perez Sanchez

Informe Practica académica para optar al título de Ingeniero Electrónico

Asesores
Orlando Carrillo Perilla, Ingeniero Electrónico
Alejandro Restrepo Medina, Ingeniero Electrónico

Universidad de Antioquia
Facultad de Ingeniería
Pregrado en Ingeniería Electrónica
Medellín
2022

Cita	Perez Sanchez [1]
Referencia Estilo IEEE (2020)	[1] N. Perez Sanchez, “Sistema de automatización de pruebas para testeo de sistemas de monitoreo de subestaciones eléctricas”, Trabajo de grado profesional, Ingeniería Electrónica, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2022.

Tabla de Contenido

Resumen

En el presente informe se muestra la recopilación del trabajo realizado durante el semestre de industria, llevado a cabo por el estudiante en un periodo de 6 meses. En este se puede observar una introducción a la temática a tratar, relacionada con el desarrollo de los sistemas de monitoreo y su importancia en la medición de la calidad de la energía de instalaciones críticas y la necesidad de automatizar ciertas pruebas para el testeo de la calidad y funcionalidad de dichos sistemas, debido a la complejidad que estos conllevan. Se muestra el marco teórico donde se definen todos los conceptos mencionados en este informe, donde se pretende dar una definición clara de todos los conceptos tratados durante el trabajo de prácticas, se muestra también los objetivos que se plantearon al iniciar este proyecto de practica académica y la metodología que se siguió para poder completarlos. Por último, se muestra detalladamente los resultados obtenidos a lo largo de los 6 meses de la práctica, y las conclusiones que se obtuvieron una vez finalizado y entregado este proyecto en su totalidad.

Introducción

En el trabajo diario, las fuentes de alimentación se desvían de esta situación ideal porque las cargas del sistema varían, y pueden producirse fenómenos transitorios y apagones. Si la calidad de la energía de la red es buena, las cargas conectadas a ella funcionarán de manera satisfactoria y eficiente. Sin embargo, una calidad de energía insuficiente puede provocar averías en la maquinaria, en los sistemas de control eléctrico o en las computadoras conectadas a la red de suministro eléctrico.

Las mediciones de la calidad de la energía definen el grado en que un suministro práctico se asemeja a la situación ideal en términos de contaminación armónica, potencia reactiva y desequilibrio de carga.

Esto implica mediciones en el suministro incluyendo frecuencia, interrupciones, parpadeo, voltaje e Inter armónicos, variaciones de voltaje como inmersiones, sobrevoltaje temporal o cambios rápidos y desequilibrio de voltaje.[5]

Qubits Energy proporciona servicios y soluciones personalizadas para instalaciones de energía crítica, centros de datos y edificios.Con más de dos décadas de experiencia de campo trabajando en monitoreo de energía y automatización de edificios y “critical facilities”, Qubits Energy se enfoca en crear aplicaciones confiables personalizadas para el control y la optimización de la energía,así como la automatización de edificios. Los sistemas de monitoreo (SCADA) son sistemas robustos, que implican bastante tiempo y desarrollo debido a la cantidad de equipos presentes en una subestación eléctrica y la gran cantidad de variables que cada uno de estos puede obtener y la información que deben estar enviando constantemente para tener conocimiento de que como es el fluido energético de las instalaciones criticas donde esté funcionando.

Una de las tareas más engorrosas cuando se está desarrollando un sistema de monitoreo es la implementación de las pruebas de funcionamiento y análisis de issues del sistema, cosa que, hasta el momento, se realiza manualmente por un ingeniero encargado de la revisión de todos los proyectos que se desarrollan en la empresa. Por ello, se plantea el proyecto de desarrollar un sistema de automatización de pruebas que permita facilitar la revisión de los proyectos, no solo al ingeniero encargado de las pruebas, sino a todos los encargados de desarrollar estos sistemas.

Objetivos

Objetivo general

Desarrollar un sistema de automatización de pruebas de SCADA´s usando C#, librerías de Python, y POWER SCADA OPERATION para agilizar el tiempo que lleva realizar las pruebas de un sistema de monitoreo de una subestación eléctrica.

Objetivos específicos

• Desarrollar un sistema de monitoreo (SCADA) para una subestación a partir de un plano eléctrico suministrado por un cliente, con la herramienta de POWER SCADA OPERATION, para entender el desarrollo de estos sistemas y las variables que lo componen.

• Analizar diagramas de red para conocer las direcciones IP que deben asignarse a los equipos de monitoreo, la configuración que poseen, como se están comunicando los dispositivos y que información envían.

• Realizar el testeo de un sistema de monitoreo con el ingeniero encargado, para conocer el procedimiento que se realiza al revisar un SCADA y hacer un levantamiento de prerrequisitos sobre qué es lo que se desea automatizar de este proceso.

• Realizar un sistema de automatización de pruebas funcional, en base a los requerimientos establecidos en el ítem anterior.

• Desarrollar una GUI usando Python, para que cualquier ingeniero pueda usar fácilmente las pruebas automatizadas y pueda revisar un proyecto sin tener que interactuar con el código fuente.

Marco Teórico

EcoStruxure™ Power SCADA Operation es un software que ofrece adquisición de datos, control y monitorización para las instalaciones de distribución eléctrica, es también, una plataforma flexible, segura, escalable y redundante. No importa el tamaño o la complejidad, este software está diseñado para supervisar cada parte de un sistema eléctrico, el cual por lo general necesita proporcionar energía confiable las 24 horas del día, los 7 días de la semana. EcoStruxure Power SCADA Operation, una solución de EcoStruxure Power, está diseñada para ayudar a las instalaciones críticas y las operaciones electro intensivas a maximizar el tiempo de actividad.[1]
Un **SCADA** es un control de Supervisión y adquisición de datos que se compone de elementos de software y hardware que permite: Controlar los procesos industriales localmente o en ubicaciones remotas, Supervisar, recopilar y procesar datos en tiempo real, Interactuar directamente con dispositivos como sensores, válvulas, bombas, motores y más a través del software de interfaz hombre-máquina (HMI).
Los sistemas SCADA son cruciales para las organizaciones industriales, ya que ayudan a mantener la eficiencia, procesan datos para tomar decisiones más inteligentes y comunican los problemas del sistema para ayudar a mitigar el tiempo de inactividad. La arquitectura SCADA básica comienza con controladores lógicos programables (PLC) o unidades terminales remotas (RTU)[2]

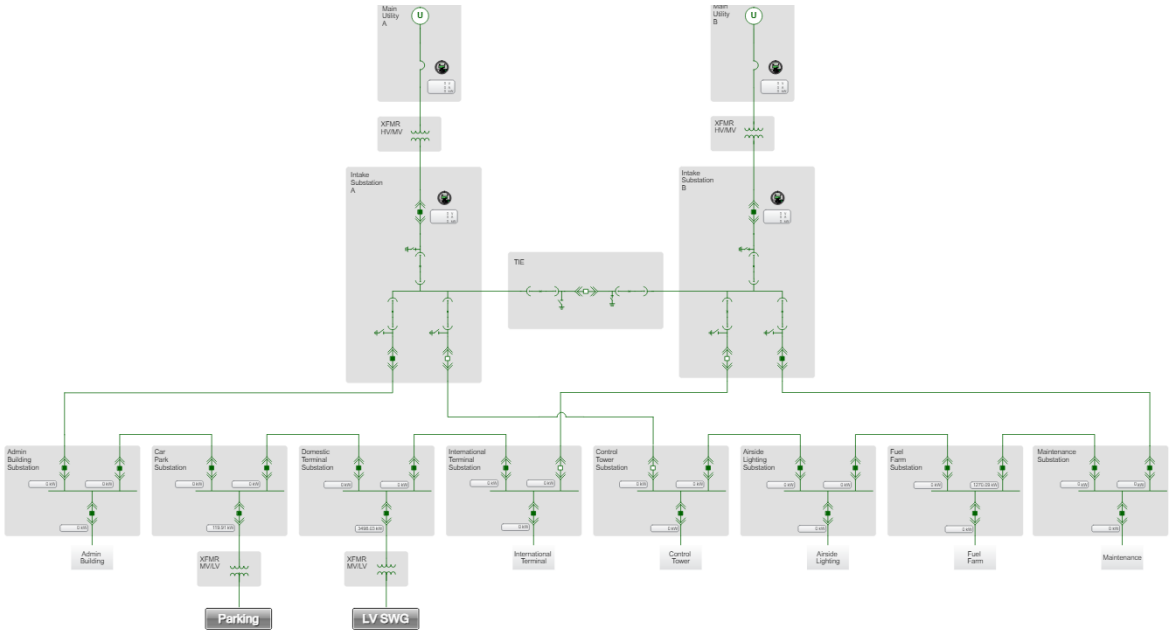

Figura 1. Diagrama unifilar en sistema SCADA.

Las **instalaciones críticas (“critical facilities”)** incluyen hospitales, estaciones de bomberos, estaciones de policía, almacenamiento de registros críticos o datacenters, e instalaciones similares. Para el campo eléctrico, estas instalaciones consideradas de vital importancia deben recibir una consideración especial ya que son ubicaciones en las que se tiene que asegurar una red constante, fiable y controlada de energía, para la mayoría de los casos, de baja y media potencia.

Cicode es un lenguaje de programación diseñado para usarse en Citect SCADA para monitorear y controlar equipos de planta. Es un lenguaje estructurado similar a Visual Basic o 'C'. Puede usar Cicode mediante su interprete, **Cicode Editor**, para acceder a datos en tiempo real (variables) en un proyecto Citect SCADA y las instalaciones de Citect

SCADA: etiquetas de variables, alarmas, tendencias, informes, etc. También puede interactuar con varios componentes en una computadora, como el sistema operativo y los puertos de comunicación. Cicode también admite funciones avanzadas que incluyen multitarea adelantada, subprocesos múltiples y llamadas a procedimientos remotos.[3]

Error humano es una expresión que indica que un suceso desfavorable está fuertemente condicionado por la actividad de las personas que participan directa o indirectamente en la realización y control de un proceso, a veces se puede atribuir a una mala praxis de las personas implicadas. Ha sido citado como una causa o factor influyente en catástrofes y accidentes en industrias.[6]

XML es un lenguaje de marcado similar a HTML. Significa Extensible Markup Language (Lenguaje de Marcado Extensible) y es una especificación de W3C como lenguaje de marcado de propósito general. Esto significa que, a diferencia de otros lenguajes de marcado, XML no está predefinido, por lo que debes definir tus propias etiquetas. El propósito principal del lenguaje es compartir datos a través de diferentes sistemas, como Internet.[9]

PM800 es un medidor de potencia de la serie PowerLogic PM800 ofrecen capacidades de medición de alto rendimiento necesarias para monitorear una instalación eléctrica en una unidad compacta de 96 x 96 mm. La pantalla grande y fácil de leer del medidor de potencia le permite ver las tres fases y el neutro al mismo tiempo.

Las características estándar de los medidores de potencia de la serie PM800 incluyen un puerto de comunicación RS485 Modbus (ASCII y RTU) y Ethernet Modbus TCP/IP (opcional), entrada digital, salida digital, medición THD y alarmas. Además, los modelos PM820 y PM850 ofrecen registros integrados personalizados y lecturas armónicas de tensión y corriente individuales.[7]

Figura 2. Medidor de energía PM800

PM5000 Analizadores de red versátiles y compactos para aplicaciones de estudio de costes energéticos y análisis de redes eléctricas. Medidores de energía multifunción para aplicaciones de asignación de costos, redes de monitoreo y seguimiento a insumos energéticos WAGES [8]

Figura 3. Medidor de energía PM5000

Un **Datacenter** es una infraestructura física o virtual utilizada para alojar sistemas informáticos que puedan procesar, servir o almacenar datos. Data Center dan servicio de almacenamiento de datos, respaldo o backup, recuperación de datos y gestión de la información para empresas. En estos datacenters se encuentran componentes como Sistemas de seguridad, Monitorización, Climatización, Energía, Conectividad de red y Servidores.

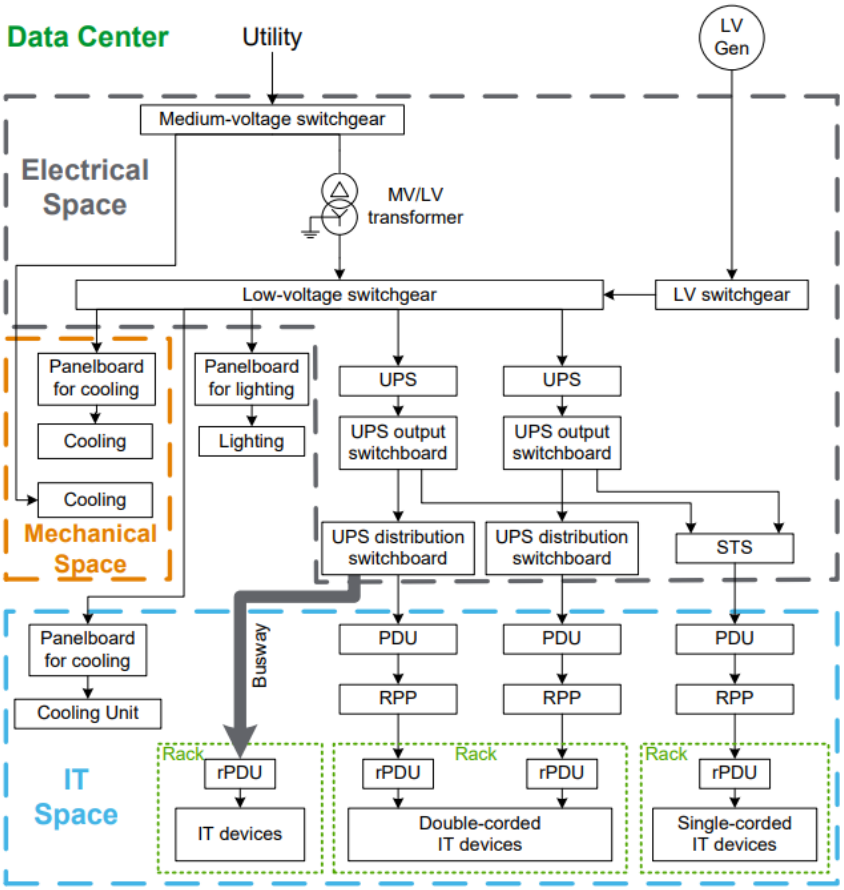

Figura 3. Distribución de una subestación eléctrica dentro de un datacenter[10]

Metodología

El desarrollo de este proyecto se dividió en tres partes principales:

Parte 1-Desarrollo del sistema SCADA

- Interpretación de los diagramas de Red y plano eléctrico de la subestación eléctrica, se debía comprender la distribución y capacidad de la subestación para la digitalización de estos planos y con ello se da el inicio del desarrollo del sistema de supervisión.

- Se desarrollo el sistema de supervisión, abarcando todas las zonas necesarias y cumpliendo todos los requerimientos establecidos por el cliente, este sistema incluye una visualización detallada y monitoreo de cada uno de los equipos de medición y su ubicación en el diagrama unifilar, se incluyeron vistas de cada una de las zonas de la subestación, donde se diferencian la red de alimentación primaria y secundaria, los cuartos donde se encuentran los servidores del cliente, la ubicación de los generadores de energía de emergencia y demás equipos presentes en la localidad del proyecto, y donde cada viste permite obtener información de las diferentes variables eléctricas monitoreables mediante los equipos instalados en el sitio.

-Agregar todos los equipos necesarios según el plano eléctrico y con las configuraciones de direccionamiento IP según el diagrama de Red, se desarrolla y revisa cuidadosamente que se agreguen todos los medidores, generadores, breakers, switches, etc, correspondientes a los que están instalados en la subestación, y que las configuraciones de red con las que estos están configurados físicamente sean iguales en el software para asegurar una correcta comunicación, y por consiguiente el correcto despliegue de la información medida.

Parte 2- Testeo manual del sistema de supervisión y todos los componentes desarrollados.

Usna vez finalizado el desarrollo del sistema de monitoreo:

- e realizó una revisión manual de cada uno de los elementos que componen el SCADA(medidores, generadores, breakers, switches, etc.) direccionamientos de estos elementos y verificación de que no haya redundancia entre equipos o zonas del sistema, también se verificó que todo el entorno gráfico, con el cual el cliente interactúa constantemente para monitorear su subestación, se comporte como debería.

- Verificación de que todos los requerimientos y funcionamientos del sistema estén en correcto estado, para ello al inicio de cada proyecto, se determinan objetivos y lineamientos que tienen que ser completados al finalizar el proyecto, la calidad del sistema y presentación, asegurándose de que todas las funcionalidades de las herramientas desarrolladas, trabajen de acuerdo a lo acordado con el cliente y revisión de todas las tareas pendientes, bugs y correcciones por parte de los ingenieros encargados de la revisión del sistema, antes de realizar la entrega del proyecto.

- Hacer entrega del proyecto al cliente e implementación del sistema con un ingeniero en campo, donde se conecta remotamente a los sistemas informáticos del cliente, y se realizan las diferentes configuraciones del sistema para

conectarse a todos los equipos de control y monitoreo, instalados previamente en la planta eléctrica y el testeo del sistema, asegurándose de que se haya conectado y comunicado exitosamente con los equipos, y que la información que se esté recibiendo, sea la esperada.

Parte 3- Desarrollo de sistema de automatización de pruebas del SCADA

- Se realizó un levantamiento de prerrequisitos en base a las necesidades vistas cuando se hizo la revisión manual de los proyectos, de manera que se sepa cuál es el primer paso y cuáles son los procesos más importantes que deben ser automatizados en un principio, dentro de estos estaba la correcta funcionalidad grafica de todos los equipos que componen el sistema de monitoreo(medidores, generadores, breakers, switches, etc), la configuración de red para cada uno de los equipos, donde se asegure que no haya equipos con la misma configuración, y que todas las medidas eléctricas que reporta cada uno de los equipos, se diferencien entre sí y sean las esperadas.

- Se desarrollaron scripts de automatización de pruebas, para testeo de los sistemas SCADA´s. Estos scripts fueron desarrollados utilizando el lenguaje de programación Python, que extraen la información de los archivos generado por el software de monitoreo, y que contienen toda la información sobre los equipos que se encuentran y que deben ser testeados; estos scripts se ejecutaban en el software de desarrollo del sistema de monitoreo, Power Scada Operation, mediante otros scripts desarrollados en el lenguaje de C#, y con ello, se testeaban automáticamente los elementos deseados del sistema de monitoreo.

- Se realizaron evaluaciones constantemente con el ingeniero encargado de las pruebas para verificar que el proceso que se está realizando en automático siga los lineamientos establecidos por la empresa, así, ejecutando los scripts de prueba se verificó que estos realizaran las pruebas de todos los parámetros necesarios; cuando no, era necesario hacer las correcciones o adiciones de funciones pertinentes a los scripts de Python y/o C#.

- Se realizó una implementación del sistema de automatización con diferentes ingenieros para verificar que el sistema pueda ser aplicado por cualquier persona y que sea general y aplicables para cualquier proyecto en desarrollo, para ello, se compartieron los scripts de automatización junto con un manual de funcionamiento a todos los ingenieros de la empresa, y se realizaba un acompañamiento al momento de realizar los procedimientos necesarios y con ello, se verificaba que el trabajo realizado, fuera funcional para cualquier proyecto y pudiera ser aplicado por cualquier ingeniero que lo necesitará.

Resultados y análisis

Se dio cumplimiento a cada uno de los objetivos planteados al comienzo de la práctica empresarial, siguiendo paso a paso la metodología explicada previamente y dejando documentado cada uno de los pasos seguidos y los resultados obtenidos que se muestran a continuación:

Sistema de Monitoreo y Control SCADA

De la siguiente figura se puede observar cómo se parte de un plano eléctrico, ya sea unifilar o trifilar, y cuyo objetivo principal es la digitalización de todos los planos diseñados para cada una de las zonas de la subestación eléctrica.

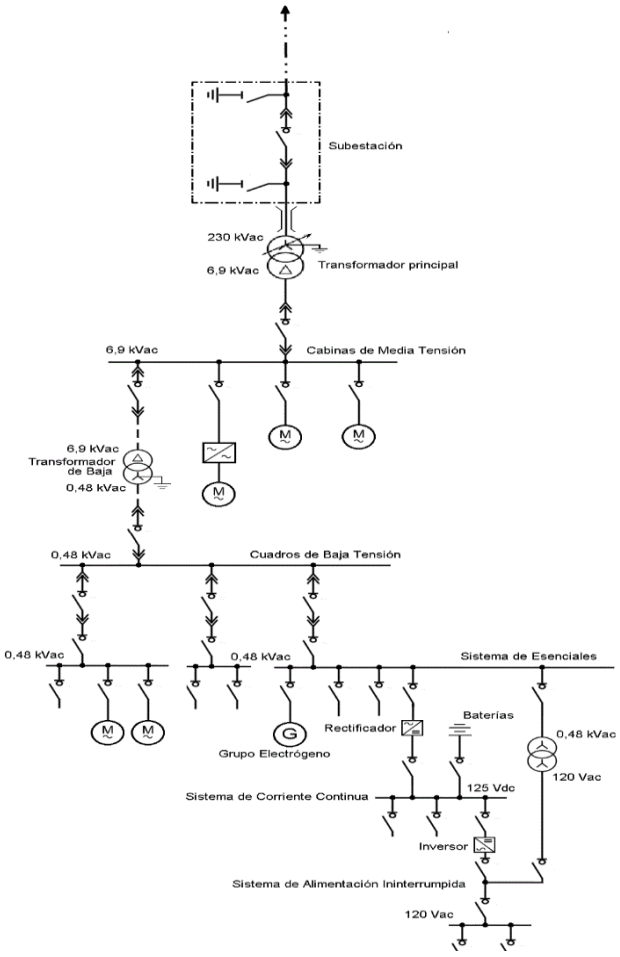

Figura 4. Ejemplo de plano eléctrico unifilar

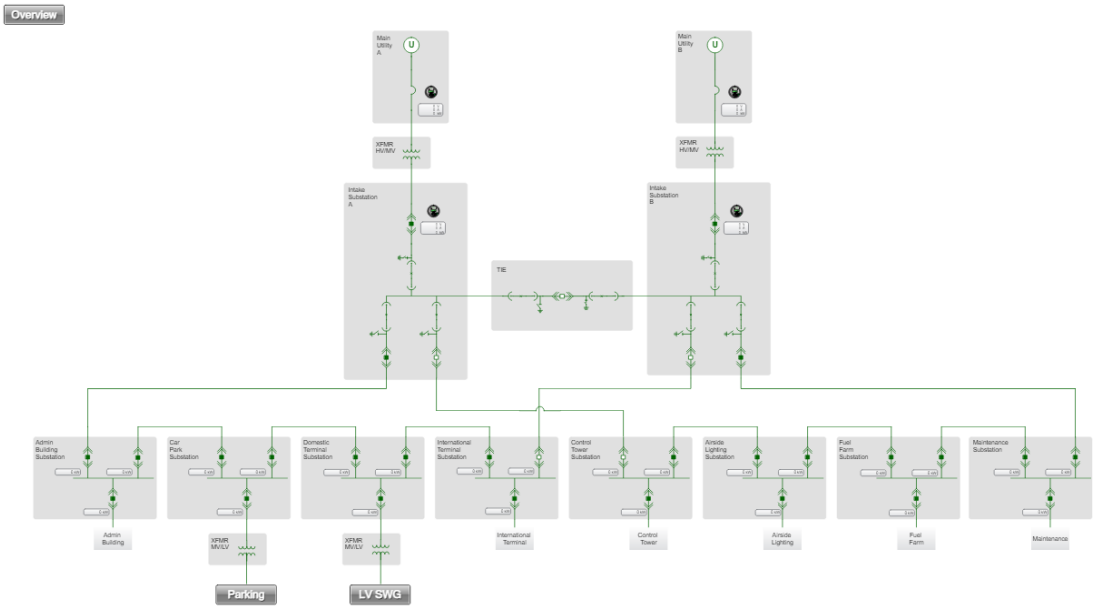

Figura 5. Diagrama unifilar digitalizado y basado de plano eléctrico

Para cada uno de los equipos presentes en el plano eléctrico, se desarrolla una ventana emergente (PopUp) como la mostrada a continuación:

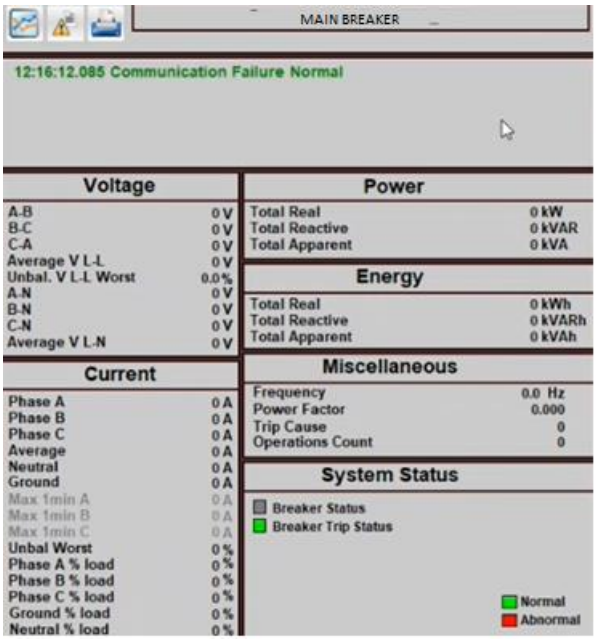

Figura 6. PopUp inactivo con diferentes medidas eléctricas.

Cada una de las variables eléctricas que se puede medir, se diferencian en el software mediante una etiqueta o “TAG”. Estas ventanas emergentes son de vital importancia, ya que como se ha mencionado anteriormente, el objetivo de estos sistemas es monitorear en todo momento y todo lugar de la subestación la calidad de la energía, y generar las debidas alertas cuando algo este fuera de los límites establecidos.

Para la asignación de las direcciones de red que debe tener cada uno de los equipos, esencial para la comunicación entre estos y el SCADA, se basa de los diagramas de red, como se observa a continuación:

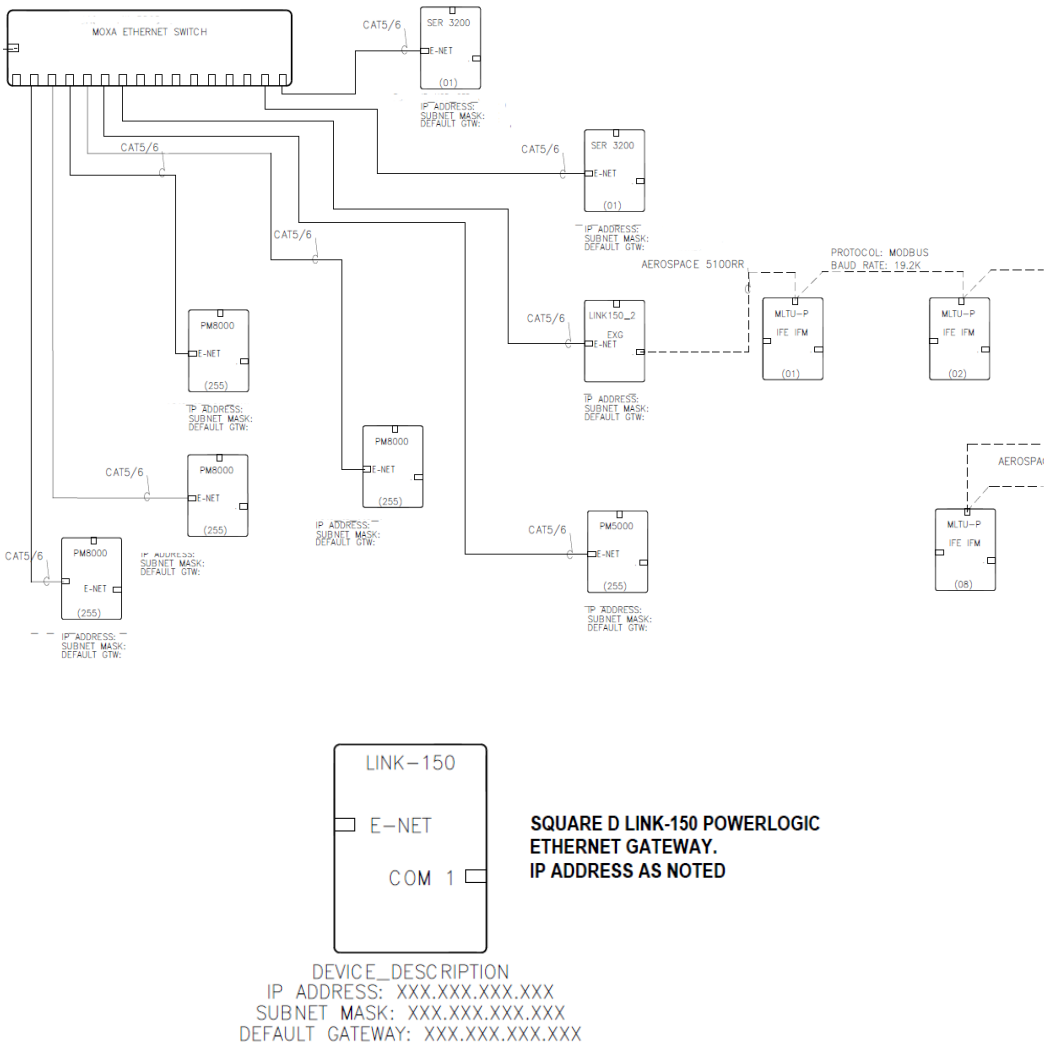

Figura 7. Ejemplo de Diagrama de Red de zona de la subestación.

Según estos diagramas, se sabe la configuración de los equipos, que se realizara tanto al momento de crear los dispositivos dentro del SCADA, como cuando se hace la configuración y puesta en marcha del sistema y los equipos en sitio.

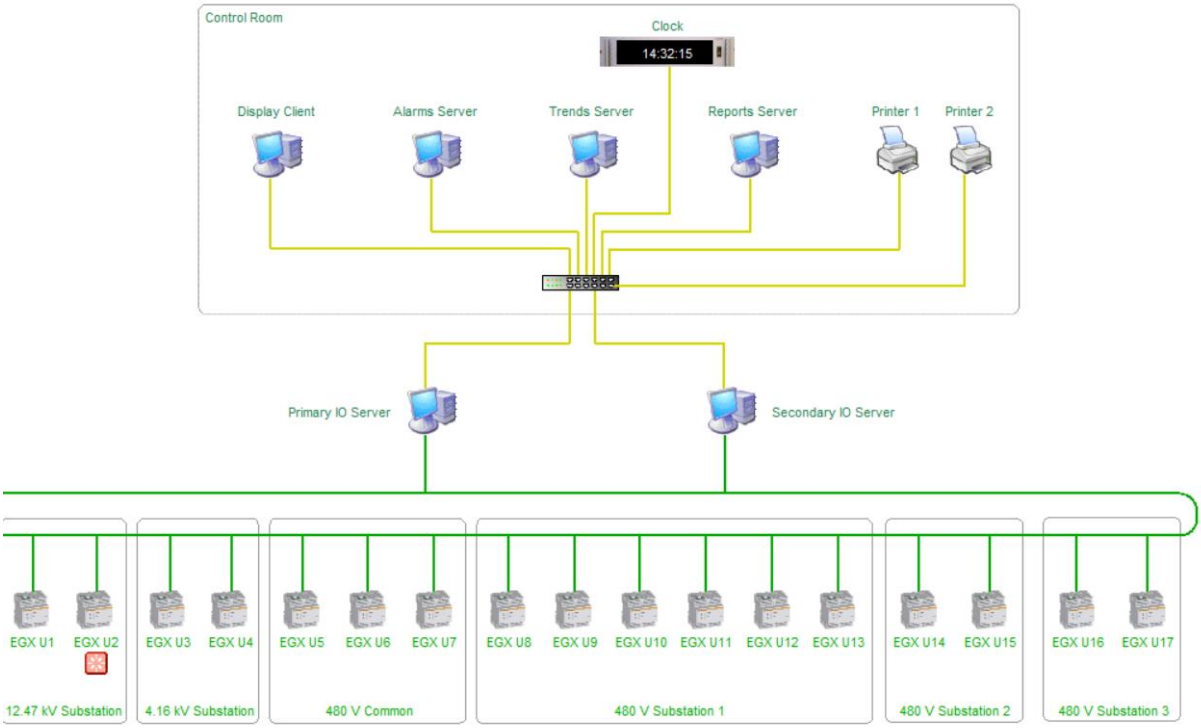

Figura 8. Diagrama de red graficado por el software de manera jerárquica.

Por último, se realiza una vista general de la subestación, la cual funciona como página principal del sistema y de donde se podrá partir para visualizar una zona detallada de la subestación, esta distribución puede partir del diagrama ilustrado en la figura (3) donde se observa de que equipos y zonas está compuesto un datacenter o critical facilities.

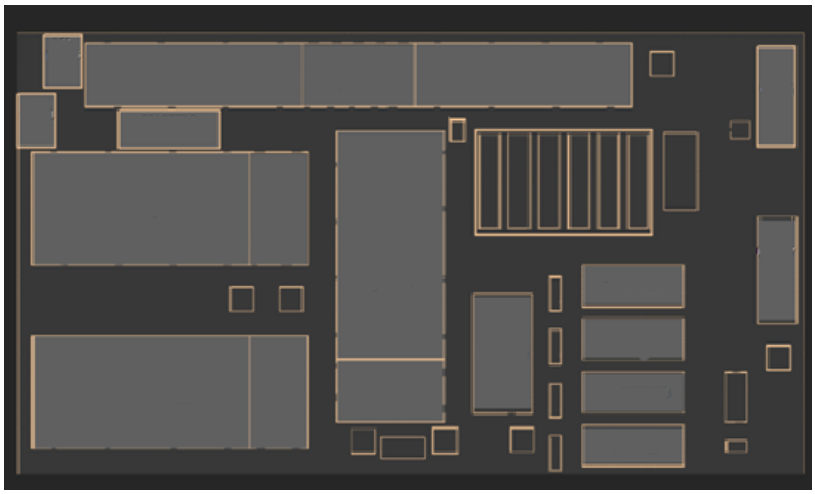

Figura 9. Distribución de subestación eléctrica

En esta vista se puede observar la ubicación de los datacenter, red principal de energía, ubicación de generadores de emergencia y UPS, y todo el circuito eléctrico que compone una subestación eléctrica.

Testeo de Sistema SCADA

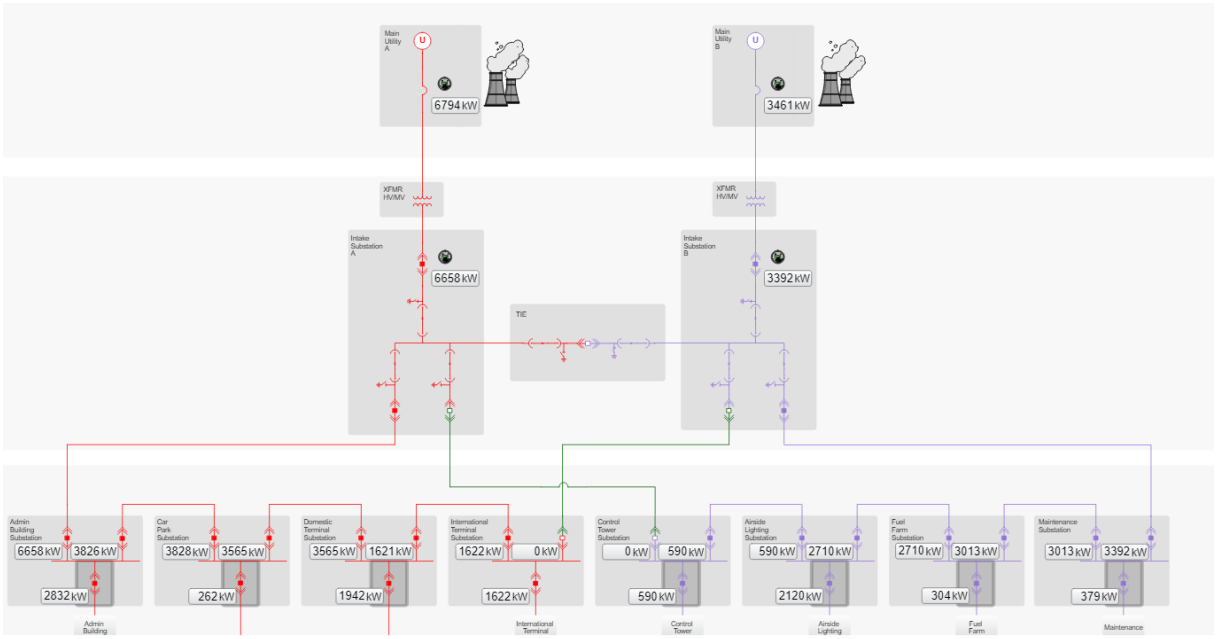

Figura 10. Diagrama unifilar Energizado por entrada principal y secundaria.

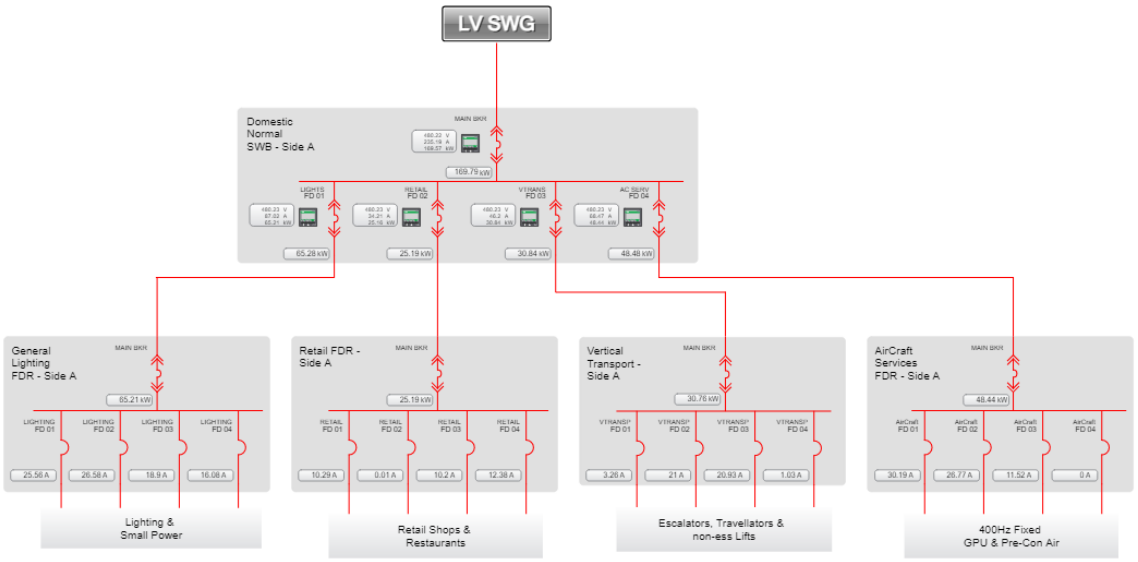

Figura 11. Diagrama unifilar Energizado por entrada principal.

En las figuras (10) y (11) se puede observar como un diagrama unifilar es testeado, primero se activa la red principal de energía manualmente, así como cada uno de los componentes aguas abajo (breakers, medidores, generadores, ups, etc.). Se denota con el color rojo cuando en el diagrama unifilar un bus y componente se encuentran energizados, con color verde cuando se encuentra desenergizada otra zona, y con color morado o naranja cuando se está energizando con una fuente secundaria como generador o baterías.

Con esta prueba se revisa que las conexiones de cada uno de los dispositivos y buses que componen el diagrama unifilar se encuentren correctamente configurados, también se revisa en el caso de los dispositivos que permiten el paso de energía de un bus a otro, funcionen correctamente cuando son activados, y que cuando sean desactivados,

interrumpan el paso de energía; para el caso de generadores, se revisa que estos energicen el unifilar una vez que sean activados y la red de energía principal sea desactivada. Así uno por uno, se debe activar manualmente todos los componentes que componen cada uno de los diagramas unifilares, de las zonas presentes en la subestación eléctrica.

Para testear las variables que obtiene directamente cada uno de los medidores y equipos, se abren los popups desarrollados de manera personalizada para cada equipo presente en el diagrama unifilar. Para estos PopUp, a cada variable se les asigna un valor manualmente usando un depurador, y cuyo resultado se puede observar a continuación:

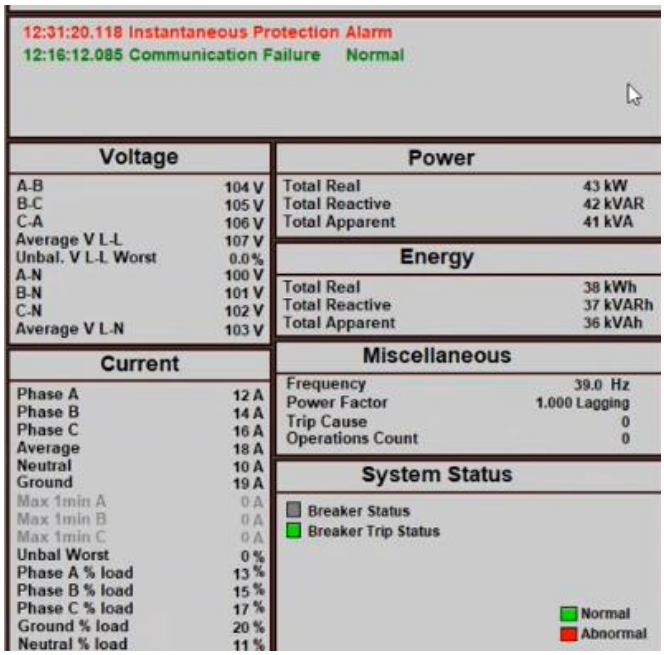

Figura 12. PopUp activo con diferentes valores para cada variable eléctrica.

El objetivo de esta prueba es verificar que los tags con los que están definidas cada una de estas medidas eléctricas se diferencien entre sí, y evitar que una variable este mal direccionada o incluso duplicada de alguna otra, y pueda generar confusión al momento de supervisar el estado de las medidas eléctricas en un equipo dentro de la subestación.

Como se pudo observar, el testeo de un sistema de monitoreo, donde se deben revisar el correcto desarrollo de cada una de las zonas de una subestación, que pueden ser cientos, y donde cada una de estas zonas está compuesta por muchos equipos, es un trabajo arduo y engorroso, donde persiste mucho el error humano, ya que es difícil garantizar de que se revisaron a detalle cada uno de los componentes del sistema, y donde el tiempo que se invierte para la revisión es muy alto.

Scripts de automatización de pruebas

Se muestra a continuación, los scripts en lenguaje Python y C#, desarrollados por el practicante, y encargados de extraer la información necesaria del SCADA, para conocer que equipos componen cada una de las zonas de la subestación y con que etiquetas se identifican, con el objetivo de ejecutar dicha información, y testear fácilmente cada componente y zona de la subestación, en una fracción del tiempo que toma hacer el testeo manual de estas.

Para la extracción de información de una página completa, equivalente a un diagrama unifilar de una zona, se tiene el siguiente script, encargado de tomar toda la información de los dispositivos presentes y guardarlas en un archivo listo para ser ejecutado por el ingeniero

```
    file="..\\_config-files\\"+'AdvOneLine.csv'
    Project=''
    Titles = ['// Source Color Map', '// Sources', '// Breakers', '// ATSs', '// Meters', '// Transformers']
    Titles2 = ['//Source Color Map', '//Sources', '//Breakers', '//ATSs', '//Meters', '//Transformers']
    #If available, find project name, and it doesn´t have to be a INPUT parameter
    file_open = open(file, "rt")
    for line in file_open:

        res= line[0:-1]
        if '//' in res:
            if res not in Titles and res not in Titles2:
                if '.' in res:
                    if "// " in res:
                        Project =res[3:res.find(".")]    # Name of the Project
                    else:
                        Project =res[2:res.find(".")]
                    break

    with open(file, newline='') as ifile:
        reader = csv.reader(ifile, delimiter=",")


        rownum = 0
        pagenum = 0
        pages = []          # Pages in AdvOneLine
        cond = []           # Individual condition
        condAll = []        # All condition
```

```
files_created=[]
pages=[]
for i in range(len(filterCSV)):

    if filterCSV[i][0] in files_created:
        #verifies if a page of the project have been created, in order to not overwrite it
        et = ET2.parse("test_"+filterCSV[i][0]+".xml")
        for j in range(len(filterCSV[i][1])):
            for k in range(int(len(filterCSV[i][1][j][1])/3)):
                child=ET.SubElement(pages[files_created.index(filterCSV[i][0])], "tagWrite", Value=filterCSV[i][1
                child.text=filterCSV[i][1][j][1][k*3]
                #exception for ATS
                if "ATS" in filterCSV[i][1][j][1][k*3]:
                    ET2.SubElement(pages[files_created.index(filterCSV[i][0])], "sleepCall", Value="2").text=' '
                    child=ET.SubElement(pages[files_created.index(filterCSV[i][0])], "tagWrite", Value=filterCSV[
                    child.text=filterCSV[i][1][j][1][k*3]



        tree = ET2.ElementTree(pages[files_created.index(filterCSV[i][0])])
        indent(pages[files_created.index(filterCSV[i][0])])
        tree.write("test_"+filterCSV[i][0]+".xml", encoding="utf-8", xml_declaration=True)

    else: #creates a new xml file for a new project page
        pageName = ET.Element("script",name="Pararell",saveAt="C:\\", incremental="yes") #creates main node call
        pages.append(pageName)# save a xml configuration to add tags to the same xml file
        files_created.append(filterCSV[i][0])
        for j in range(len(filterCSV[i][1])):
```

Figura 13. Código para extracción de información, con etiquetas de equipos que componen una zona

```
<?xml version='1.0' encoding='utf-8'?>
<script name="Pararell" saveAt="C:\" incremental="yes">
  <tagWrite Value="51">DEVICE_USB_UTILITY_MTR\MMXU1\PPV\zavg</tagWrite>
  <tagWrite Value="51">DEVICE_GEN_EMCP_GEN\MMXU1\PPV\zavg</tagWrite>
  <tagWrite Value="1">DEVICE_SER_01\GGIO1\Ind2</tagWrite>
  <tagWrite Value="0">DEVICE_SER_01\GGIO1\Ind3</tagWrite>
  <tagWrite Value="1">DEVICE_SER_01\GGIO1\Ind5</tagWrite>
  <tagWrite Value="0">DEVICE_SER_01\GGIO1\Ind6</tagWrite>
  <tagWrite Value="1">DEVICE_SER_01\GGIO1\Ind8</tagWrite>
  <tagWrite Value="0">DEVICE_SER_01\GGIO1\Ind9</tagWrite>
  <tagWrite Value="1">DEVICE_USB_SER_01\GGIO1\Ind2</tagWrite>
  <tagWrite Value="0">DEVICE_USB_SER_01\GGIO1\Ind3</tagWrite>
  <tagWrite Value="1">DEVICE_USB_SER_01\GGIO1\Ind5</tagWrite>
  <tagWrite Value="0">DEVICE_USB_SER_01\GGIO1\Ind6</tagWrite>
  <tagWrite Value="1">DEVICE_USB_SER_01\GGIO1\Ind14</tagWrite>
  <tagWrite Value="0">DEVICE_USB_SER_01\GGIO1\Ind15</tagWrite>
  <tagWrite Value="1">DEVICE_USB_SER_01\GGIO1\Ind17</tagWrite>
  <tagWrite Value="0">DEVICE_USB_SER_01\GGIO1\Ind18</tagWrite>
  <tagWrite Value="1">DEVICE_USB_SER_01\GGIO1\Ind8</tagWrite>
```

Figura 14. Script desarrollado donde se observan las etiquetas de los equipos presentes en un unifilar, en su mayoría breakers.

Se observa la estructura de los scripts desarrollados, guardados en un archivo .xml, el cual hace que la lectura de la información se haga de una manera sencilla, este archivo posee en cada línea un comando que el interprete del SCADA identifica y sabe con que comando se activa el dispositivo, también indica con que valor lógico se activan dichos dispositivos, ya que estos pueden ser normalmente abiertos o cerrados y, por último, se indica el tag con el que está definido cada equipo.

Para la extracción de las variables eléctricas que se encuentran dentro de las ventanas emergentes (PopUps) de cada uno de los dispositivos, se hizo uso del siguiente código:

```python
def extract_popup(equipo,tag_file,value_voltage,value_current,value_power,value_PF):
    file= "..\\_database-files"+'\\'+equipo+"_variable.xml"

    #obtain data from xml files
    from xml.dom import minidom
    while True:
        try:
            doc = minidom.parse(file)
            break
        except :
            print("Cannot Find File")
            return False



    #extract tag name of variable.xml located in qubits network folder
    nombre = doc.getElementsByTagName("NAME")
    tipo= doc.getElementsByTagName("TYPE")
    tags=[]
    value_tags=[]
    for x in nombre:  #obtain tag name of variable
        tags.append(x.firstChild.data)

    for y in tipo:  #obtain tag type of variable
        value_tags.append(y.firstChild.data)


    #condition to create an text files for easy acces of tags
    if tag_file==True:
        f= open(equipo+".txt","w+")
```

```python
pageName = ET.Element("script",name="Pararell",saveAt="C:\\", incremental="yes")
for j in range(len(tags)):
    if "DIGITAL" in value_tags[j] :
        if "\\dchg" in tags[j]:
            ET.SubElement(pageName, "AlarmNotifyVarChange", Value="1").text= tags[j]
        else:
            ET.SubElement(pageName, "tagWrite", Value="1").text= tags[j]
    elif "\\A\\" in tags[j]:
        ET.SubElement(pageName, "tagWrite", Value=str(value_current)).text=  tags[j]
        value_current+=1
    elif "\\PPV\\" in tags[j] or "\\PhV\\" in tags[j]:
        ET.SubElement(pageName, "tagWrite", Value=str(value_voltage)).text= tags[j]
        value_voltage+=1
    elif "PF" in tags[j] :
        ET.SubElement(pageName, "tagWrite", Value=str(value_PF)).text= tags[j]
        value_voltage+=1
    else:
        ET.SubElement(pageName, "tagWrite", Value=str(value_power)).text= tags[j]
        #ET.SubElement(pageName, "sleepCall", Value="1", sleepName="Sleep")
        value_power+=1

#commands needed to indent linear xml and create .xml file with tags and values
tree = ET.ElementTree(pageName)
indent(pageName)
tree.write("POPUPTEST_"+equipo+".xml", encoding="utf-8", xml_declaration=True)
#print("Realizado con éxito")
time.sleep(1)
return True
```

Figura 15. Script desarrollado para la extracción de variables eléctricas dentro de un PopUp.

Cuando se ejecuta el anterior Script, aparece una GUI, que facilitara la asignación de valores de testeo para las variables que componen el PopUp.

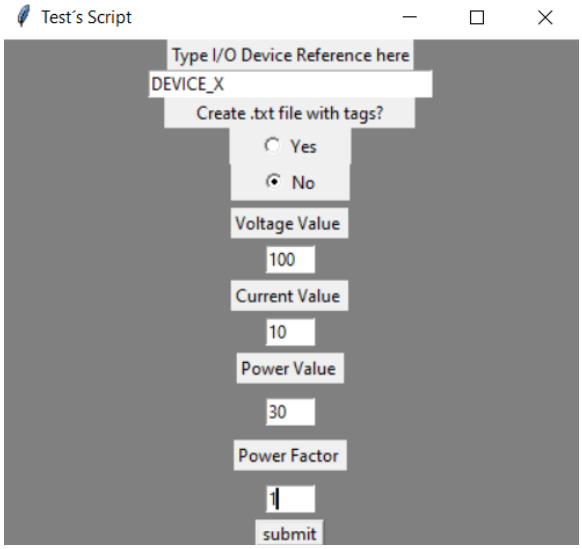

Figura 16. Interfaz gráfica para ingresar valores de variables eléctricas.

```
<?xml version='1.0' encoding='utf-8'?>
<script name="Pararell" saveAt="C:\" incremental="yes">
  <tagWrite Value="1">DEVICE_X\A5027_PIOC1\Op</tagWrite>
  <tagWrite Value="1">DEVICE_X\A51_PTOC1\Op</tagWrite>
  <tagWrite Value="1">DEVICE_X\A51_PTOC2\Op</tagWrite>
  <tagWrite Value="1">DEVICE_X\A51N_PTOC1\Op</tagWrite>
  <tagWrite Value="1">DEVICE_X\Dig\St\OvHz</tagWrite>
  <tagWrite Value="1">DEVICE_X\Dig\St\OvVolt</tagWrite>
  <tagWrite Value="1">DEVICE_X\Dig\St\RvPwr</tagWrite>
  <tagWrite Value="1">DEVICE_X\Dig\St\UnbVolt</tagWrite>
  <tagWrite Value="1">DEVICE_X\Dig\St\UnHz</tagWrite>
  <tagWrite Value="1">DEVICE_X\Dig\St\UnVolt</tagWrite>
  <tagWrite Value="30">DEVICE_X\MHAI1\ThdPPV\phsAB</tagWrite>
  <tagWrite Value="31">DEVICE_X\MHAI1\ThdPPV\phsBC</tagWrite>
  <tagWrite Value="32">DEVICE_X\MHAI1\ThdPPV\phsCA</tagWrite>
  <tagWrite Value="33">DEVICE_X\MHAI1\ZThdPhV\phsA</tagWrite>
  <tagWrite Value="34">DEVICE_X\MHAI1\ZThdPhV\phsB</tagWrite>
  <tagWrite Value="35">DEVICE_X\MHAI1\ZThdPhV\phsC</tagWrite>
  <tagWrite Value="36">DEVICE_X\MMTR1\TotVAh</tagWrite>
  <tagWrite Value="37">DEVICE_X\MMTR1\TotVArh</tagWrite>
  <tagWrite Value="38">DEVICE_X\MMTR1\TotWh</tagWrite>
  <tagWrite Value="10">DEVICE_X\MMXU1\A\neut</tagWrite>
  <tagWrite Value="11">DEVICE_X\MMXU1\A\neut\zloadPerc</tagWrite>
  <tagWrite Value="12">DEVICE_X\MMXU1\A\phsA</tagWrite>
  <tagWrite Value="13">DEVICE_X\MMXU1\A\phsA\zloadPerc</tagWrite>
  <tagWrite Value="14">DEVICE_X\MMXU1\A\phsB</tagWrite>
```

Figura 17. Script generado con etiquetas de variables eléctricas presentes en un PopUp

De la imagen anterior se puede observar que en las etiquetas de las variables exiten ciertas expresiones que permiten diferenciar cuando un tag correponde a una variable de voltaje, corriente, potencia, entre otras.

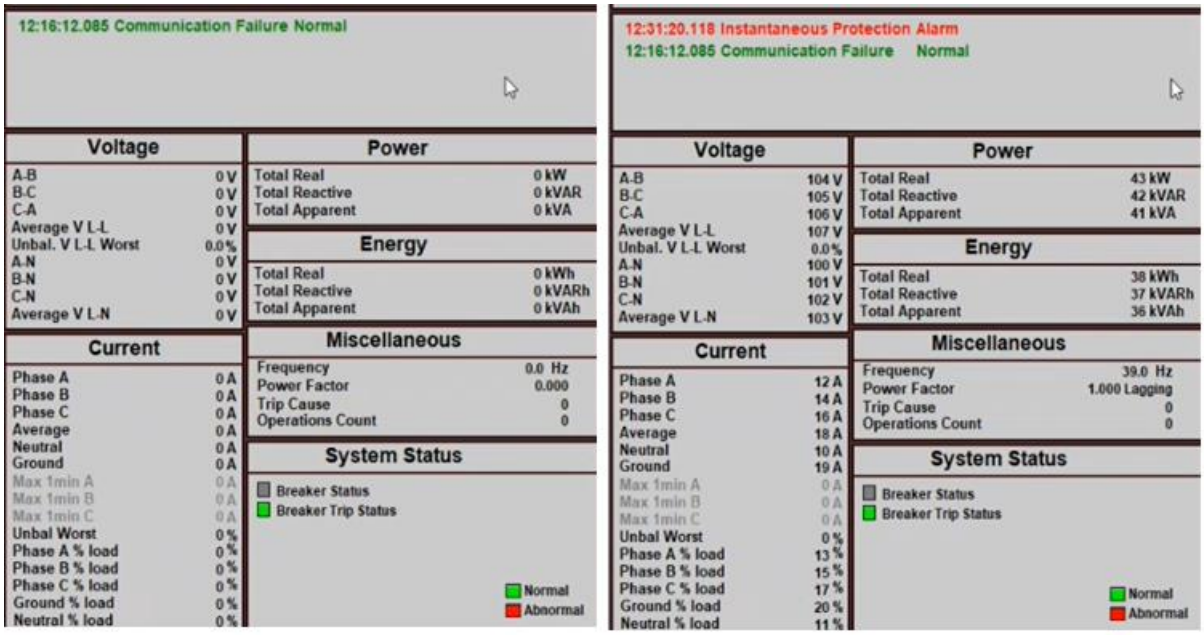

Figura 18. PopUp antes y después de la ejecución de script de prueba.

Como se pudo observar en la figura anterior, se realizó el mismo procedimiento que en la sección de testeo manual, a diferencia de que cuando se ejecuta esta tarea con los scripts automatizados, el testeo se hace mucho más rápido y eficiente, sin omitir ninguna variable, y solo teniendo que ejecutar un código de Python para generar dicho script usando el nombre del dispositivo que se requiera testear.

Para el testeo de las Alarmas del sistema, que van desde alarmas generales por fallas de comunicación, pruebas, entre otros, hasta alarmas específicas de cada zona o dispositivo indicando que un umbral definido de alguna variable fue sobrepasado, indicando un estado de alerta.

```
for line in dirs:
    if equipo in line:
        print(line)
        if ("trend" or "variable") in line:
            pass
        else:
            #obtain data from xml files
            if "digalm" in line:
                # Open a file
                doc = minidom.parse(file+'\\'+line)
                alm_dig = doc.getElementsByTagName("TAG")
                flag=1
            elif "anaalm" in line:
                doc = minidom.parse(file+'\\'+line)
                alm_analog = doc.getElementsByTagName("TAG")
                alm_analog_value_HIGH = doc.getElementsByTagName("HIGH")
                alm_analog_value_LOW = doc.getElementsByTagName("LOW")
                flag=1
            elif "advalm" in line:
                doc = minidom.parse(file+'\\'+line)
                alm_adv = doc.getElementsByTagName("TAG")
                flag=1
            elif "tsdig" in line:
                doc = minidom.parse(file+'\\'+line)
                alm_times_stamp = doc.getElementsByTagName("TAG")
```

```
if flag==0: #flag used to indicate if there were no files with the equipment name
    return False

tags=[]
value_tags=[]
tags.append("digital-------------")
for x in alm_dig:  #obtain tag name of variable
    tags.append(x.firstChild.data)
    value_tags.append('1')

#process to determine if value of analog alarm y high or LOW
tags.append("analoga-------------------------------------------------------------")
for i in range(len(alm_analog)):  #obtain tag name of variable
    tags.append(alm_analog[i].firstChild.data)
    if alm_analog_value_LOW[i].firstChild==None and alm_analog_value_HIGH[i].firstChild!=None:
        value_tags.append(str(round(float(alm_analog_value_HIGH[i].firstChild.data)+1)))
    elif alm_analog_value_HIGH[i].firstChild==None and alm_analog_value_LOW[i].firstChild!=None:
        value_tags.append(str(round(float(alm_analog_value_LOW[i].firstChild.data)-1)))
    else:
        value_tags.append('1')



tags.append("advanced------------------------------------------------------------")
for x in alm_adv:  #obtain tag name of variable
    tags.append(x.firstChild.data)
    value_tags.append('1')

tags.append("time stamp----------------------------------------------------------")
for x in alm_times_stamp:  #obtain tag name of variable
```

Figura 19. Script en Python para generar archivo con etiquetas de alarmas.

```python
pageName = ET.Element("script",name="Pararell",saveAt="C:\\", incremental="yes")
for j in range(len(tags)):
    if "--" in tags[j] :
        pass

    elif "\\dchg" in tags[j]:
        ET.SubElement(pageName, "AlarmNotifyVarChange", Value="1").text=tags[j]

    else:
        ET.SubElement(pageName, "tagWrite", Value=value_tags[j]).text=tags[j]



tree = ET.ElementTree(pageName)
indent(pageName)
tree.write("ALARMTEST_"+equipo+".xml", encoding="utf-8", xml_declaration=True)
#print("Realizado con éxito")
time.sleep(1)
return True
```

Figura 20. Script en Python para identificar tipos de alarmas.

De las figuras (19) y (20) se puede observar el script desarrollado para extraer todas las alarmas(entre ellas análogas, digitales, avanzadas y especiales) que pueden ser generadas por un dispositivo, este script tiene una importancia en particular, porque al momento de testear las alarmas, es una tarea más tediosa y a la vez es fundamental, porque si al momento de implementar el sistema las alarmas no se ejecutan cuando deberían, se pueden presentar problemas en el futuro que el sistema no detectara y/o notificara al cliente.

```xml
<?xml version='1.0' encoding='utf-8'?>
<script name="Pararell" saveAt="C:\" incremental="yes">
  <tagWrite Value="1">DEVICE_Y\A5027_PIOC1\Op</tagWrite>
  <tagWrite Value="1">DEVICE_Y\A51_PTOC1\Op</tagWrite>
  <tagWrite Value="1">DEVICE_Y\A51_PTOC2\Op</tagWrite>
  <tagWrite Value="1">DEVICE_Y\A51N_PTOC1\Op</tagWrite>
  <tagWrite Value="1">DEVICE_Y\Dig\St\OvHzAlm</tagWrite>
  <tagWrite Value="1">DEVICE_Y\Dig\St\OvVoltAlm</tagWrite>
  <tagWrite Value="1">DEVICE_Y\Dig\St\RvPwrAlm</tagWrite>
  <tagWrite Value="1">DEVICE_Y\Dig\St\UnbVoltAlm</tagWrite>
  <tagWrite Value="1">DEVICE_Y\Dig\St\UnHztAlm</tagWrite>
  <tagWrite Value="1">DEVICE_Y\Dig\St\UnVoltAlm</tagWrite>
  <tagWrite Value="1">DEVICE_Y\PTRC1\Op</tagWrite>
  <tagWrite Value="91">DEVICE_Y\MMXU1\A\phsA\zloadPerc</tagWrite>
  <tagWrite Value="91">DEVICE_Y\MMXU1\A\phsB\zloadPerc</tagWrite>
  <tagWrite Value="91">DEVICE_Y\MMXU1\A\phsC\zloadPerc</tagWrite>
  <tagWrite Value="1">DEVICE_Y\Dig\St\CurUnbal</tagWrite>
  <tagWrite Value="1">DEVICE_Y\LPHD1\EEHealth</tagWrite>
  <tagWrite Value="1">DEVICE_Y\LPHD1\SumHiPriAlmPresent</tagWrite>
  <tagWrite Value="1">DEVICE_Y\XCBR1\Pos</tagWrite>
</script>
```

Figura 21. Script generado con tags de alarmas a testear para un dispositivo

De la figura (21) se puede observar que para cada uno de los tags se asigna un valor en particular, estos valores son, según lo extraído del SCADA, los valores con los cuales las alarmas deberían activarse, dependiendo el dispositivo,

necesidades del cliente, y además si esta es una alarma análoga o digital, por lo tanto en la interfaz gráfica, ilustrada en la figura (22) se observa que solo se tiene como entrada el nombre del dispositivo que se desea testear.

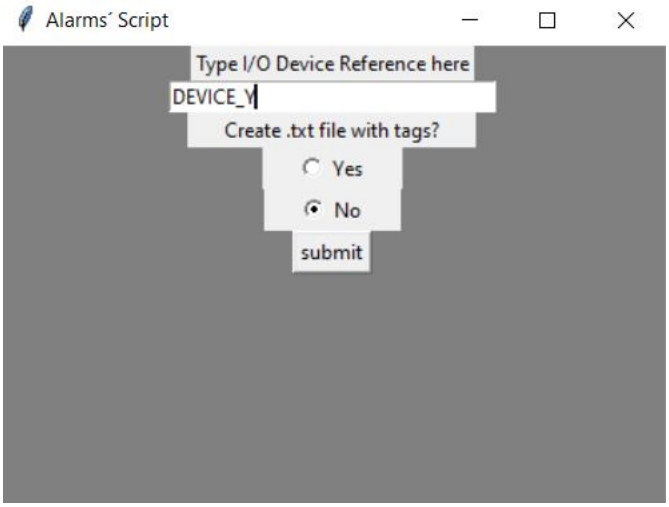

Figura 22. Interfaz gráfica para ejecución de scripts para alarmas.

La última modificación o funcionalidad que se agregó al sistema de automatización de pruebas fue un código con la misma funcionalidad de extraer información y tags de los scripts mostrados con anterioridad, con la diferencia de que para este desarrollo se usan parámetros, esto es, generar un script que permitiera testear un numero X de dispositivos que compartan las mismas etiquetas, y que facilitara aún más la revisión de los sistemas SCADA´s al testear muchos dispositivos al mismo tiempo.

```
#create the parameters tag, extract only the tag of the equipment and skip the I/O Devices
pageName = ET.Element("script",name="Pararell",saveAt="C:\\", incremental="yes")
parent =ET.SubElement(pageName, "Selected_Topics")
for j in range(len(tags)):
    line1=ET.SubElement(parent, "Selected_Topic")
    line1.set("Value","Insert Value Here") #message to indicate the developer to set the specific values of each
    if "\\dchg" in tags[j]:
        line1.set("TS","True")  #attributte to indicate the cicode, if the variable is a time stamp
    else:                       #which change the command to execute an specific tag
        line1.set("TS","False")
    line1.text=tags[j][tags[j].index('\\'):]

tree = ET.ElementTree(pageName)
indent(pageName)
tree.write("PARAMETER_TAGS__"+equipo+".xml", encoding="utf-8", xml_declaration=True)
print("Realizado con éxito")
time.sleep(1)
return True
```

Figura 23. Código para generación de scripts con parámetros.

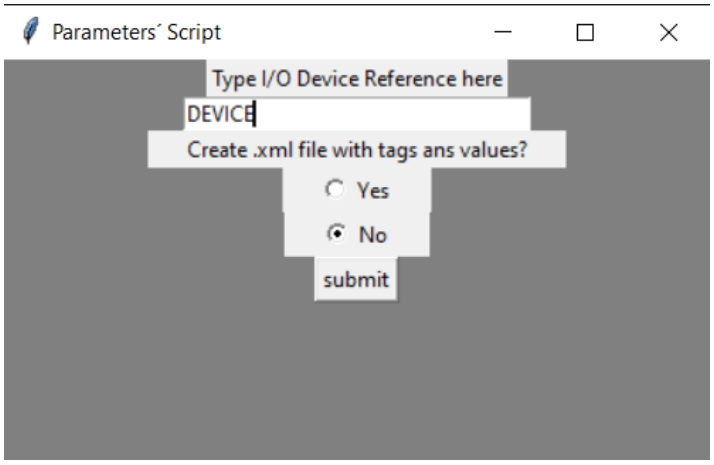

Figura 24. Interfaz para generación de scripts parametrizados

La diferencia de este Código o característica principal se basa en que al extraer las etiquetas de los dispositivos y/o variables a testear, no extrae la información junto con el dispositivo que se desea testear, sino que aparte del archivo XML generado, se crea manualmente otro archivo donde se definen los dispositivos que se desean testear, y que correspondan a la misma referencia y también tiene la función de que el desarrollador ingrese con que valores quiere testear los dispositivos, para probar valores de funcionamiento normal y de falla.

```
<?xml version='1.0' encoding='utf-8'?>
<script name="Pararell" saveAt="C:\" incremental="yes">
  <Selected_Topics>
    <Selected_Topic Value="Insert Value Here" TS="False">\A5027_PIOC1\Op</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\A51_PTOC1\Op</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\A51_PTOC2\Op</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\A51N_PTOC1\Op</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\Dig\St\OvHz</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\Dig\St\OvVolt</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\Dig\St\RvPwr</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\Dig\St\UnbVolt</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\Dig\St\UnHz</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\Dig\St\UnVolt</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MHAI1\ThdPPV\phsAB</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MHAI1\ThdPPV\phsBC</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MHAI1\ThdPPV\phsCA</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MHAI1\ZThdPhV\phsA</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MHAI1\ZThdPhV\phsB</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MHAI1\ZThdPhV\phsC</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MMTR1\TotVAh</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MMTR1\TotVArh</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MMTR1\TotWh</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MMXU1\A\neut</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MMXU1\A\neut\zloadPerc</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MMXU1\A\phsA</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MMXU1\A\phsA\zloadPerc</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MMXU1\A\phsB</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MMXU1\A\phsB\zloadPerc</Selected_Topic>
    <Selected_Topic Value="Insert Value Here" TS="False">\MMXU1\A\phsC</Selected_Topic>
```

Figura 25. Script generado con tags parametrizados.

PARAMETER_DEVICES_Templates.xml

```
<?xml version='1.0' encoding='utf-8'?>
<script name="Pararell" saveAt="C:\" incremental="yes">
  <Devices>
    <Device>BKR_MAIN</Device>
    <Device>BKR_GEN</Device>
    <Device>BKR_FEEDER</Device>
    <Device>BKR_1</Device>
    <Device>BKR_2</Device>
  </Devices>
</script>
```

Figura 26. Script generado con dispositivos a testear con parámetros.

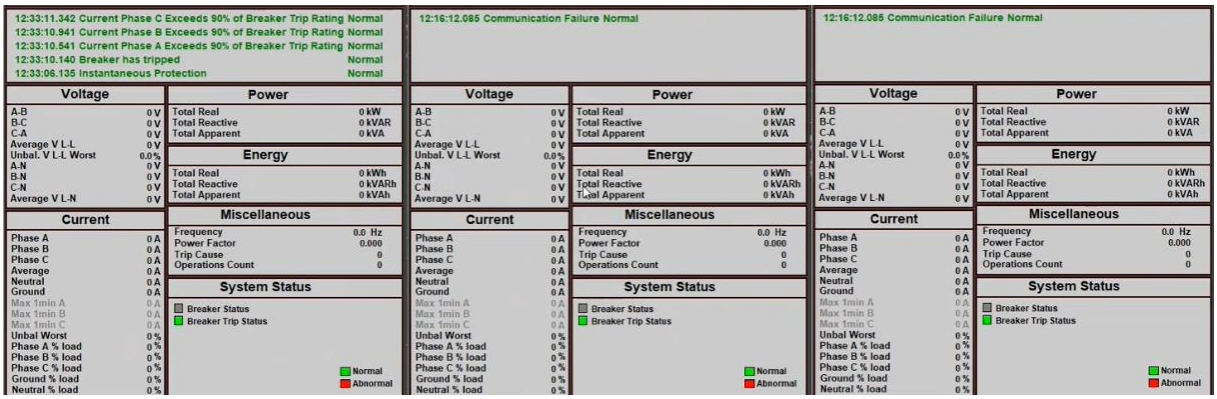

Figura 27. Ejemplo de varios PopUps a testear de la misma referencia.

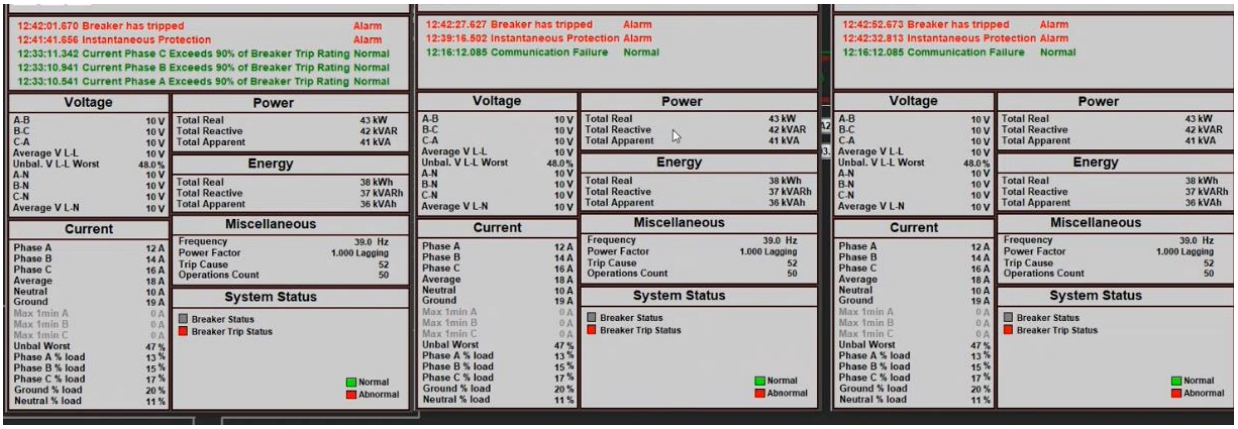

Figura 28. Ejemplo de varios PopUps testeados mediante parámetros.

Como se puede observar, este script es de gran utilidad al revisar una cantidad X de dispositivos en un instante de tiempo, y aunque en su mayoría se prueben los dispositivos individualmente, al momento de hacer una prueba con varios dispositivos, esto agiliza en gran medida dichas pruebas.

Se muestra a continuación y, por último, los scripts desarrollados en lenguaje C# y la interfaz realizada, todo en el back end del software del SCADA, para la ejecución de los scripts de automatización de pruebas.

En la figura (29) se observa la interfaz desarrollada, en donde se tiene varias funcionalidades, entre ellas está la ejecución de un solo script de automatización, el reseteo de dicho script para tomar las etiquetas y asignarles valor '0' a todas ellas, la ejecución de los scripts parametrizados, una función para tomar captura de pantalla de las zonas del sistema de monitoreo, para la realización de informes de prueba, y un cuadro de texto que indica la zona que está mostrando el SCADA, útil para saber con mayor facilidad el script correspondiente que debe ser ejecutado.

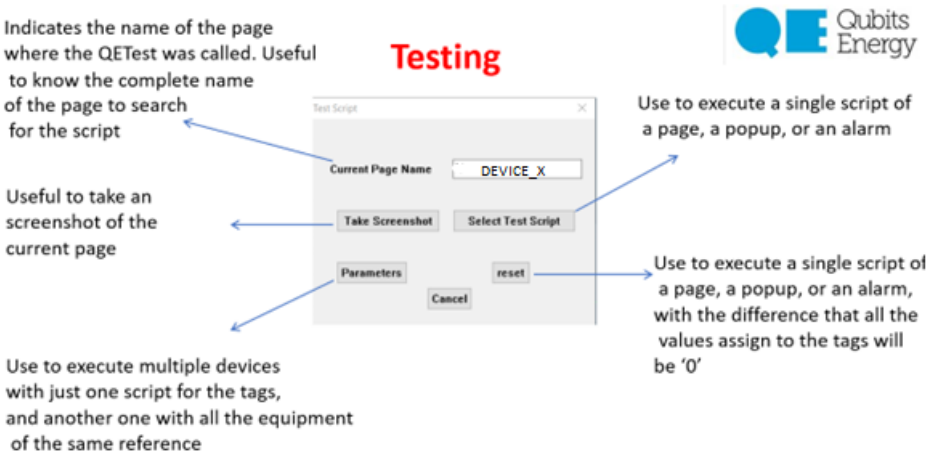

Figura 29. Interfaz de ejecución en software Power SCADA para ejecución de scripts de automatización

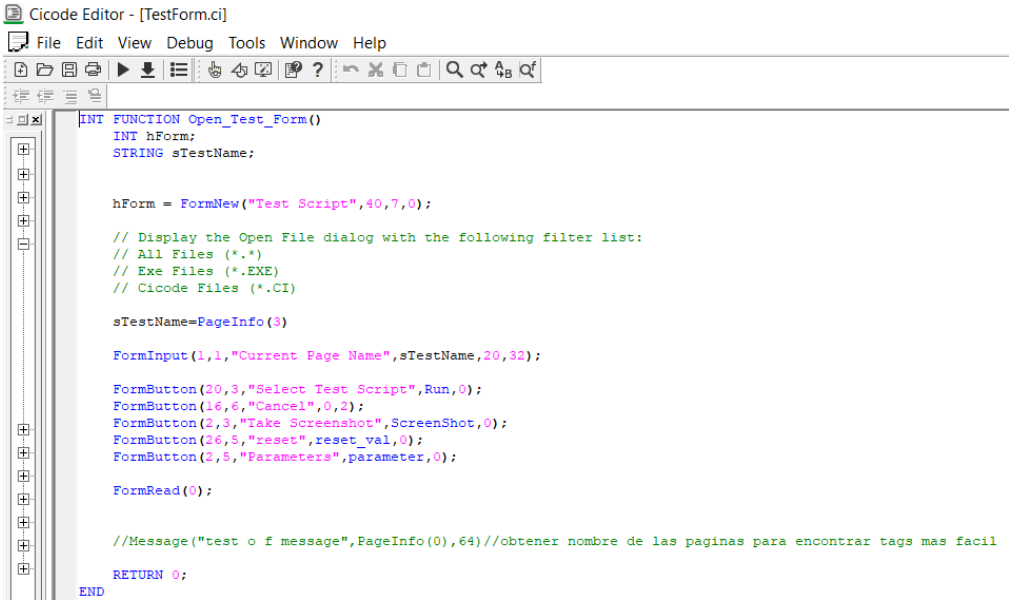

Figura 30. Código fuente de Power SCADA para construir interfaz gráfica de ejecución de scripts.

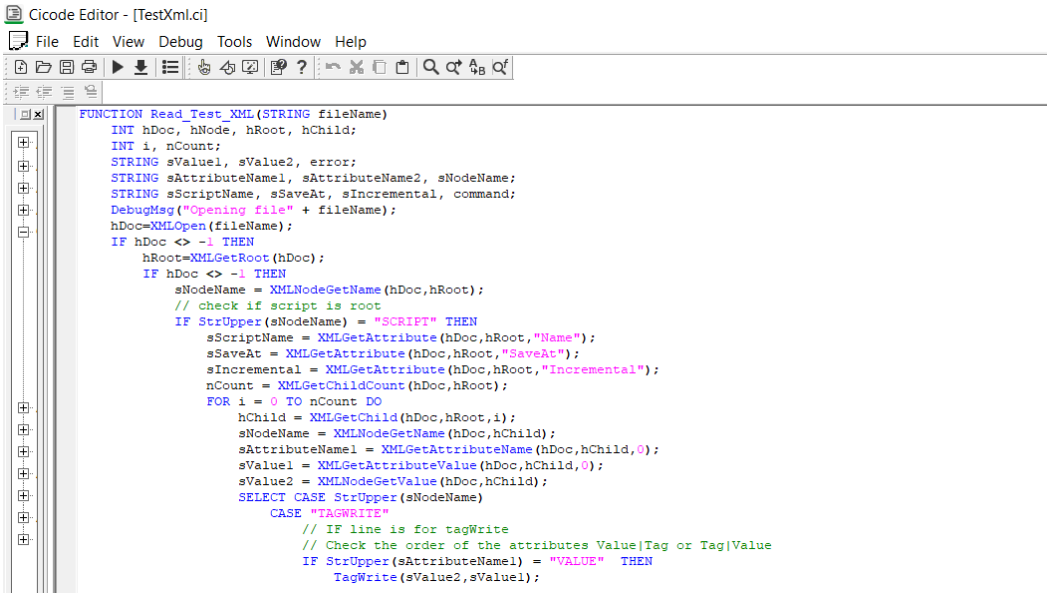

Figura 31. Código fuente de Power SCADA en C# para de ejecución de scripts según opción elegida en interfaz gráfica.

De las figuras (30) y (31) se puede observar una parte del Código desarrollado para la ejecución de los scripts de prueba, allí se puede observar en parte, la creación de la interfaz gráfica ilustrada en la figura (29), y el desglose de los archivos XML para extraer las etiquetas que se les asignara un valor dentro del SCADA.

Mejoras en la productividad

A continuación, se muestra una tabla comparativa, en la cual se pueden observar los tiempos que anteriormente tomaba revisar un proyecto promedio en la empresa, y como se mejoraron esos tiempos con la aplicación de scripts de automatización de pruebas.

Tabla #1. Estadísticas de mejora en las pruebas de funcionamiento de sistema SCADA.

Actividad	Antes de la aplicación de script	Después de la aplicación de scripts
revisión de paginas	30 horas	4 horas
revisión de PopUp y alarmas	20 horas	2 horas
Revisión de configuración de red y conexiones eléctricas	5 horas	½ hora
Otras revisiones que se deben hacer al sistema	10 horas	7 horas

De la tabla 1, se debe tener en cuenta que para un proyecto promedio de la empresa y en el desarrollo de un sistema SCADA, una subestación cuenta con 30 zonas, que se ilustran en el SCADA en páginas, para cada una de estas páginas, hay en promedio 15 dispositivos, de los cuales, cada uno tiene un PopUp, y donde estos ilustran y miden cada uno independientemente de los otros, en promedio, 20 variables eléctricas, donde según el equipo, pueden ser más o menos.

Conclusiones

- La implementación del sistema de automatización de pruebas era una tarea que se tenía pensada hace mucho por parte de la empresa, y efectivamente se comprobó que además de que reduce en gran cantidad el tiempo que se debe dedicar a las revisiones de un proyecto.
- Al ingeniero no tener que evaluar cada uno de los componentes que componen el sistema de automatización, sino solo tener que hacer la generación de los scripts y la ejecución de estos, se reduce el denominado "error humano", casi en su totalidad, ya que estos ayudan en gran medida a no cometer ningún error u omitir equipos y variables, ya que para ello se realizaron muchas pruebas en el desarrollo de estos scripts y se revisó minuciosamente que no se omitieran detalles importantes cuando se debe evaluar un sistema de monitoreo.
- La aplicación de los scripts y la automatización de ciertos procesos que antes se hacían manualmente en la empresa, dio apertura a la aplicación de estos y otras metodologías de automatización en distintos proyectos, permitiendo a la empresa ser mas eficientes no solo en uno, sino en varios campos en los que se desempeña. Además, dio partida para el uso de nuevas herramientas que en la empresa actualmente, se están aplicando para el desarrollo de nuevos productos y servicios, que serán ofrecidos a los clientes más adelante.
- En cuanto a las mejoras en productividad, se pudo concluir que se disminuyó aproximadamente en un 80%, el tiempo que toma hacer la revisión de un sistema de monitoreo, esto además de agilizar los procesos de prueba y despliegue de proyectos, permite a los ingenieros encargados del testeo y desarrollo, usar este tiempo para la ejecución de distintas tareas, y además el ahorro de tiempo en la revisión de los errores que se presentan las pruebas ya que con los scripts, se sabe exactamente donde se encuentra una falla.
- El que el trabajo de prácticas desarrollado involucrara en gran medida los conocimientos aprendidos a lo largo de mi carrera profesional, y que esta fuera un abrebocas a lo que me espera en el enfoque laboral elegido, fue una gran satisfacción personal y una buena preparación para mi futuro como profesional.

Referencias Bibliográficas

[1] Schneider Electric (2020), EcoStruxure™ Power Operation. Disponible en: https://www.se.com/ww/en/product-range/65405-ecostruxure-poweroperation/#overview

[2] Inductive Automation (2018) What is SCADA? Disponible en: https://inductiveautomation.com/resources/article/what-isscada#:~:text=Supervisory%20control%20and%20data%20acquisition%20(SCADA)%20is%20a%20system%20of,and%20process%20real%2Dtime%20data

[3] Cicode (2018) Cicode Programming Reference. Disponible en: https://johnwiltshire.com/citecthelp/Subsystems/CicodeReferenceCitectHTML/Content/Cicode_Programming_Reference.html

[4] Qubits Energy, Critical Power and Building Technologies (Estados Unidos). Disponible en: https://qubitsenergy.com/

[5] Rohde & Schwarz. Medición de la calidad de la energía en instalaciones eléctricas. Disponible en: https://www.rohde-schwarz.com/lat/aplicaciones/medici-n-de-la-calidad-de-la-energ-a-en-instalaciones-el-ctricas-ficha-de-aplicacion_56279-469952.html#:~:text=Las%20mediciones%20de%20la%20calidad,reactiva%20y%20desequilibrio%20de%20carga.

[6] Error humano. Disponible en: https://es.wikipedia.org/wiki/Error_humano

[7] PM800 Medidor de energía modular avanzado. Disponible en: https://www.se.com/mx/es/product-range/918-pm800/#overview

[8] Serie Power Logic PM5000. Medidores de energía multifunción para aplicaciones de asignación de costos, redes de monitoreo y seguimiento a insumos energéticos. Disponible en: https://www.se.com/mx/es/product-range/61281-serie-power-logic-pm5000/

[9] Introducción a XML.

Disponible en: https://developer.mozilla.org/es/docs/Web/XML/XML_introduction

[10] Electrical Distribution Equipment in Data Center Environments. Disponible en: https://download.schneider-electric.com/files?p_enDocType=White+Paper&p_File_Name=VAVR-8W4MEX_R2_EN.pdf&p_Doc_Ref=SPD_VAVR-8W4MEX_EN&_ga=2.5046003.1598406160.1657402189-698468707.1654283668