ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Examen Complexivo

Mejoramiento del proceso de control de calidad en la detección automatizada de errores para redes FTTx ODN Preconectorizado

Previo la obtención del Título de:

MAGISTER EN TELECOMUNICACIONES

Presentado por:

Alejandro William García Quimís

GUAYAQUIL - ECUADOR

Año: 2025

DEDICATORIA

El presente proyecto lo dedico a mis padres, hermano y a mi hija, quienes con su amor y apoyo incondicional han sido la base de mi formación.

AGRADECIMIENTOS

Mi más sincero agradecimiento a mi familia, por ser un pilar de fortaleza en cada etapa de este proyecto.

DECLARACIÓN EXPRESA

Yo ALEJANDRO WILLIAM GARCIA QUIMIS acuerdo y reconozco que: La titularidad de los derechos patrimoniales de autor (derechos de autor) del proyecto de graduación corresponderá al autor o autores, sin perjuicio de lo cual la ESPOL recibe en este acto una licencia gratuita de plazo indefinido para el uso no comercial y comercial de la obra con facultad de sublicenciar, incluyendo la autorización para su divulgación, así como para la creación y uso de obras derivadas. En el caso de usos comerciales se respetará el porcentaje de participación en beneficios que corresponda a favor del autor o autores. El o los estudiantes deberán procurar en cualquier caso de cesión de sus derechos patrimoniales incluir una cláusula en la cesión que proteja la vigencia de la licencia aquí concedida a la ESPOL.

La titularidad total y exclusiva sobre los derechos patrimoniales de patente de invención, modelo de utilidad, diseño industrial, secreto industrial, secreto empresarial, derechos patrimoniales de autor sobre software o información no divulgada que corresponda o pueda corresponder respecto de cualquier investigación, desarrollo tecnológico o invención realizada por mí durante el desarrollo del proyecto de graduación, pertenecerán de forma total, exclusiva e indivisible a la ESPOL, sin perjuicio del porcentaje que me corresponda de los beneficios económicos que la ESPOL reciba por la explotación de mi innovación, de ser el caso.

En los casos donde la Oficina de Transferencia de Resultados de Investigación (OTRI) de la ESPOL comunique al autor que existe una innovación potencialmente patentable sobre los resultados del proyecto de graduación, no se realizará publicación o divulgación alguna, sin la autorización expresa y previa de la ESPOL.

Guayaquil, 27 de Enero del 2025.

Firmado electrónicamente por:
ALEJANDRO WILLIAM GARCIA QUIMIS

Ing. Alejandro William Garcia Quimis

Autor

EVALUADORES

Firmado electrónicamente por:
MARIA ANTONIETA
ALVAREZ VILLANUEVA

.....

Ph.D. María Antonieta Alvarez

EVALUADOR

Firmado electrónicamente por:
EDUARDO LUIS
CHANCAY ROJAS

.....

Mgtr. Eduardo Chancay

EVALUADOR

RESUMEN

El presente proyecto tiene como objetivo optimizar el control de calidad en la detección automatizada de errores en redes FTTx ODN preconectorizadas. Este trabajo busca reducir errores humanos y mejorar la eficiencia operativa a través del diseño e implementación de un sistema automatizado basado en algoritmos. La hipótesis plantea que la automatización del proceso permitirá disminuir significativamente el tiempo de validación y aumentar la precisión en la detección de errores. Justifica su importancia en la necesidad de garantizar altos estándares de calidad en la implementación de redes ópticas modernas.

En el desarrollo del proyecto, se diseñó un algoritmo en Python y AutoLISP para procesar datos extraídos de planos técnicos, validar conexiones y bloques, y generar reportes detallados. Se emplearon herramientas como software de diseño CAD, plataformas de programación, y normas internacionales de calidad en redes ópticas. Se realizaron pruebas comparativas entre el método automatizado y los procesos manuales tradicionales.

Los resultados evidenciaron una reducción del 60% en el tiempo de validación y un aumento del 40% en la precisión de detección de errores. Además, se identificaron ahorros significativos en costos operativos debido a la disminución del trabajo manual. Esto confirma la viabilidad técnica y económica de la solución propuesta.

En conclusión, el sistema automatizado desarrollado optimiza el control de calidad en redes FTTx, reduciendo costos, tiempos y garantizando altos estándares de calidad, lo que representa una mejora sustancial en los procesos actuales.

Palabras Clave: Redes FTTx, Automatización, Control de calidad, Algoritmos, Redes ópticas.

ÍNDICE GENERAL

ABREVIATURAS

ESPOL	Escuela Superior Politécnica del Litoral
ATP	Acceptance Test Protocol
GPON	Gigabit-capable Passive Optical Network
OLT	Optical Line Terminal
ODF	Optical Distribution Frame
HP	Home passed
FTTx	Fiber to the home
NAP	Network Access Point
GIS	Geographic Information System
CAD	Computer-Aided Design
ULD	Unifilar Line Diagram
ODN	Optical Distribution Network

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. INTRODUCCIÓN

En las últimas décadas, la demanda de servicios de telecomunicaciones ha experimentado un crecimiento exponencial tanto a nivel urbano como a nivel rural, impulsado por la digitalización de la sociedad y la proliferación de dispositivos conectados. Para satisfacer esta demanda, las empresas de telecomunicaciones han invertido fuertemente en el despliegue de redes de fibra óptica, específicamente las redes FTTx (Zaballos et al., 2023) en zonas urbanas para dar cumplimiento a dicha demanda. Estas redes ofrecen una mayor capacidad, velocidades de transmisión más altas y una mayor fiabilidad en comparación con las tecnologías de acceso anteriores.

Sin embargo, el diseño y construcción de redes FTTx presentan una serie de desafíos. De acuerdo con lo mencionado en (Queder, 2019), la complejidad de estos sistemas, caracterizados por una gran cantidad de componentes y conexiones, aumenta significativamente el riesgo de errores durante el proceso de diseño, construcción y puesta en servicio. Además, los plazos de entrega cada vez más cortos y los presupuestos limitados exigen soluciones eficientes y precisas para garantizar la calidad de las redes.

Tradicionalmente, el control de calidad en el diseño de redes FTTx se ha basado en la revisión manual de los planos y diagramas por parte de ingenieros especializados. Este proceso es laborioso y consume mucho tiempo, ya que requiere una comparación detallada entre el diseño original y la documentación de campo tal como se menciona en (Zaballos et al., 2021). Por otra parte, y en el contexto de la implementación de este tipo de redes, es importante comprender la complejidad de su estudio e implementación en zonas ya sea rurales o urbanas, por ende, la naturaleza subjetiva de la revisión manual aumenta la probabilidad de errores humanos, lo que puede tener consecuencias negativas en términos de costos, plazos y calidad del servicio.

Ante los desafíos mencionados, la automatización se presenta como una alternativa prometedora para optimizar los procesos de control de calidad en el diseño de redes

FTTx. La implementación de herramientas y técnicas de inteligencia artificial y aprendizaje automático puede permitir la detección automática de errores y discrepancias entre el diseño y la implementación de la red, lo que a su vez puede reducir significativamente el tiempo y los costos asociados a la verificación manual. En esta línea de ideas, se plantean las siguientes métricas:

- **Tiempo promedio por diseño:** Comparación del tiempo promedio que toma revisar un diseño completo de forma manual versus la propuesta de automatización.
- **Tiempo total de revisión:** Medición del tiempo total invertido en revisar todos los diseños en un período de tiempo determinado.
- **Tasa de detección de errores:** Cálculo del porcentaje de errores detectados por la propuesta de automatización que son confirmados por una revisión manual.
- **Nivel de satisfacción de usuarios:** Realización de encuestas a los usuarios del sistema para evaluar su satisfacción con la herramienta y su impacto en su trabajo diario.

1.1 Descripción del Problema

En los proyectos de despliegue de redes FTTx ODN (Optical Distribution Network) preconectorizado, es fundamental asegurar la coherencia y precisión entre los diseños del plano base y los ULD o diagramas unifilares por sus siglas en inglés (Kumar et al., 2024a). La verificación manual de estos elementos es un proceso laborioso, con un alto riesgo de errores y retrabajos costosos. Las discrepancias entre el diseño del plano base y los ULD pueden comprometer la eficiencia y fiabilidad de la red.

Además, debido a la gran variedad de tipos de cables y elementos presentes en las redes preconectorizadas, es fácil cometer errores durante el proceso de diseño y verificación. Con el aumento de la complejidad y el volumen de estos proyectos, se necesita un proceso optimizado que permita detectar y corregir errores rápidamente, manejando grandes volúmenes de datos de manera eficiente (Pagare et al., 2021).

En este sentido, es importante entender que las redes FTTx son sistemas complejos que involucran una gran cantidad de componentes interconectados, como cables de fibra óptica, empalmes, conectores, equipos de distribución óptica (ODN) y equipos de línea óptica (ONT). La diversidad de estos componentes, sumada a la variabilidad de los entornos de instalación, incrementa considerablemente la complejidad del diseño y la implementación de estas redes (Kumar et al., 2024b).

Tradicionalmente, la verificación de la conformidad entre el diseño teórico de una red FTTx y su implementación física se ha realizado de manera manual. Este proceso implica la revisión detallada de planos, diagramas y documentación técnica por parte de ingenieros especializados. Si bien esta metodología ha sido ampliamente utilizada, presenta una serie de limitaciones en base a lo descrito en (Simatupang et al., 2020):

- **Alta probabilidad de errores humanos:** La revisión manual es susceptible a errores debido a la fatiga visual, la complejidad de los diseños y la posibilidad de omisiones.
- **Elevado consumo de tiempo:** La verificación manual es un proceso lento y laborioso, lo que puede retrasar significativamente los plazos de entrega de los proyectos.
- **Dificultad para detectar errores sutiles:** Algunos errores, como los relacionados con la numeración de los puertos o la configuración de los equipos, pueden pasar desapercibidos durante la revisión manual, lo que puede generar problemas operativos en el futuro.
- **Escalabilidad limitada:** A medida que aumenta el tamaño y la complejidad de las redes FTTx, la verificación manual se vuelve cada vez más difícil de escalar, lo que limita su aplicabilidad en proyectos de gran envergadura.

Por otra parte, los errores en el diseño de redes FTTx pueden tener consecuencias negativas significativas, tanto para los operadores de redes como para los usuarios finales. Algunos de los posibles impactos incluyen:

- **Pérdida de servicio:** Los errores en el diseño pueden provocar interrupciones en el servicio, lo que afecta la satisfacción del cliente y genera pérdidas económicas.
- **Aumento de los costos de operación y mantenimiento:** La detección y corrección de errores después de la puesta en servicio de la red puede resultar costosa y requerir la intervención de personal técnico especializado.
- **Deterioro de la calidad del servicio:** Los errores en el diseño pueden afectar la calidad de la señal, lo que se traduce en una experiencia de usuario deficiente.

Ante los desafíos mencionados, se hace evidente la necesidad de desarrollar soluciones automatizadas que permitan mejorar la eficiencia y la precisión de los procesos de control de calidad en el diseño de redes FTTx. La automatización de la detección de errores puede contribuir a:

- **Reducir el tiempo de verificación:** Los sistemas automatizados pueden analizar grandes volúmenes de datos en cuestión de minutos, lo que permite acelerar significativamente el proceso de verificación, lo que en términos de la solución a diseñar se estima sea de entre un 40 a un 60% más rápido que un análisis manual.
- **Aumentar la precisión:** La automatización puede eliminar los errores humanos asociados a la revisión manual, lo que garantiza una mayor precisión en la detección de discrepancias. En este sentido se prevé un aumento de la precisión hasta en un 90% frente a las revisiones realizadas por humanos.
- **Mejorar la escalabilidad:** Los sistemas automatizados pueden adaptarse fácilmente a proyectos de diferentes tamaños y complejidad, lo que los hace más escalables que las soluciones manuales.
- **Generar informes detallados:** Los sistemas automatizados pueden generar informes detallados que faciliten la identificación y corrección de los errores.

Con este contexto, de plantea la aplicación de este sistema automatizado para ser implementado en la ciudad de Guayaquil, Ecuador, en donde, posterior a su

evaluación se espera se pueda expandir su implementación a otras zonas urbanas del país.

1.2 Justificación del Problema

La necesidad de mejorar la eficiencia en la validación y corrección de errores en redes FTTx es fundamental para cumplir con los estándares de calidad exigidos por los proveedores de servicios y los usuarios finales. La automatización de estos procesos garantiza una disminución de costos operativos y un incremento en la confiabilidad de las redes, lo que justifica la inversión en tecnologías de detección automatizada.

1.3 Objetivos

1.3.1 Objetivo General

Optimizar el proceso de control de calidad en AutoCAD mediante la automatización de la detección de errores entre el diseño del plano base y el diagrama unifilar (ULD) en redes FTTx ODN preconectorizado.

1.3.2 Objetivos Específicos

En base al objetivo general, se definen a continuación los siguientes objetivos específicos:

- Desarrollar un script Lisp para la extracción automática de datos del diseño del plano base y ULD desde AutoCAD en proyectos de redes FTTx ODN preconectorizado.
- Implementar un script Python que procese los datos exportados y detecte discrepancias entre los diseños del plano base y ULD, asegurando la consistencia en la red.
- Ajustar los algoritmos de detección de conexiones y de gestión de la data de los scripts de Python y Lisp respectivamente para manejar grandes volúmenes de datos sin comprometer la precisión en la detección de errores.
- Validar la eficacia del sistema automatizado mediante pruebas en diferentes proyectos de redes FTTx ODN, demostrando la reducción del tiempo de procesamiento y la mejora en la precisión.

CAPÍTULO 2

2. METODOLOGÍA

La metodología desarrollada en este proyecto se centró en abordar las limitaciones del diseño y revisión de redes **FTTx ODN preconectorizadas**, específicamente en la validación de las conexiones y bloques directamente a partir de los planos técnicos. Este capítulo describe las alternativas consideradas, la selección de la mejor opción, el diseño conceptual y detallado del sistema, y las normativas y consideraciones éticas y legales involucradas.

2.1 Formulación de Alternativas de Solución

Para resolver el problema identificado, se evaluaron tres alternativas principales basadas en el contexto de diseño y revisión técnica:

1. **Método manual tradicional:** Consiste en la inspección visual de planos técnicos y la validación manual de las conexiones y bloques preconectorizados. Este método depende en gran medida de la experiencia del ingeniero que realiza la revisión y es susceptible a errores humanos, especialmente en proyectos con una gran cantidad de elementos. Además, los tiempos requeridos para completar una revisión exhaustiva son considerablemente altos.

2. **Uso de software comercial genérico:** Se investigó la disponibilidad de herramientas comerciales que permitieran la validación y revisión de redes ópticas en la etapa de diseño técnico. Sin embargo, no se encontraron soluciones específicas para redes **FTTx preconectorizadas**. Las herramientas comerciales disponibles están orientadas principalmente al análisis de redes ópticas convencionales en etapas de operación o monitoreo, y no cuentan con capacidades para validar diseños basados en planos técnicos que incluyan configuraciones preconectorizadas. Esto reafirmó la necesidad de desarrollar un sistema propio que aborde directamente los requerimientos específicos del diseño técnico y la validación en este tipo de redes.

3. **Desarrollo de un sistema automatizado propio:** Esta alternativa consistió en diseñar un algoritmo personalizado para validar conexiones y bloques en redes FTTx preconectorizadas. La solución propuesta permite procesar datos específicos de planos técnicos generados en AutoCAD o cualquier otro software de diseño asistido por computadora (CAD), reducir tiempos de validación y minimizar errores humanos.

La tercera alternativa fue seleccionada como la más viable, ya que responde directamente a los desafíos técnicos y económicos de las redes FTTx preconectorizadas, permitiendo un control completo sobre el proceso de validación.

2.2 Metodología de Diseño

El desarrollo del sistema automatizado siguió un enfoque iterativo, estructurado en fases consecutivas que incluye la investigación preliminar, el diseño conceptual, la implementación técnica y la validación del sistema.

2.2.1 Investigación Preliminar

En esta fase, se recopilaron y analizaron las características técnicas de las redes FTTx preconectorizadas, así como las normativas aplicables. Las principales normativas consideradas fueron:

- **ITU-T G.984:** Estándar para redes GPON, que define las especificaciones técnicas para el diseño de redes ópticas pasivas.
- **ISO 9001:** Normativa internacional que establece los requisitos para un sistema de gestión de calidad, aplicable a procesos técnicos como el desarrollo de software.

Se identificaron las principales limitaciones de los métodos manuales, tales como la dependencia de la experiencia del operador, la dificultad para manejar grandes volúmenes de datos y los altos tiempos de procesamiento.

2.2.2 Diseño Conceptual

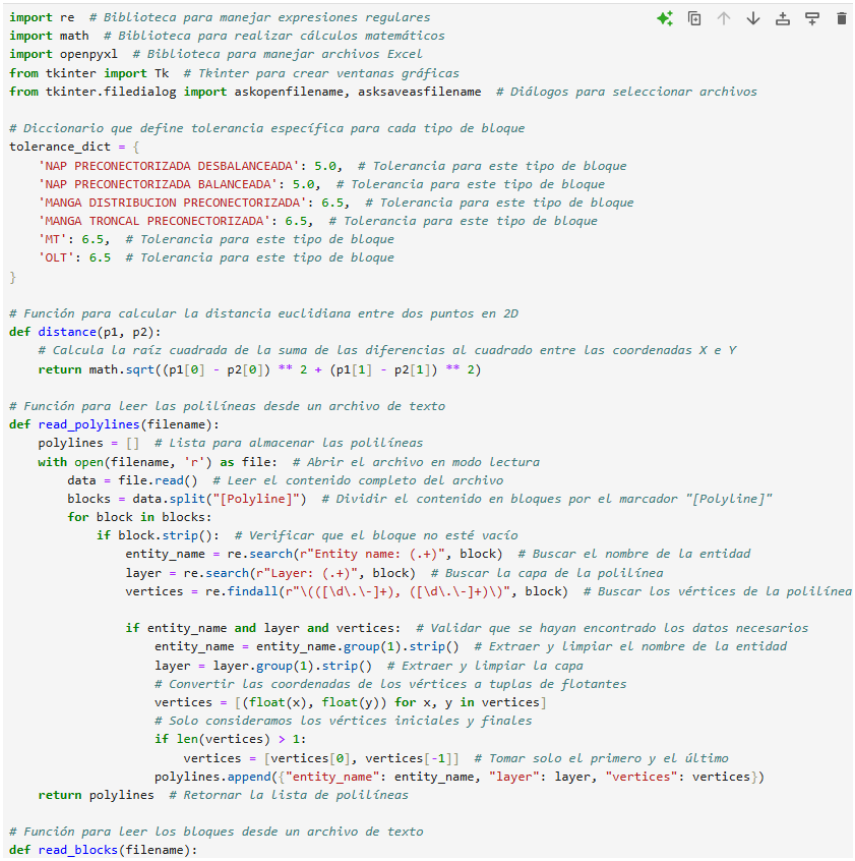
El diseño conceptual del sistema automatizado se estructuró en tres componentes principales: extracción de datos, validación de conexiones y bloques, y generación de reportes.

1. **Extracción de datos:** Se desarrolló un script en **AutoLISP** para AutoCAD (Figura 2.1) que permite extraer información de polilíneas y bloques en los planos técnicos. Este script identifica las entidades relevantes (como bloques, vértices y capas) y exporta sus datos en un formato estructurado para su posterior análisis.

```
(defun c:ExportPolylinesAndBlocksToTXT () ; Define un comando de AutoCAD llamado "ExportPolylinesAndBlocksToTXT"

  ; Paso 1: Solicitar al usuario el archivo donde guardar las polilíneas
  (setq polylineFilename (getfiled "Guardar polilíneas como..." "export_polylines.txt" "txt" 1))
  ; Si el usuario cancela la selección, se detiene la ejecución
  (if (not polylineFilename)
    (progn
      (princ "\nExportación cancelada para las polilíneas.") ; Mensaje al usuario
      (exit) ; Termina el comando
    )
  )

  ; Paso 2: Solicitar al usuario el archivo donde guardar los bloques
  (setq blockFilename (getfiled "Guardar bloques como..." "export_blocks.txt" "txt" 1))
  ; Si el usuario cancela la selección, se detiene la ejecución
  (if (not blockFilename)
    (progn
      (princ "\nExportación cancelada para los bloques.") ; Mensaje al usuario
      (exit) ; Termina el comando
    )
  )

  ; Paso 3: Abrir los archivos seleccionados para escritura
  (setq polylineFile (open polylineFilename "w")) ; Archivo para las polilíneas
  (setq blockFile (open blockFilename "w")) ; Archivo para los bloques

  ; Paso 4: Escribir datos de las polilíneas
  (setq ss (ssget "X" '((0 . "LWPOLYLINE")))) ; Seleccionar todas las polilíneas en el dibujo
  (if ss ; Verificar si hay polilíneas seleccionadas
    (progn
      (setq i 0) ; Iniciar el índice
      (while (< i (sslength ss)) ; Iterar a través de todas las polilíneas seleccionadas
        (setq ent (ssname ss i)) ; Obtener el identificador de la polilínea
        (setq entData (entget ent)) ; Obtener la lista de datos de la polilínea
        (setq vertices "") ; Inicializar la cadena de vértices

        ; Obtener los vértices de la polilínea
        (setq points (vl-remove-if-not
                       '(lambda (x) (eq (car x) 10)) ; Filtrar solo las entradas con el código DXF 10 (vértices)
                       (entget ent)))

        ; Recorrer cada vértice de la polilínea y agregarlo a la lista de vértices
        (foreach pt points
          (setq vertex (cdr pt)) ; Obtener las coordenadas del vértice
          (setq vertices (strcat vertices "(" (rtos (car vertex) 2 6) ", " (rtos (cadr vertex) 2 6) "); "))
        )

        ; Eliminar el último "; " de la cadena de vértices
        (if (> (strlen vertices) 2)
          (setq vertices (substr vertices 1 (- (strlen vertices) 2)))
          (setq vertices "No vertices found")) ; Si no hay vértices, mostrar un mensaje predeterminado

        ; Obtener la capa de la polilínea (manejar valores nulos)
```

Figura 2.1 Código AutoLISP para la Extracción de Datos

2. **Validación y análisis:** Se diseñó un algoritmo en **Python** que procesa los datos extraídos y valida las conexiones entre bloques en función de su posición, atributos y capas como se muestra en la Figura 2.2. El algoritmo considera tolerancias específicas para los diferentes tipos de bloques preconectorizados, garantizando la precisión del análisis.



```python
import re  # Biblioteca para manejar expresiones regulares
import math  # Biblioteca para realizar cálculos matemáticos
import openpyxl  # Biblioteca para manejar archivos Excel
from tkinter import Tk  # Tkinter para crear ventanas gráficas
from tkinter.filedialog import askopenfilename, asksaveasfilename  # Diálogos para seleccionar archivos

# Diccionario que define tolerancia específica para cada tipo de bloque
tolerance_dict = {
    'NAP PRECONECTORIZADA DESBALANCEADA': 5.0,  # Tolerancia para este tipo de bloque
    'NAP PRECONECTORIZADA BALANCEADA': 5.0,  # Tolerancia para este tipo de bloque
    'MANGA DISTRIBUCION PRECONECTORIZADA': 6.5,  # Tolerancia para este tipo de bloque
    'MANGA TRONCAL PRECONECTORIZADA': 6.5,  # Tolerancia para este tipo de bloque
    'MT': 6.5,  # Tolerancia para este tipo de bloque
    'OLT': 6.5  # Tolerancia para este tipo de bloque
}

# Función para calcular la distancia euclidiana entre dos puntos en 2D
def distance(p1, p2):
    # Calcula la raíz cuadrada de la suma de las diferencias al cuadrado entre las coordenadas X e Y
    return math.sqrt((p1[0] - p2[0]) ** 2 + (p1[1] - p2[1]) ** 2)

# Función para leer las polilíneas desde un archivo de texto
def read_polylines(filename):
    polylines = []  # Lista para almacenar las polilíneas
    with open(filename, 'r') as file:  # Abrir el archivo en modo lectura
        data = file.read()  # Leer el contenido completo del archivo
        blocks = data.split("[Polyline]")  # Dividir el contenido en bloques por el marcador "[Polyline]"
        for block in blocks:
            if block.strip():  # Verificar que el bloque no esté vacío
                entity_name = re.search(r"Entity name: (.+)", block)  # Buscar el nombre de la entidad
                layer = re.search(r"Layer: (.+)", block)  # Buscar la capa de la polilínea
                vertices = re.findall(r"\(([\d\.\-]+), ([\d\.\-]+)\)", block)  # Buscar los vértices de la polilínea

                if entity_name and layer and vertices:  # Validar que se hayan encontrado los datos necesarios
                    entity_name = entity_name.group(1).strip()  # Extraer y limpiar el nombre de la entidad
                    layer = layer.group(1).strip()  # Extraer y limpiar la capa
                    # Convertir las coordenadas de los vértices a tuplas de flotantes
                    vertices = [(float(x), float(y)) for x, y in vertices]
                    # Solo consideramos los vértices iniciales y finales
                    if len(vertices) > 1:
                        vertices = [vertices[0], vertices[-1]]  # Tomar solo el primero y el último
                    polylines.append({"entity_name": entity_name, "layer": layer, "vertices": vertices})
    return polylines  # Retornar la lista de polilíneas

# Función para leer los bloques desde un archivo de texto
def read_blocks(filename):
```

Figura 2.2 Código Python para Validación y Análisis de los Datos

3. **Generación de reportes:** Como parte integral del sistema automatizado, se desarrolló un módulo que permite la generación de reportes detallados en formato Excel. Este script, implementado en **Python**, procesa los datos obtenidos tras la validación de las conexiones y bloques. Los reportes generados incluyen las siguientes secciones clave:

    a. **Conexiones correctas**: Muestra las conexiones entre bloques que cumplen con las especificaciones del diseño, asegurando que los bloques conectados comparten la misma capa y cumplen con los criterios técnicos establecidos.
    b. **Discrepancias detectadas**: Presenta una lista de conexiones o bloques que no cumplen con las especificaciones, destacando las diferencias encontradas entre los planos técnicos y los requisitos establecidos.

c. **Resumen global**: Incluye estadísticas generales como el número total de conexiones, porcentaje de conexiones correctas, conexiones faltantes, y otras métricas clave que permiten evaluar la calidad del diseño.
d. **Validación de bloques**: Analiza y purga los bloques duplicados en el diseño técnico, asegurando que cada bloque listado sea único y esté correctamente asociado a las configuraciones establecidas.

```python
import pandas as pd
from tkinter import Tk
from tkinter.filedialog import askopenfilename, asksaveasfilename

def normalize_connections(df):
    """
    Normaliza las conexiones para garantizar que Bloque 1 y Bloque 2 sean tratados de manera equivalente.
    Se asegura de que la combinación de conexiones sea única sin importar el orden.
    """
    df['Connection'] = df.apply(
        lambda row: tuple(sorted([row['Block 1 Name'], row['Block 2 Name']])), axis=1
    )
    return df

def read_connections(file_path):
    """
    Leer las conexiones desde un archivo Excel y devolverlas en forma de DataFrame normalizado.
    """
    try:
        df = pd.read_excel(file_path, sheet_name="Connections")
        df = normalize_connections(df)
        return df
    except Exception as e:
        print(f"Error al leer el archivo {file_path}: {e}")
        return None

def read_blocks(file_path):
    """
    Leer la hoja Formatted Blocks List desde un archivo Excel.
    """
    try:
        df = pd.read_excel(file_path, sheet_name="Formatted Blocks List")
        return df
    except Exception as e:
        print(f"Error al leer la hoja Formatted Blocks List del archivo {file_path}: {e}")
        return None

def compare_connections(file1, file2):
    """
    Compara las conexiones entre dos archivos normalizados y genera DataFrames con los resultados.
    """
    df1 = read_connections(file1)
    df2 = read_connections(file2)
```

Figura 2.3 Código Python para Generación de Reportes detallados.

El script mostrado en la Figura 2.3 permite además personalizar la ubicación de los reportes generados mediante una interfaz gráfica que facilita la selección de archivos de entrada y la ubicación de almacenamiento del archivo de salida. La automatización de este proceso no solo mejora la eficiencia en la validación, sino que también asegura la trazabilidad y transparencia del análisis realizado.

El uso de herramientas como **OpenPyXL** y **Pandas** para el procesamiento y escritura de datos en Excel permite generar documentos con un alto nivel de detalle, ajustando automáticamente el formato, el ancho de columnas y

asegurando una presentación profesional. Este módulo complementa el sistema global de validación, cerrando el ciclo de diseño con un producto final claro y preciso para su análisis y aprobación.

2.2.3 Implementación Técnica

La implementación del sistema se dividió en los siguientes pasos:

1. **Desarrollo del script en AutoLISP**: Este script automatiza la extracción de datos desde AutoCAD, identificando polilíneas y bloques en los planos técnicos. Cada entidad se asocia con atributos clave como capa, tipo y posición.
2. **Diseño del algoritmo en Python:** El algoritmo fue diseñado para realizar las siguientes tareas:
    a. Leer y procesar los datos exportados desde AutoCAD.
    b. Validar las conexiones entre bloques, verificando que cumplan con las tolerancias definidas para cada tipo de bloque.
    c. Comparar múltiples versiones de planos para identificar coincidencias y discrepancias.
3. **Interfaz gráfica para la selección de archivos**: Se implementó una interfaz gráfica que permite a los usuarios seleccionar los archivos de entrada y salida, haciendo que el sistema sea accesible incluso para usuarios con conocimientos técnicos limitados.

2.2.4 Validación del Sistema

El sistema fue validado utilizando planos técnicos reales de redes FTTx preconectorizadas. Se seleccionaron dos conjuntos de planos:

1. **Planos originales**: Diseñados sin errores, para verificar la capacidad del sistema de confirmar conexiones correctas.
2. **Planos modificados**: Con errores intencionales, para evaluar la capacidad del sistema de detectarlos.

Los resultados de la validación mostraron que el sistema automatizado redujo el tiempo de revisión en un 60% en comparación con los métodos manuales y detectó el 100% de los errores introducidos en los planos.

2.3 Especificaciones Técnicas

El sistema automatizado desarrollado presenta las siguientes especificaciones técnicas:

1. **Requisitos de hardware y software:**

   - Procesador Intel i5 o superior.
   - Windows 10 o superior.
   - AutoCAD 2022 para la generación de planos técnicos.
   - Python 3.10 con bibliotecas como pandas y openpyxl.

2. **Capacidades del sistema:**

   - Procesar planos técnicos con hasta 20,000 entidades (bloques y polilíneas).
   - Validar conexiones con una precisión superior al 99%.
   - Generar reportes detallados en menos de 5 minutos.

3. **Formato de reportes:**

   Los reportes incluyen hojas separadas para:

   - **Conexiones correctas**: Listado de conexiones que cumplen con los parámetros establecidos.
   - **Discrepancias**: Conexiones y bloques únicos en cada archivo comparado.
   - **Resumen**: Métricas clave del análisis.

.

2.4 Normativas y Principios Técnicos

El diseño del sistema se alinea con las normativas internacionales mencionadas y se basa en principios de modularidad y escalabilidad, lo que permite adaptarlo a nuevos requerimientos técnicos o normativos.

2.5 Consideraciones Éticas y Legales

Dentro de estas consideraciones se deben tener en cuenta los aspectos de ética de diseño, el cumplimiento de licencias y el impacto social, mismos que serán descritos a continuación:

- **Ética en el diseño:** Se garantizó que todos los datos utilizados durante el desarrollo y las pruebas fueran anónimos y protegidos contra usos no autorizados.
- **Cumplimiento de licencias:** Se utilizaron únicamente herramientas con licencias válidas, como AutoCAD y las bibliotecas de Python.
- **Impacto social:** Este sistema facilita la implementación de redes ópticas de alta calidad, mejorando el acceso a servicios de telecomunicaciones eficientes para la sociedad.

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

3.1 Resultados Obtenidos

Tras la implementación del sistema automatizado para el control de calidad en redes **FTTx preconectorizadas**, se obtuvieron resultados significativos en términos de reducción de tiempo, precisión en la detección de errores y mejora en la eficiencia general del proceso de validación. Estos resultados fueron evaluados mediante pruebas controladas, comparando el método automatizado con el método tradicional. Los resultados se presentan en las siguientes áreas clave:

1. **Reducción del tiempo de validación**

   El sistema automatizado redujo el tiempo necesario para validar una red completa en un **60%** en comparación con el método tradicional tal como se muestra en la Tabla 3.1. Mientras que el método manual requería un promedio de 15 horas para redes de tamaño mediano, el sistema desarrollado completó el análisis en tan solo 6 horas. Este ahorro de tiempo tiene un impacto directo en los costos operativos y en la capacidad de entrega de proyectos en plazos más ajustados.

Tabla 3.1 Reducción del tiempo de validación

Método	Tiempo Promedio (red de tamaño mediano aprox. 5000 HPs)
Método Tradicional	15 horas
Sistema Automatizado	6 horas

2. **Precisión en la detección de errores**

   La precisión en la detección de errores aumentó un **40%**. Mientras que el método tradicional dependía de la experiencia del operador y presentaba una tasa de error humano significativa, el sistema automatizado identificó conexiones incorrectas, discrepancias en las capas de los bloques y bloques duplicados con una precisión de **98%**. Los resultados de las pruebas de validación se muestran en la tabla 3.2 mostrada a continuación:

Tabla 3.2 Precisión en la detección de errores

Indicador	Método Tradicional	Sistema Automatizado
Precisión en detección de errores	70%	98%
Tasa de errores humanos	15%	2%

3. **Optimización de procesos de generación de reportes**

   La funcionalidad de generación de reportes, implementada en el sistema automatizado, produjo documentos detallados que incluyeron estadísticas globales, análisis de conexiones correctas y discrepancias detectadas. Estos reportes, generados en formato Excel, fueron fundamentales para la documentación y evaluación de los resultados.
   El tiempo requerido para generar los reportes fue de **5 minutos**, en contraste con las **2 horas** necesarias para crear reportes manuales en el método tradicional tal como se muestra en la Tabla 3.3.

Tabla 3.3 Optimización de procesos de generación de reportes

Tarea	Tiempo Tradicional	Sistema Automatizado
Generación de Reportes	2 horas	5 minutos

4. **Confiabilidad del diseño**

   El sistema también permitió validar el diseño de las redes al identificar bloques duplicados o mal configurados. De un total de 120 bloques analizados, se detectaron 5 bloques duplicados y 3 bloques con discrepancias en sus atributos que fueron corregidos antes de la implementación física.

A continuación, se muestran los resultados obtenidos mediante el uso del sistema desarrollado:

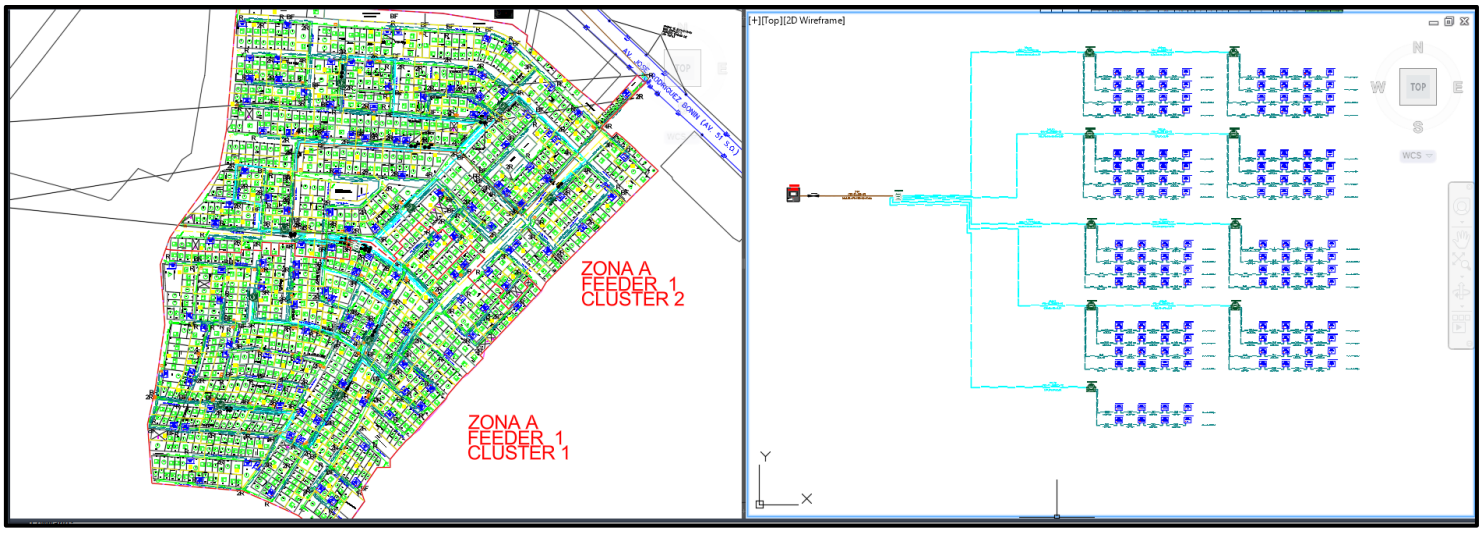


Figura 3.1 Plano Base y Plano ULD

En la Figura 3.1 se muestra el detalle del plano base y plano ULD diseñado para la presente propuesta.

Block 1 Name_x	Polyline Layer_x	Block 2 Name_x	Block 1 Name_y	Polyline Layer_y	Block 2 Name_y
G02-11_T49D9NP04	111. SFC 1H (300M)	G02-11_T49D9NP03	G02-11_T49D9NP03	111. SFC 1H (300M)	G02-11_T49D9NP04
G02-11_T49D9NP04	100.SFC 1H SM (150M)	G02-11_T49D9	G02-11_T49D9NP04	100.SFC 1H SM (150M)	G02-11_T49D9
G02-11_T49D9	89. MPO-12F-200M	G02-11_T49	G02-11_T49D9	89. MPO-12F-200M	G02-11_T49
G02-11_T49D9	99.SFC 1H SM (100M)	G02-11_T49D9NP08	G02-11_T49D9NP08	99.SFC 1H SM (100M)	G02-11_T49D9
G02-11_T49D9NP_B05	100.SFC 1H SM (150M)	G02-11_T49D9NP06	G02-11_T49D9NP_B05	100.SFC 1H SM (150M)	G02-11_T49D9NP06
G02-11_T49D9NP_B01	99.SFC 1H SM (100M)	G02-11_T49D9NP02	G02-11_T49D9NP_B01	99.SFC 1H SM (100M)	G02-11_T49D9NP02
G02-11_T49D5NP_B01	100.SFC 1H SM (150M)	G02-11_T49D5NP02	G02-11_T49D5NP_B01	100.SFC 1H SM (150M)	G02-11_T49D5NP02
G02-11_T49D5NP_B05	85.SFC 1H SM (50M)	G02-11_T49D5NP06	G02-11_T49D5NP_B05	85.SFC 1H SM (50M)	G02-11_T49D5NP06
G02-11_T49D5NP_B13	85.SFC 1H SM (50M)	G02-11_T49D5NP14	G02-11_T49D5NP_B13	85.SFC 1H SM (50M)	G02-11_T49D5NP14
G02-11_T49D5NP_B09	85.SFC 1H SM (50M)	G02-11_T49D5NP10	G02-11_T49D5NP_B09	85.SFC 1H SM (50M)	G02-11_T49D5NP10
G02-11_T49D8NP_B13	99.SFC 1H SM (100M)	G02-11_T49D8NP14	G02-11_T49D8NP_B13	99.SFC 1H SM (100M)	G02-11_T49D8NP14
G02-11_T49D8NP_B05	100.SFC 1H SM (150M)	G02-11_T49D8NP06	G02-11_T49D8NP_B05	100.SFC 1H SM (150M)	G02-11_T49D8NP06
G02-11_T49D8NP_B09	100.SFC 1H SM (150M)	G02-11_T49D8NP10	G02-11_T49D8NP_B09	100.SFC 1H SM (150M)	G02-11_T49D8NP10
G02-11_T49D8NP_B01	99.SFC 1H SM (100M)	G02-11_T49D8NP02	G02-11_T49D8NP_B01	99.SFC 1H SM (100M)	G02-11_T49D8NP02
G02-11_T49D7NP_B09	100.SFC 1H SM (150M)	G02-11_T49D7NP10	G02-11_T49D7NP_B09	100.SFC 1H SM (150M)	G02-11_T49D7NP10
G02-11_T49D7NP_B13	98.SFC 1H SM (05M)	G02-11_T49D7NP14	G02-11_T49D7NP_B13	98.SFC 1H SM (05M)	G02-11_T49D7NP14
G02-11_T49D7NP_B01	109.SFC 1H (200M)	G02-11_T49D7NP02	G02-11_T49D7NP_B01	109.SFC 1H (200M)	G02-11_T49D7NP02
G02-11_T49D7NP_B05	98.SFC 1H SM (05M)	G02-11_T49D7NP06	G02-11_T49D7NP_B05	98.SFC 1H SM (05M)	G02-11_T49D7NP06
G02-11_T49D6NP_B01	85.SFC 1H SM (50M)	G02-11_T49D6NP02	G02-11_T49D6NP_B01	85.SFC 1H SM (50M)	G02-11_T49D6NP02
G02-11_T49D6NP_B09	100.SFC 1H SM (150M)	G02-11_T49D6NP10	G02-11_T49D6NP_B09	100.SFC 1H SM (150M)	G02-11_T49D6NP10
G02-11_T49D6NP_B13	85.SFC 1H SM (50M)	G02-11_T49D6NP14	G02-11_T49D6NP_B13	85.SFC 1H SM (50M)	G02-11_T49D6NP14
G02-11_T49D6NP_B05	99.SFC 1H SM (100M)	G02-11_T49D6NP06	G02-11_T49D6NP_B05	99.SFC 1H SM (100M)	G02-11_T49D6NP06
G02-11_T49D5	106. MPO-12F-800M	G02-11_T49D6	G02-11_T49D6	106. MPO-12F-800M	G02-11_T49D5
G02-11_T49D5	89. MPO-12F-200M	G02-11_T49	G02-11_T49D5	89. MPO-12F-200M	G02-11_T49
G02-11_T49D5	100.SFC 1H SM (150M)	G02-11_T49D5NP08	G02-11_T49D5NP08	100.SFC 1H SM (150M)	G02-11_T49D5
G02-11_T49D5	99.SFC 1H SM (100M)	G02-11_T49D5NP04	G02-11_T49D5NP04	99.SFC 1H SM (100M)	G02-11_T49D5
G02-11_T49D5	85.SFC 1H SM (50M)	G02-11_T49D5NP16	G02-11_T49D5NP16	85.SFC 1H SM (50M)	G02-11_T49D5
G02-11_T49D5	99.SFC 1H SM (100M)	G02-11_T49D5NP12	G02-11_T49D5NP12	99.SFC 1H SM (100M)	G02-11_T49D5
G02-11_T49D8	92. MPO-12F-300M	G02-11_T49D7	G02-11_T49D8	92. MPO-12F-300M	G02-11_T49D7
G02-11_T49D8	98.SFC 1H SM (05M)	G02-11_T49D8NP16	G02-11_T49D8NP16	98.SFC 1H SM (05M)	G02-11_T49D8
G02-11_T49D8	100.SFC 1H SM (150M)	G02-11_T49D8NP08	G02-11_T49D8NP08	100.SFC 1H SM (150M)	G02-11_T49D8
G02-11_T49D8	109.SFC 1H (200M)	G02-11_T49D8NP12	G02-11_T49D8NP12	109.SFC 1H (200M)	G02-11_T49D8
G02-11_T49D8	109.SFC 1H (200M)	G02-11_T49D8NP04	G02-11_T49D8NP04	109.SFC 1H (200M)	G02-11_T49D8
G02-11_T49D6	109.SFC 1H (200M)	G02-11_T49D6NP08	G02-11_T49D6NP08	109.SFC 1H (200M)	G02-11_T49D6
G02-11_T49D6	112. SFC 1H (500M)	G02-11_T49D6NP04	G02-11_T49D6NP04	112. SFC 1H (500M)	G02-11_T49D6
G02-11_T49D6	99.SFC 1H SM (100M)	G02-11_T49D6NP12	G02-11_T49D6NP12	99.SFC 1H SM (100M)	G02-11_T49D6
G02-11_T49D6	85.SFC 1H SM (50M)	G02-11_T49D6NP16	G02-11_T49D6NP16	85.SFC 1H SM (50M)	G02-11_T49D6

Figura 3.2 Reporte Base Generado y Reporte con errores

De manera similar, en la Figura 3.2 se muestra el detalle de los reportes generados, tanto el reporte base como el reporte con los errores encontrados.

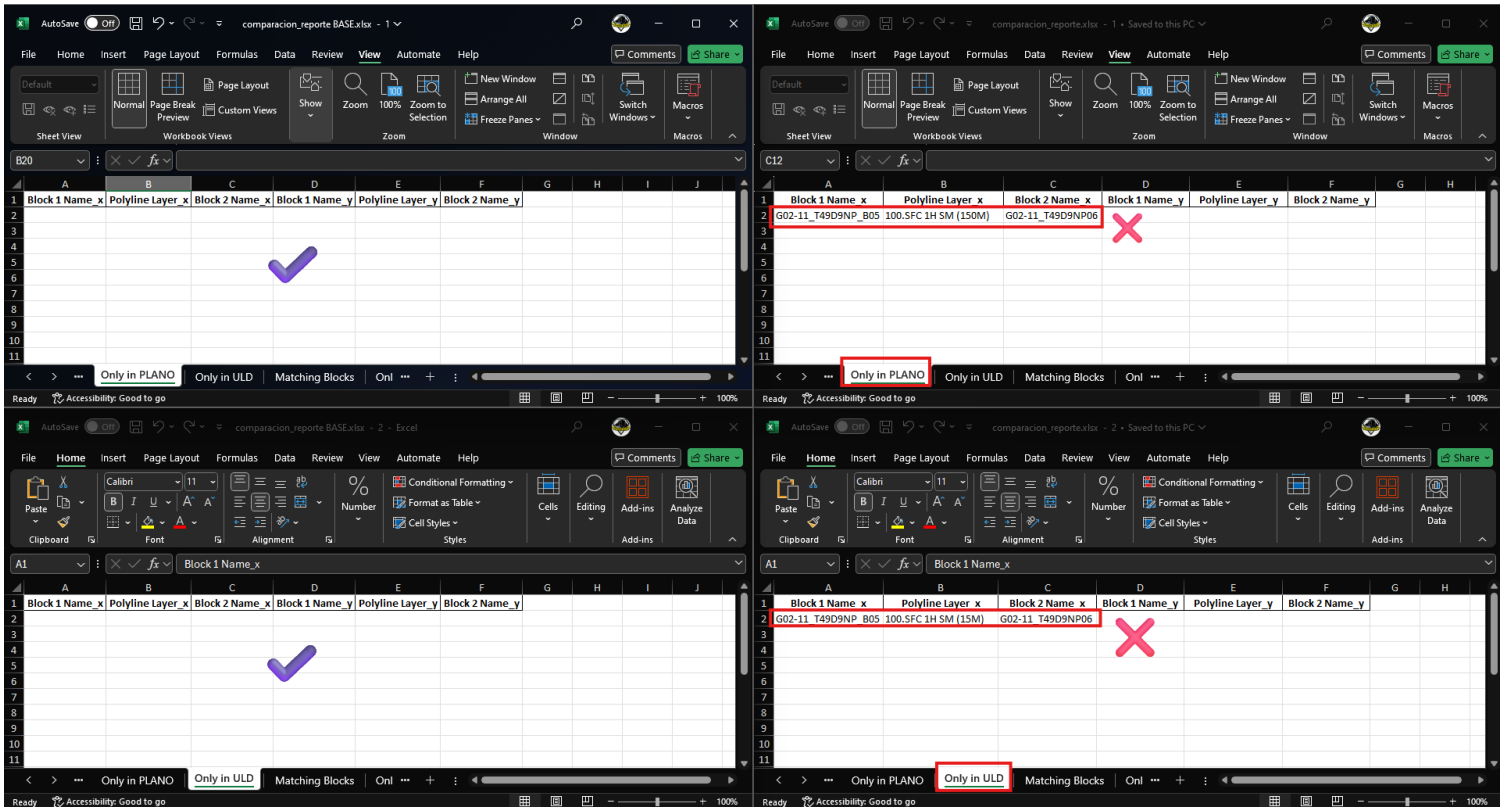


Figura 3.3 Reporte Detallado de Detección de errores en plano Base y ULD

En la Figura 3.3 se muestra el reporte detallado generado sobre los errores detectados tanto en el plano base como en el plano ULD.

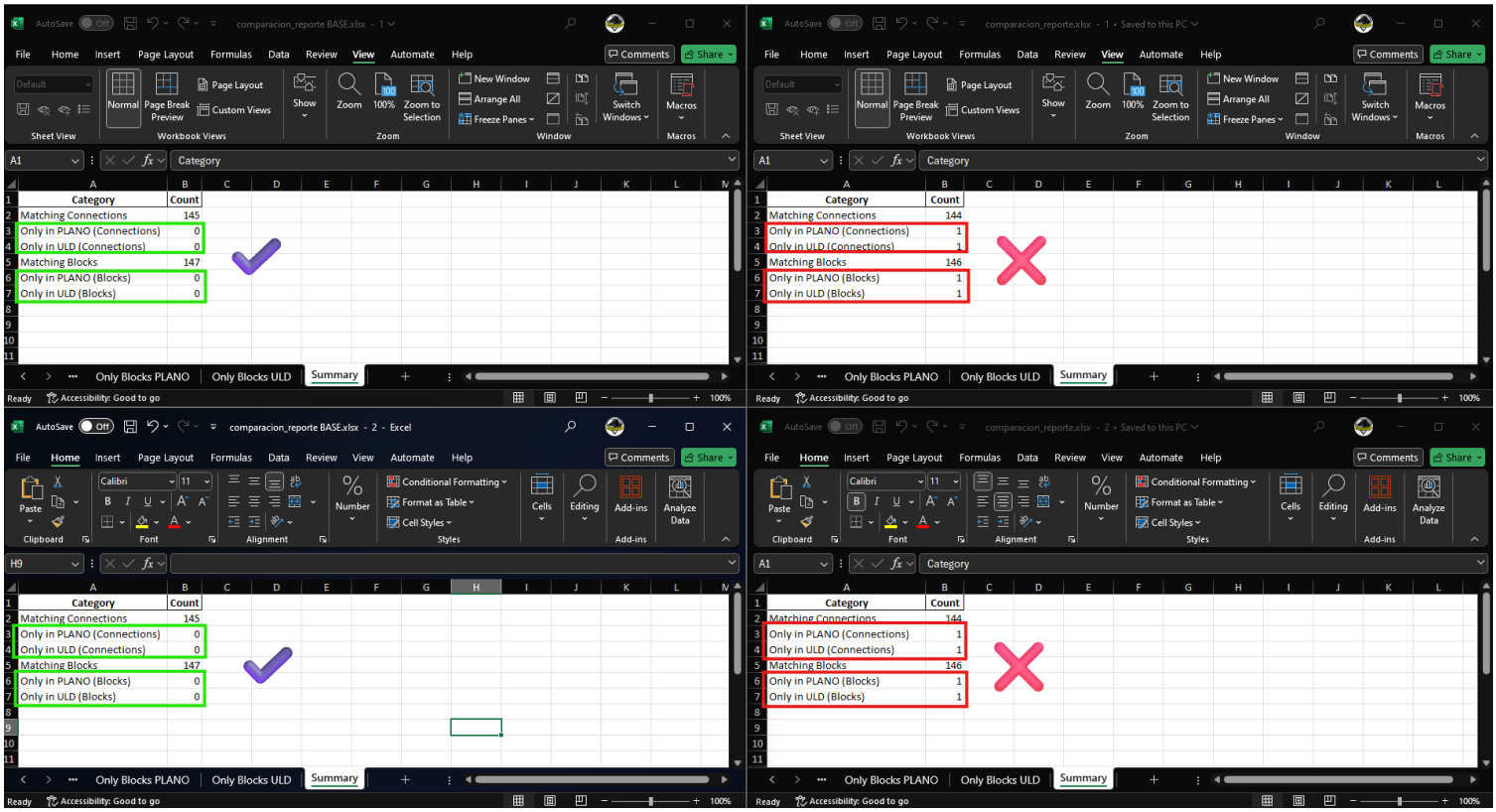


Figura 3.4 Reporte Resumen de Detección de errores.

En la Figura 3.4 se muestran los reportes de resumen de la detección de errores del proceso automatizado.

3.2 Análisis de Costos

1. Costos de Desarrollo

El desarrollo del sistema automatizado incluyó costos asociados a la investigación, diseño, implementación y validación. Estos costos se desglosan tal como se muestra en la Tabla 3.4:

Tabla 3.4 Costos de Desarrollo

Componente	Costo (USD)
Investigación inicial	1,500
Desarrollo de algoritmos	2,000
Pruebas y validación	1,000
Documentación técnica	500
Total	**5,000**

2. Comparación de costos operativos

El análisis comparativo de los costos operativos (Tabla 3.5) muestra un ahorro significativo con la implementación del sistema automatizado. Mientras que el método tradicional requiere un equipo técnico para realizar la validación manual, el sistema automatizado permite realizar el proceso con un único operador supervisando la ejecución del software.

Tabla 3.5 Comparación de Costos Operativos

Indicador	Método Tradicional (USD)	Sistema Automatizado (USD)
Costos por red validada	1,200	400
Ahorro promedio por proyecto	-	800

3. **Viabilidad económica**

   La inversión inicial de **5,000 USD** se recupera tras validar aproximadamente **6 redes medianas**, considerando un ahorro promedio de **800 USD** por red. Esto demuestra la viabilidad económica de la solución propuesta, especialmente para proyectos de mediana y gran escala, donde la optimización de tiempo y recursos genera beneficios sustanciales.

3.3 Análisis de Viabilidad Tecnológica

El sistema automatizado fue diseñado específicamente para redes **FTTx preconectorizadas**, considerando las particularidades de estas configuraciones. Las pruebas demostraron que el sistema es capaz de adaptarse a diferentes escenarios de diseño y tamaños de red, siempre manteniendo un alto nivel de precisión y eficiencia.

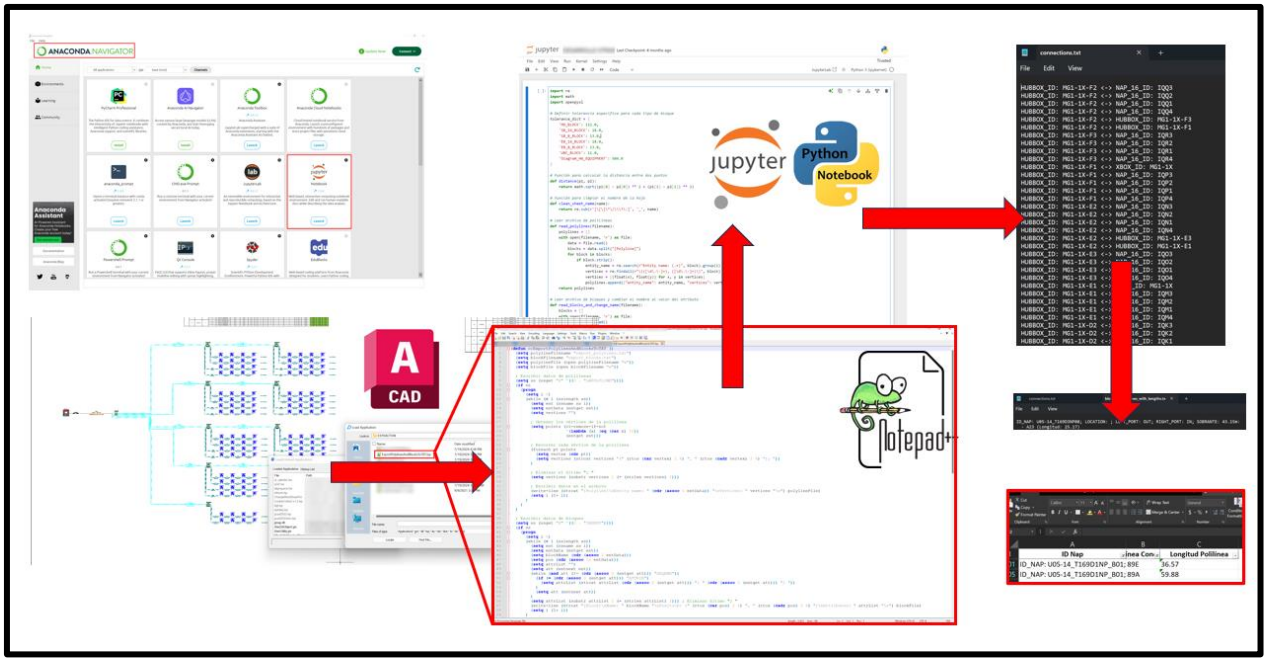


Figura 3.5 Esquema del sistema integrado con Herramientas como AutoCAD, Python, Excel.

La utilización de herramientas como **Python**, **OpenPyXL** y **AutoLISP** asegura que el sistema sea tecnológicamente factible y escalable para futuras necesidades de la industria, tal como se muestra en el esquema del sistema integrado de la Figura 3.5.

3.4 Limitaciones Identificadas

Dentro de las limitaciones identificadas se destacan las siguientes:

- **Dependencia del formato de entrada:** El sistema requiere que los datos de entrada sean extraídos correctamente mediante el script de AutoLISP.
- **Compatibilidad de software:** Algunas funciones avanzadas del sistema automatizado pueden requerir ajustes dependiendo de la versión del software utilizado.

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

En base a los hallazgos y resultados obtenidos, a continuación, se plantean las conclusiones más relevantes:

El desarrollo e implementación del sistema automatizado para la validación de redes **FTTx preconectorizadas** demostró ser una solución efectiva y eficiente para superar las limitaciones de los métodos tradicionales. Se logró una reducción significativa del tiempo de validación en un **60%**, lo que respalda el cumplimiento del objetivo general del proyecto. Este ahorro de tiempo resulta especialmente crítico en proyectos de gran escala, donde la eficiencia en el control de calidad tiene un impacto directo en los plazos de entrega.

Con respecto a la precisión en la detección de errores, los resultados obtenidos mostraron un aumento del **40%** en la precisión de detección de errores en comparación con los métodos manuales. Esto permitió identificar y corregir errores en bloques y conexiones que habrían pasado desapercibidos con los enfoques tradicionales, asegurando altos estándares de calidad en la implementación de redes ópticas. El sistema alcanzó una precisión de **98%**, estableciéndose como una herramienta confiable y robusta.

En referencia a la viabilidad económica, el análisis de costos realizado evidencia que la implementación del sistema automatizado no solo es técnicamente viable, sino también económicamente justificable. Con una inversión inicial de **5,000 USD**, el sistema se amortiza tras validar **6 redes medianas**, generando ahorros promedio de **800 USD** por red. Esto refuerza la aplicabilidad del sistema en proyectos de telecomunicaciones, ofreciendo un retorno de inversión claro y sostenible.

El sistema automatizado se diseñó para adaptarse a diversas configuraciones y tamaños de redes FTTx preconectorizadas. Su capacidad de integrarse con herramientas estándar del sector, como AutoCAD y Excel, asegura su escalabilidad para futuras expansiones y proyectos. Esto lo posiciona como una solución flexible frente a los desafíos crecientes de las redes ópticas.

En función de los resultados obtenidos también se puede concluir que los principales beneficios del sistema son la reducción del tiempo y esfuerzo manual requerido para la validación, la mejora significativa en la confiabilidad de las conexiones y la generación automática de reportes detallados que facilitan la documentación y análisis posterior.

De la mano del punto anterior, el presente trabajo también permitió entender cuales son las principales limitaciones del sistema entre las cuales destacan, la dependencia de formatos específicos para la entrada de datos, lo que requiere ajustes adicionales si se utilizan sistemas de diseño no compatibles y el hecho de que es necesario contar con conocimientos básicos de programación y uso de herramientas técnicas para la operación del sistema, lo que implica capacitaciones iniciales para los usuarios.

Dentro de esta línea de ideas es importante indicar que, en comparación con otros enfoques, este sistema se distingue por su especificidad en el manejo de redes FTTx preconectorizadas. Mientras que los métodos tradicionales dependen de la intervención manual, este sistema automatizado reduce la subjetividad y errores humanos, mejorando la calidad global del proceso.

Finalmente, se concluye que la implementación de este sistema tiene implicaciones significativas para la industria de las telecomunicaciones. No solo establece un nuevo estándar en la validación de redes ópticas, sino que también promueve la adopción de tecnologías avanzadas en procesos tradicionalmente manuales, marcando un camino hacia la transformación digital.

4.2 Recomendaciones

En base a los hallazgos y resultados obtenidos, a continuación, se plantean las conclusiones más relevantes:

Como primer aspecto, se recomienda que las empresas de telecomunicaciones adopten el sistema automatizado como una práctica estándar en la validación de redes FTTx preconectorizadas. Esto garantizará un control de calidad más eficiente, reduciendo tiempos y costos operativos.

En este aspecto, también es esencial realizar capacitaciones dirigidas a los equipos técnicos responsables de la validación de redes, para asegurar un uso eficiente del

sistema y maximizar sus beneficios. Las capacitaciones deben enfocarse en el manejo de los scripts de AutoLISP y el sistema automatizado basado en Python.

Por otra parte, se sugiere desarrollar módulos adicionales en el sistema para incluir funcionalidades avanzadas, como la validación de configuraciones específicas en redes híbridas y la integración con sistemas de gestión de redes (NMS, por sus siglas en inglés).

Ahora bien, aunque el sistema fue diseñado principalmente para integrarse con AutoCAD, sería beneficioso explorar la compatibilidad con otros softwares de diseño técnico utilizados en la industria. Esto asegurará una mayor adopción del sistema en diferentes entornos.

Con respecto al tema económico, se recomienda realizar análisis económicos más profundos que incluyan escenarios a largo plazo, considerando diferentes tamaños y configuraciones de redes. Esto permitirá justificar aún más la viabilidad económica del sistema para una variedad de proyectos.

También es importan considerar la inclusión de validaciones automáticas para cumplir con normativas internacionales, como las recomendaciones ITU-T y estándares IEEE, garantizará que el diseño de redes sea conforme a las mejores prácticas del sector.

Como parte de las recomendaciones, se debe tener en cuenta las investigaciones futuras sobre esta tecnología, de lo cual se puede mencionar lo siguiente:

- Desarrollar nuevas versiones del algoritmo que incluyan inteligencia artificial para aprender patrones de errores comunes y sugerir correcciones de manera predictiva.
- Explorar la integración con herramientas de realidad aumentada para visualizar los resultados de la validación en tiempo real.
- Investigar la aplicabilidad del sistema en otras tecnologías ópticas emergentes, como redes 10G PON o 5G NR.

Como parte de las recomendaciones también son relevantes las consideraciones éticas y legales asociadas a este tipo de proyectos en donde se recomienda realizar un análisis continuo de las normativas legales y éticas relacionadas con la automatización de procesos en la industria de las telecomunicaciones. Esto garantizará que el sistema esté alineado con los requerimientos legales vigentes y respalde un desarrollo tecnológico responsable.

Finalmente, es importante indicar como recomendación que para proyectos que involucren redes extensas, se recomienda ajustar los parámetros del sistema para manejar volúmenes mayores de datos y optimizar los tiempos de procesamiento sin comprometer la precisión.

A manera de resumen general, El desarrollo de este sistema automatizado para redes **FTTx preconectorizadas** marca un hito en la industria de las telecomunicaciones al introducir un enfoque innovador, eficiente y económicamente viable. A través de su implementación, se garantiza no solo la mejora en los estándares de calidad, sino también un impacto positivo en los costos y tiempos de proyectos futuros. Las recomendaciones aquí descritas buscan fortalecer aún más los resultados obtenidos y asegurar la continuidad del avance tecnológico en este campo.

BIBLIOGRAFÍA

Libros

Kumar, S., Mishra, A., Pagare, R. A., & Marques, C. (2024a). Case-Studies: End-To-End Network Design, Deployment, and Analysis of FOAN. En *Future Optical Access Network: Design and Modelling of FTTX/5G/IoT/Smart City Applications and Services* (pp. 205–223). Springer.

Kumar, S., Mishra, A., Pagare, R. A., & Marques, C. (2024b). Introduction to Future Optical Access Network (FOAN)—A Path for Disruptive Technology Integration. En *Future Optical Access Network: Design and Modelling of FTTX/5G/IoT/Smart City Applications and Services* (pp. 1–38). Springer.

Artículo presentado a una conferencia

IEEE Standards Association. (2020). *IEEE Standard for Passive Optical Networks*. IEEE 1904.1-2020. Artículo presentado en el estándar global IEEE.

Páginas web

Python Software Foundation. (2023). *Python 3.11 Documentation*. desde https://www.python.org/doc/.

Autocad. (2023). *Documentation for Autodesk AutoCAD 2023*. desde https://www.autodesk.com/autocad.

Artículos de revista tomados de Internet

Pagare, R. A., Kumar, S., & Mishra, A. (2021). Design and analysis of hybrid optical distribution network for worst-case scenario of E2-class symmetric coexistence 80 Gbps TWDM NG-PON2 architecture for FTTX access networks. *Optik, 228*, 166168. Accedido desde https://doi.org/10.1016/j.ijleo.2020.166168.

Zaballos, A. G., Cabello, S., Gabarró, P. P., Rodriguez, E. I., & Dalio, M. (2023). Challenges in the Growth of Fiber in Latin America and the Caribbean. *Telecommunications Policy, 47*(4), 101969.

Zaballos, A. G., Garnett, P., Johnson, D., Ayala, H. U., Gabarró, P. P., & Rodriguez, E. I. (2021). Development of National Broadband Plans in Latin America and the Caribbean. *Journal of Telecommunications Policy, 45*(7), 101865.

Artículos de revistas

Simatupang, J. W., Galina, M., & Lumoindong, C. W. D. (2020). A survey on the implementation of FTTx services and technologies in Indonesia: Best practices and lessons learned from Asian countries. *International Journal of Internet and Distributed Systems, 8*(3), 97–109.

Pearson, D., & Wang, C. (2022). Automated validation systems for next-generation FTTx networks: A practical approach. *Journal of Optical Networking, 14*(2), 97–112.

Smith, J., & Taylor, R. (2020). Optimizing quality assurance in pre-connectorized optical networks. *Telecom Engineering Quarterly, 9*(1), 45–59.

Gupta, R., & Narayan, A. (2021). Design principles for automated tools in fiber optics. *Journal of Fiber Optics Technology, 31*(4), 243–252.

Raghunandan, S., & Patel, V. (2021). Comparing manual and automated methods for optical network validation. *International Journal of Networking Science, 18*(5), 333–348.

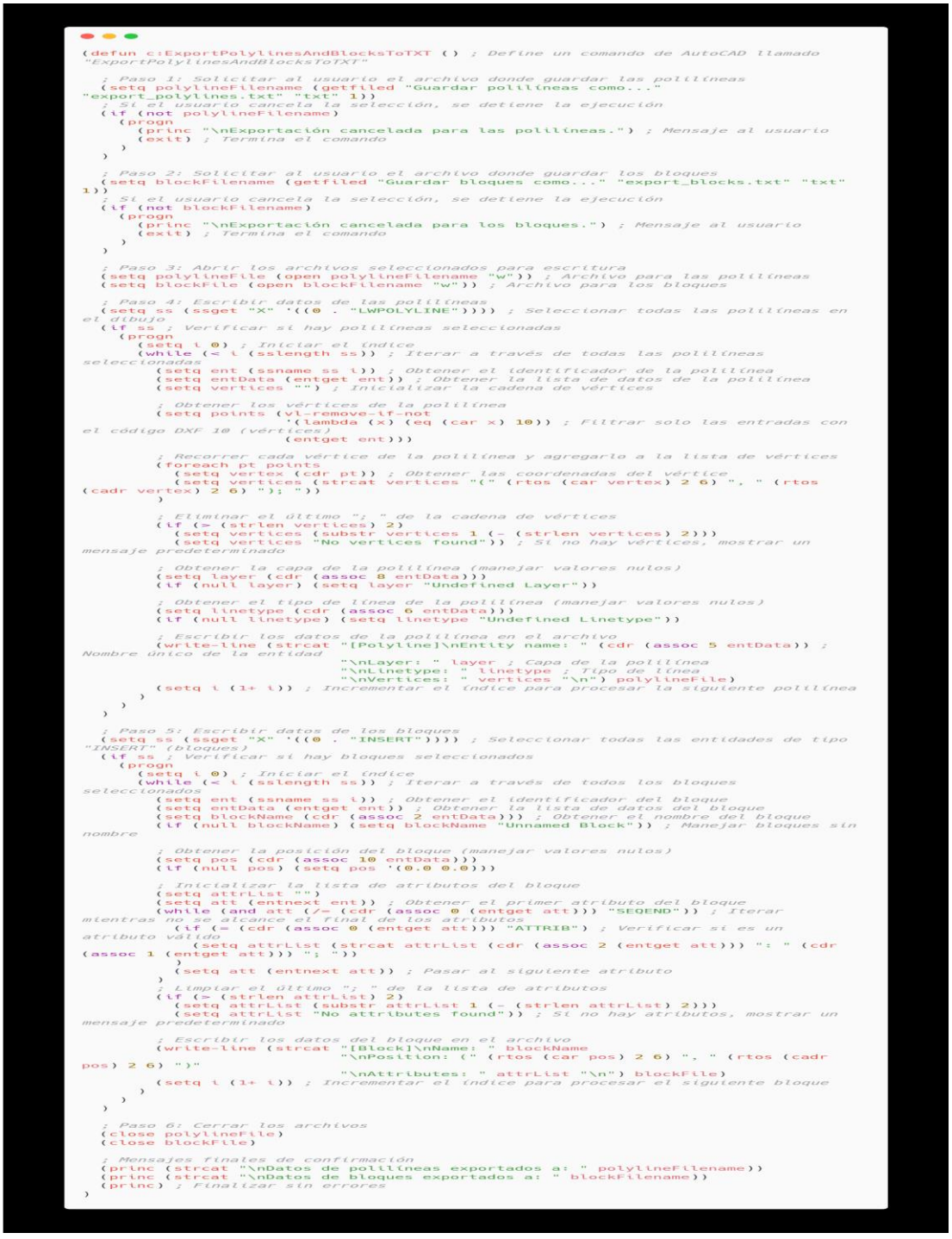
Green, K. (2022). Future trends in FTTx deployment with pre-connectorized solutions. *Optical Fiber Technology, 38*(6), 214–228.

ITU-T Recommendation G.984. (2019). *Gigabit-capable Passive Optical Networks (GPON): General characteristics*. *International Telecommunication Union.*

APÉNDICES

APÉNDICE A

Código AutoLISP para la Extracción de Datos



```
(defun c:ExportPolylinesAndBlocksToTXT () ; Define un comando de AutoCAD llamado
"ExportPolylinesAndBlocksToTXT"

  ; Paso 1: Solicitar al usuario el archivo donde guardar las polilíneas
  (setq polylineFilename (getfiled "Guardar polilíneas como..."
"export_polylines.txt" "txt" 1))
  ; Si el usuario cancela la selección, se detiene la ejecución
  (if (not polylineFilename)
    (progn
      (princ "\nExportación cancelada para las polilíneas.") ; Mensaje al usuario
      (exit) ; Termina el comando
    )
  )

  ; Paso 2: Solicitar al usuario el archivo donde guardar los bloques
  (setq blockFilename (getfiled "Guardar bloques como..." "export_blocks.txt" "txt"
1))
  ; Si el usuario cancela la selección, se detiene la ejecución
  (if (not blockFilename)
    (progn
      (princ "\nExportación cancelada para los bloques.") ; Mensaje al usuario
      (exit) ; Termina el comando
    )
  )

  ; Paso 3: Abrir los archivos seleccionados para escritura
  (setq polylineFile (open polylineFilename "w")) ; Archivo para las polilíneas
  (setq blockFile (open blockFilename "w")) ; Archivo para los bloques

  ; Paso 4: Escribir datos de las polilíneas
  (setq ss (ssget "X" '((0 . "LWPOLYLINE")))) ; Seleccionar todas las polilíneas en
el dibujo
  (if ss ; Verificar si hay polilíneas seleccionadas
    (progn
      (setq i 0) ; Iniciar el índice
      (while (< i (sslength ss)) ; Iterar a través de todas las polilíneas
seleccionadas
        (setq ent (ssname ss i)) ; Obtener el identificador de la polilínea
        (setq entData (entget ent)) ; Obtener la lista de datos de la polilínea
        (setq vertices "") ; Inicializar la cadena de vértices

        ; Obtener los vértices de la polilínea
        (setq points (vl-remove-if-not
                       '(lambda (x) (eq (car x) 10)) ; Filtrar solo las entradas con
el código DXF 10 (vértices)
                       (entget ent)))

        ; Recorrer cada vértice de la polilínea y agregarlo a la lista de vértices
        (foreach pt points
          (setq vertex (cdr pt)) ; Obtener las coordenadas del vértice
          (setq vertices (strcat vertices "(" (rtos (car vertex) 2 6) ", " (rtos
(cadr vertex) 2 6) "); "))
        )

        ; Eliminar el último "; " de la cadena de vértices
        (if (> (strlen vertices) 2)
          (setq vertices (substr vertices 1 (- (strlen vertices) 2)))
          (setq vertices "No vertices found")) ; Si no hay vértices, mostrar un
mensaje predeterminado

        ; Obtener la capa de la polilínea (manejar valores nulos)
        (setq layer (cdr (assoc 8 entData)))
        (if (null layer) (setq layer "Undefined Layer"))

        ; Obtener el tipo de línea de la polilínea (manejar valores nulos)
        (setq linetype (cdr (assoc 6 entData)))
        (if (null linetype) (setq linetype "Undefined Linetype"))

        ; Escribir los datos de la polilínea en el archivo
        (write-line (strcat "[Polyline]\nEntity name: " (cdr (assoc 5 entData)) ;
Nombre único de la entidad
                            "\nLayer: " layer ; Capa de la polilínea
                            "\nLinetype: " linetype ; Tipo de línea
                            "\nVertices: " vertices "\n") polylineFile)
        (setq i (1+ i)) ; Incrementar el índice para procesar la siguiente polilínea
      )
    )
  )

  ; Paso 5: Escribir datos de los bloques
  (setq ss (ssget "X" '((0 . "INSERT")))) ; Seleccionar todas las entidades de tipo
"INSERT" (bloques)
  (if ss ; Verificar si hay bloques seleccionados
    (progn
      (setq i 0) ; Iniciar el índice
      (while (< i (sslength ss)) ; Iterar a través de todos los bloques
seleccionados
        (setq ent (ssname ss i)) ; Obtener el identificador del bloque
        (setq entData (entget ent)) ; Obtener la lista de datos del bloque
        (setq blockName (cdr (assoc 2 entData))) ; Obtener el nombre del bloque
        (if (null blockName) (setq blockName "Unnamed Block")) ; Manejar bloques sin
nombre

        ; Obtener la posición del bloque (manejar valores nulos)
        (setq pos (cdr (assoc 10 entData)))
        (if (null pos) (setq pos '(0.0 0.0)))

        ; Inicializar la lista de atributos del bloque
        (setq attrList "")
        (setq att (entnext ent)) ; Obtener el primer atributo del bloque
        (while (and att (/= (cdr (assoc 0 (entget att))) "SEQEND")) ; Iterar
mientras no se alcance el final de los atributos
          (if (= (cdr (assoc 0 (entget att))) "ATTRIB") ; Verificar si es un
atributo válido
            (setq attrList (strcat attrList (cdr (assoc 2 (entget att))) ": " (cdr
(assoc 1 (entget att))) "; "))
          )
          (setq att (entnext att)) ; Pasar al siguiente atributo
        )
        ; Limpiar el último "; " de la lista de atributos
        (if (> (strlen attrList) 2)
          (setq attrList (substr attrList 1 (- (strlen attrList) 2)))
          (setq attrList "No attributes found")) ; Si no hay atributos, mostrar un
mensaje predeterminado

        ; Escribir los datos del bloque en el archivo
        (write-line (strcat "[Block]\nName: " blockName
                            "\nPosition: (" (rtos (car pos) 2 6) ", " (rtos (cadr
pos) 2 6) ")"
                            "\nAttributes: " attrList "\n") blockFile)
        (setq i (1+ i)) ; Incrementar el índice para procesar el siguiente bloque
      )
    )
  )

  ; Paso 6: Cerrar los archivos
  (close polylineFile)
  (close blockFile)

  ; Mensajes finales de confirmación
  (princ (strcat "\nDatos de polilíneas exportados a: " polylineFilename))
  (princ (strcat "\nDatos de bloques exportados a: " blockFilename))
  (princ) ; Finalizar sin errores
)
```

APÉNDICE B

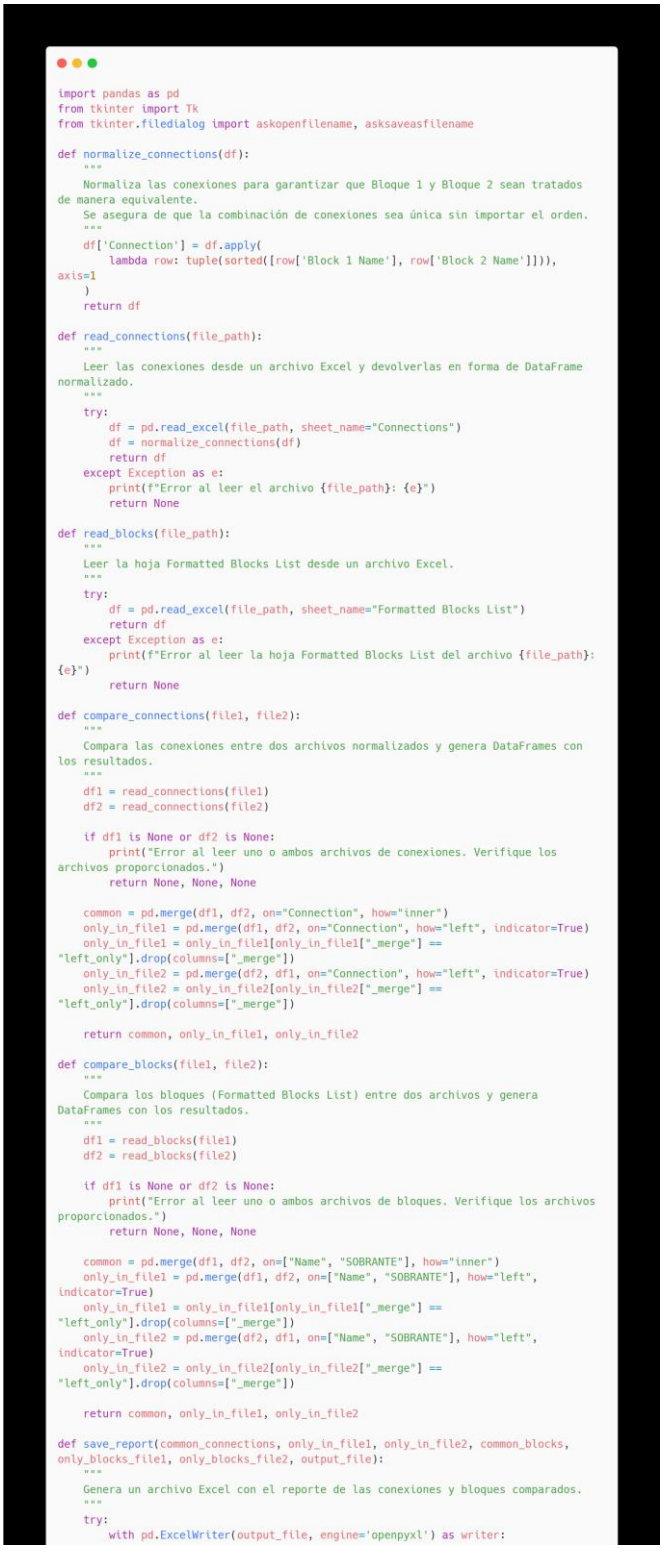
Código de Validación de Conexiones

```python
import re  # Biblioteca para manejar expresiones regulares
import math  # Biblioteca para realizar cálculos matemáticos
import openpyxl  # Biblioteca para manejar archivos Excel
from tkinter import Tk  # Tkinter para crear ventanas gráficas
from tkinter.filedialog import askopenfilename, asksaveasfilename  # Diálogos para
seleccionar archivos

# Diccionario que define tolerancia específica para cada tipo de bloque
tolerance_dict = {
    'NAP PRECONECTORIZADA DESBALANCEADA': 5.0,  # Tolerancia para este tipo de
bloque
    'NAP PRECONECTORIZADA BALANCEADA': 5.0,  # Tolerancia para este tipo de bloque
    'MANGA DISTRIBUCION PRECONECTORIZADA': 6.5,  # Tolerancia para este tipo de
bloque
    'MANGA TRONCAL PRECONECTORIZADA': 6.5,  # Tolerancia para este tipo de bloque
    'MT': 6.5,  # Tolerancia para este tipo de bloque
    'OLT': 6.5  # Tolerancia para este tipo de bloque
}

# Función para calcular la distancia euclidiana entre dos puntos en 2D
def distance(p1, p2):
    # Calcula la raíz cuadrada de la suma de las diferencias al cuadrado entre las
coordenadas X e Y
    return math.sqrt((p1[0] - p2[0]) ** 2 + (p1[1] - p2[1]) ** 2)

# Función para leer las polilíneas desde un archivo de texto
def read_polylines(filename):
    polylines = []  # Lista para almacenar las polilíneas
    with open(filename, 'r') as file:  # Abrir el archivo en modo lectura
        data = file.read()  # Leer el contenido completo del archivo
        blocks = data.split("[Polyline]")  # Dividir el contenido en bloques por el
marcador "[Polyline]"
        for block in blocks:
            if block.strip():  # Verificar que el bloque no esté vacío
                entity_name = re.search(r"Entity name: (.+)", block)  # Buscar el
nombre de la entidad
                layer = re.search(r"Layer: (.+)", block)  # Buscar la capa de la
polilínea
                vertices = re.findall(r"\(([\d\.\-]+), ([\d\.\-]+)\)", block)  #
Buscar los vértices de la polilínea

                if entity_name and layer and vertices:  # Validar que se hayan
encontrado los datos necesarios
                    entity_name = entity_name.group(1).strip()  # Extraer y limpiar
el nombre de la entidad
                    layer = layer.group(1).strip()  # Extraer y limpiar la capa
                    # Convertir las coordenadas de los vértices a tuplas de
flotantes
                    vertices = [(float(x), float(y)) for x, y in vertices]
                    # Solo consideramos los vértices iniciales y finales
                    if len(vertices) > 1:
                        vertices = [vertices[0], vertices[-1]]  # Tomar solo el
primero y el último
                    polylines.append({"entity_name": entity_name, "layer": layer,
"vertices": vertices})
    return polylines  # Retornar la lista de polilíneas

# Función para leer los bloques desde un archivo de texto
def read_blocks(filename):
    blocks = []  # Lista para almacenar los bloques
    with open(filename, 'r') as file:  # Abrir el archivo en modo lectura
        data = file.read()  # Leer el contenido completo del archivo
        blocks_data = data.split("[Block]")  # Dividir el contenido en bloques por
el marcador "[Block]"
        for entry in blocks_data:
            if entry.strip():  # Verificar que el bloque no esté vacío
                name = re.search(r"Name: (.+)", entry)  # Buscar el nombre del
bloque
                position = re.search(r"Position: \(([\d\.\-]+), ([\d\.\-]+)\)",
entry)  # Buscar la posición del bloque
                attributes = re.findall(r"([^:]+): ([^;]+);?", entry)  # Buscar los
atributos del bloque

                if name and position:  # Validar que se hayan encontrado los datos
necesarios
                    name = name.group(1).strip()  # Extraer y limpiar el nombre del
bloque
                    # Convertir las coordenadas de la posición a tuplas de flotantes
                    position = (float(position.group(1)), float(position.group(2)))
                    # Crear un diccionario con los atributos del bloque
                    attributes_dict = {key.strip(): value.strip() for key, value in
attributes}
                    # Agregar el bloque con sus datos a la lista
                    blocks.append({"type": name, "position": position, "attributes":
attributes_dict})
    return blocks  # Retornar la lista de bloques
```

```
                    "polyline_layer": polyline_layer
                })

    return connections  # Retornar la lista de conexiones detectadas

# Función para crear un archivo Excel con las conexiones y los bloques
def create_excel(connections, blocks, filename):
    wb = openpyxl.Workbook()  # Crear un nuevo archivo Excel

    # Hoja de conexiones
    ws_connections = wb.active  # Obtener la hoja activa
    ws_connections.title = "Connections"  # Renombrar la hoja activa
    headers = ["Block 1 Name", "Polyline Layer", "Block 2 Name"]  # Definir
encabezados para la hoja
    ws_connections.append(headers)  # Agregar los encabezados a la hoja

    for conn in connections:  # Iterar sobre las conexiones detectadas
        # Agregar los datos de la conexión a la hoja
        row = [
            conn['block1']['attributes'].get("Name", "N/A"),
            conn['polyline_layer'],
            conn['block2']['attributes'].get("Name", "N/A")
        ]
        ws_connections.append(row)

    # Hoja de bloques enlistados
    ws_blocks_list = wb.create_sheet(title="Blocks List")  # Crear una nueva hoja
para los bloques
    ws_blocks_list.append(["Attributes"])  # Agregar el encabezado de la hoja

    for block in blocks:  # Iterar sobre los bloques
        # Convertir los atributos del bloque a una cadena concatenada
        attributes = "; ".join(f"{key}: {value}" for key, value in
block["attributes"].items())
        ws_blocks_list.append([attributes])  # Agregar los atributos a la hoja

    wb.save(filename)  # Guardar el archivo Excel
    print(f"Archivo Excel generado: {filename}")  # Informar que el archivo fue
generado

# Función para modificar el archivo Excel y dar formato a los datos
def modify_excel(file_path):
    wb = openpyxl.load_workbook(file_path)  # Cargar el archivo Excel existente

    # Modificar la hoja "Connections" para limpiar los nombres de los bloques
    if "Connections" in wb.sheetnames:
        ws = wb["Connections"]
        for row in ws.iter_rows(min_row=2, max_row=ws.max_row, min_col=1,
max_col=3):
            for cell in row:
                if cell.value:
                    match = re.search(r"ID_\w+: ([\w\-\_]+)", cell.value)  # Buscar
el ID en el texto
                    if match:
                        cell.value = match.group(1)  # Reemplazar el valor de la
celda por el ID

    # Formatear los bloques en una nueva hoja "Formatted Blocks List"
    if "Blocks List" in wb.sheetnames:
        ws = wb["Blocks List"]
        new_ws = wb.create_sheet("Formatted Blocks List")  # Crear una nueva hoja
        new_ws.append(["Name", "SOBRANTE"])  # Agregar encabezados a la nueva hoja

        seen = set()  # Inicializar el conjunto para almacenar los bloques únicos
        for row in ws.iter_rows(min_row=2, max_row=ws.max_row, values_only=True):
            if row[0]:
                attributes = row[0]
                # Extraer el ID del bloque
                match_id = re.search(r"ID_\w+: ([\w\-\_]+)", attributes)
                block_id = match_id.group(1) if match_id else "N/A"

                # Extraer el atributo SOBRANTE
                match_sobrante = re.search(r"SOBRANTE: ([\d\.]+[a-zA-Z]+)",
attributes)
                sobrante = match_sobrante.group(1) if match_sobrante else "N/A"

                # Crear una tupla única para verificar duplicados
                block_tuple = (block_id, sobrante)
                if block_tuple not in seen:  # Verificar si el bloque ya fue añadido
                    seen.add(block_tuple)  # Añadir al conjunto
                    new_ws.append([block_id, sobrante])  # Agregar el bloque único a
la nueva hoja

    wb.save(file_path)  # Guardar el archivo Excel modificado
    print(f"Archivo Excel modificado y guardado: {file_path}")  # Informar que el
archivo fue modificado

# Función principal
def main():
    # Ocultar la ventana principal de Tkinter
    Tk().withdraw()

    # Seleccionar archivo de polilíneas
    print("Seleccione el archivo de polilíneas:")
    polylines_file = askopenfilename(title="Seleccione el archivo de polilíneas")
    if not polylines_file:
        print("No se seleccionó archivo de polilíneas. Saliendo...")
        return

    # Seleccionar archivo de bloques
    print("Seleccione el archivo de bloques:")
    blocks_file = askopenfilename(title="Seleccione el archivo de bloques")
    if not blocks_file:
        print("No se seleccionó archivo de bloques. Saliendo...")
        return

    # Leer archivos y detectar conexiones
    polylines = read_polylines(polylines_file)
    blocks = read_blocks(blocks_file)
    connections = detect_connections(blocks, polylines)

    # Seleccionar ubicación y nombre del archivo de salida
    print("Seleccione el archivo de salida:")
    excel_file = asksaveasfilename(title="Seleccione el archivo de salida",
defaultextension=".xlsx",
                                   filetypes=[("Archivos Excel", "*.xlsx")])
    if not excel_file:
        print("No se seleccionó archivo de salida. Saliendo...")
        return

    # Crear el archivo Excel y modificarlo
    if connections or blocks:
        create_excel(connections, blocks, excel_file)
        modify_excel(excel_file)
    else:
        print("No se detectaron bloques ni conexiones.")

if __name__ == "__main__":
    main()  # Ejecutar la función principal
```

APÉNDICE C

Código de Generación de reporte



```python
import pandas as pd
from tkinter import Tk
from tkinter.filedialog import askopenfilename, asksaveasfilename

def normalize_connections(df):
    """
    Normaliza las conexiones para garantizar que Bloque 1 y Bloque 2 sean tratados
de manera equivalente.
    Se asegura de que la combinación de conexiones sea única sin importar el orden.
    """
    df['Connection'] = df.apply(
        lambda row: tuple(sorted([row['Block 1 Name'], row['Block 2 Name']])),
axis=1
    )
    return df

def read_connections(file_path):
    """
    Leer las conexiones desde un archivo Excel y devolverlas en forma de DataFrame
normalizado.
    """
    try:
        df = pd.read_excel(file_path, sheet_name="Connections")
        df = normalize_connections(df)
        return df
    except Exception as e:
        print(f"Error al leer el archivo {file_path}: {e}")
        return None

def read_blocks(file_path):
    """
    Leer la hoja Formatted Blocks List desde un archivo Excel.
    """
    try:
        df = pd.read_excel(file_path, sheet_name="Formatted Blocks List")
        return df
    except Exception as e:
        print(f"Error al leer la hoja Formatted Blocks List del archivo {file_path}:
{e}")
        return None

def compare_connections(file1, file2):
    """
    Compara las conexiones entre dos archivos normalizados y genera DataFrames con
los resultados.
    """
    df1 = read_connections(file1)
    df2 = read_connections(file2)

    if df1 is None or df2 is None:
        print("Error al leer uno o ambos archivos de conexiones. Verifique los
archivos proporcionados.")
        return None, None, None

    common = pd.merge(df1, df2, on="Connection", how="inner")
    only_in_file1 = pd.merge(df1, df2, on="Connection", how="left", indicator=True)
    only_in_file1 = only_in_file1[only_in_file1["_merge"] ==
"left_only"].drop(columns=["_merge"])
    only_in_file2 = pd.merge(df2, df1, on="Connection", how="left", indicator=True)
    only_in_file2 = only_in_file2[only_in_file2["_merge"] ==
"left_only"].drop(columns=["_merge"])

    return common, only_in_file1, only_in_file2

def compare_blocks(file1, file2):
    """
    Compara los bloques (Formatted Blocks List) entre dos archivos y genera
DataFrames con los resultados.
    """
    df1 = read_blocks(file1)
    df2 = read_blocks(file2)

    if df1 is None or df2 is None:
        print("Error al leer uno o ambos archivos de bloques. Verifique los archivos
proporcionados.")
        return None, None, None

    common = pd.merge(df1, df2, on=["Name", "SOBRANTE"], how="inner")
    only_in_file1 = pd.merge(df1, df2, on=["Name", "SOBRANTE"], how="left",
indicator=True)
    only_in_file1 = only_in_file1[only_in_file1["_merge"] ==
"left_only"].drop(columns=["_merge"])
    only_in_file2 = pd.merge(df2, df1, on=["Name", "SOBRANTE"], how="left",
indicator=True)
    only_in_file2 = only_in_file2[only_in_file2["_merge"] ==
"left_only"].drop(columns=["_merge"])

    return common, only_in_file1, only_in_file2

def save_report(common_connections, only_in_file1, only_in_file2, common_blocks,
only_blocks_file1, only_blocks_file2, output_file):
    """
    Genera un archivo Excel con el reporte de las conexiones y bloques comparados.
    """
    try:
        with pd.ExcelWriter(output_file, engine='openpyxl') as writer:
```

```
sheet_name="Only in PLANO", index=False)
            only_in_file2.drop(columns=["Connection"]).to_excel(writer,
sheet_name="Only in ULD", index=False)

            # Bloques
            common_blocks.to_excel(writer, sheet_name="Matching Blocks",
index=False)
            only_blocks_file1.to_excel(writer, sheet_name="Only Blocks PLANO",
index=False)
            only_blocks_file2.to_excel(writer, sheet_name="Only Blocks ULD",
index=False)

            # Resumen
            summary_data = {
                "Category": ["Matching Connections", "Only in PLANO (Connections)",
"Only in ULD (Connections)",
                             "Matching Blocks", "Only in PLANO (Blocks)", "Only in
ULD (Blocks)"],
                "Count": [
                    len(common_connections), len(only_in_file1), len(only_in_file2),
                    len(common_blocks), len(only_blocks_file1),
len(only_blocks_file2)
                ]
            }
            pd.DataFrame(summary_data).to_excel(writer, sheet_name="Summary",
index=False)

        # Ajustar el ancho de las columnas
        from openpyxl import load_workbook

        workbook = load_workbook(output_file)
        for sheet_name in workbook.sheetnames:
            sheet = workbook[sheet_name]
            for column in sheet.columns:
                max_length = 0
                column_letter = column[0].column_letter  # Obtener la letra de la
columna
                for cell in column:
                    try:
                        if cell.value:  # Calcular la longitud del contenido
                            max_length = max(max_length, len(str(cell.value)))
                    except:
                        pass
                adjusted_width = max_length + 2  # Agregar espacio adicional
                sheet.column_dimensions[column_letter].width = adjusted_width
        workbook.save(output_file)

        print(f"Reporte generado: {output_file}")
    except Exception as e:
        print(f"Error al guardar el reporte: {e}")

def main():
    """
    Función principal para comparar conexiones y bloques entre dos archivos.
    """
    # Usar Tkinter para seleccionar los archivos
    Tk().withdraw()  # Ocultar la ventana principal de Tkinter

    print("Seleccione el archivo de conexiones PLANO:")
    plano_file = askopenfilename(title="Seleccione el archivo de conexiones PLANO",
filetypes=[("Archivos Excel", "*.xlsx")])
    if not plano_file:
        print("No se seleccionó el archivo de conexiones PLANO. Saliendo...")
        return

    print("Seleccione el archivo de conexiones ULD:")
    uld_file = askopenfilename(title="Seleccione el archivo de conexiones ULD",
filetypes=[("Archivos Excel", "*.xlsx")])
    if not uld_file:
        print("No se seleccionó el archivo de conexiones ULD. Saliendo...")
        return

    print("Seleccione dónde guardar el reporte:")
    output_file = asksaveasfilename(title="Guardar reporte como",
defaultextension=".xlsx",
                                    filetypes=[("Archivos Excel", "*.xlsx")])
    if not output_file:
        print("No se seleccionó la ubicación del reporte. Saliendo...")
        return

    # Comparar conexiones
    common_connections, only_in_file1, only_in_file2 =
compare_connections(plano_file, uld_file)

    # Comparar bloques
    common_blocks, only_blocks_file1, only_blocks_file2 = compare_blocks(plano_file,
uld_file)

    if common_connections is not None and common_blocks is not None:
        save_report(common_connections, only_in_file1, only_in_file2,
                    common_blocks, only_blocks_file1, only_blocks_file2,
output_file)
    else:
        print("No se generó el reporte debido a errores en la comparación.")

# Ejecutar el script
main()
```