CONFIANZA 23
Inteligencia y Seguridad

Plataforma SCADA23

Documento de Ingeniería de Detalle

FECHA: 7 de junio de 2025

CONTACTO

Rafael Ausejo Prieto rafael.ausejo@confianza23.es

ÍNDICE

1 Documento de Ingeniería de Detalle

1.1 Introducción

Este documento detalla los requisitos funcionales y no funcionales para el desarrollo de la plataforma SCADA23.

SCADA23 actuará como laboratorio de pruebas y simulación de PLCs, monitorizará y leerá tráfico de PLCs reales y funcionará como un sistema SCADA para la supervisión, control y adquisición de datos de sensores y actuadores reales.

La fase inicial se centrará en el soporte del protocolo Modbus/TCP y el sector de Agua Potable, permitiendo la extensión a otros protocolos y sectores mediante configuración.

El primer módulo implementado por defecto será el del sector de Agua Potable, simulando un sistema de tres tanques (Tank1, Tank2, Tank3) con sus respectivos sensores y actuadores.

Debido a las limitaciones de tiempo, se realizará una versión limitada de la metodología expuesta para poder disponer de una primera versión funcional lo antes posible, en un fin de semana, realizando uso intensivo de la IA.

El estado actual del proyecto se encuentra en la siguienteURL:
https://github.com/rausejop/SCADA23

Debido a que se trata de un proyecto Open Source, se permite y promueve la colaboración en dicho proyecto, si bien se ruega ponerse en contacto con la coordinación del proyecto para alinear las contribuciones a la situación en cada momento.

1.2 Requisitos Funcionales

Se muestran a continuación los requerimientos funcionales

RFUNC001: Simulación y Emulación de PLCs

- **Descripción:** La plataforma SCADA23 debe ser capaz de simular y emular el comportamiento de PLCs. Esto incluye la inyección de tráfico Modbus/TCP simulando lecturas de sensores y respuestas a comandos de actuadores, utilizando la librería Scapy.
- **Comentarios:** Se espera que los scripts de simulación sean modulares, permitiendo la fácil adición de nuevos tipos de PLCs o comportamientos.

RFUNC002: Monitorización de Tráfico PLC Real

- **Descripción:** SCADA23 debe poder capturar y analizar el tráfico Modbus/TCP de PLCs reales del mercado para su monitorización.
- **Comentarios:** La captura de tráfico debe ser no intrusiva y la plataforma debe ser capaz de interpretar y visualizar los datos Modbus/TCP de manera significativa.

RFUNC003: Sistema SCADA para Supervisión y Control

- **Descripción:** La plataforma debe proporcionar una interfaz gráfica de usuario (HMI) para la supervisión, control y adquisición de datos de sensores y actuadores. Esta interfaz se construirá utilizando tcl/tk y representará un Diagrama de Tuberías e Instrumentación (P&ID).
- **Comentarios:** El HMI debe ser intuitivo y mostrar claramente el estado de los sensores y actuadores, permitiendo la interacción con los actuadores.

RFUNC004: Soporte de Protocolos Industriales

- **Descripción:** En esta primera fase, la plataforma debe implementar y soportar completamente el protocolo Modbus/TCP para la comunicación con PLCs simulados y reales.
- **Comentarios:** Aunque se mencionan otros protocolos (PROFINET, S7, OPC-UA), su implementación se pospondrá para fases posteriores. La arquitectura debe ser extensible para permitir su futura integración.

RFUNC005: Configuración de Sectores Industriales

- **Descripción:** La plataforma debe cargar por defecto el módulo para el sector "Agua Potable" y permitir la carga de otros sectores predefinidos a través de un archivo de configuración.
- **Comentarios:** El archivo de configuración debe ser de fácil edición y permitir la especificación de los parámetros relevantes para cada sector (e.g., tipos de sensores, actuadores, rangos, etc.).

1.3 Requisitos No Funcionales

RNFUNC001: Tecnología de Desarrollo

- **Descripción:** La plataforma debe ser desarrollada íntegramente en Python 3.13.4 y ejecutarse en Windows 11. La comunicación con PLCs (simulados o reales) debe realizarse mediante scripts Python que utilicen la librería Scapy para la inyección y lectura de tráfico Modbus/TCP. La interfaz gráfica se desarrollará con tcl/tk.
- **Comentarios:** Se debe asegurar la compatibilidad con las versiones especificadas de Python y Windows. El uso de Scapy debe ser eficiente para evitar latencias significativas.

RNFUNC002: Modularidad del Código

- **Descripción:** El código fuente debe ser modular, bien estructurado y documentado, permitiendo el desarrollo en solitario o en equipo.
- **Comentarios:** Se deben seguir las mejores prácticas de codificación Python (PEP 8), utilizar comentarios claros y dividir el código en funciones y clases lógicas.

RNFUNC003: Rendimiento

- **Descripción:** La plataforma debe ser capaz de procesar y visualizar datos Modbus/TCP en tiempo real con una latencia mínima para garantizar un control y supervisión efectivos.
- **Comentarios:** Se deben realizar pruebas de rendimiento para asegurar que la actualización de los datos en el HMI y la respuesta a los comandos de actuadores sea fluida y sin retrasos perceptibles.

RNFUNC004: Escalabilidad

- **Descripción:** La arquitectura de la plataforma debe ser escalable para permitir la futura integración de nuevos protocolos industriales, tipos de sensores, actuadores y sectores.
- **Comentarios:** Se recomienda un diseño que desacople las capas de comunicación, lógica de negocio y presentación para facilitar futuras expansiones.

RNFUNC005: Seguridad

- **Descripción:** La plataforma debe estar diseñada con consideraciones de ciberseguridad industrial, minimizando las vulnerabilidades.
- **Comentarios:** Aunque Modbus/TCP no incluye seguridad intrínseca, se deben implementar las mejores prácticas de desarrollo seguro a nivel de aplicación para proteger la integridad y disponibilidad de los datos y el control. Se debe considerar la segregación de la red y el uso de firewalls.

2 Proceso de Sistema de Agua Potable

2.1 Descripción Funcional del Proceso (FDS - Functional Design Specification)

El sistema de agua potable simulará el llenado y vaciado de tres tanques, con la posibilidad de monitorizar sus parámetros (presión, temperatura, flujo y nivel) y controlar sus válvulas de entrada y salida.

a. Rangos Operativos Normales

- **Presión:** 2 a 6 bar
- **Temperatura:** 15 a 30 °C
- **Flujo:** 30 a 70 L/min
- **Nivel:** 20% a 80%

b. Puntos de Ajuste para Alarmas y Control

- **Alarma de Presión Alta:** > 8 bar
- **Alarma de Presión Baja:** < 1 bar
- **Alarma de Temperatura Alta:** > 90 °C
- **Alarma de Temperatura Baja:** < 5 °C
- **Alarma de Flujo Alto:** > 95 L/min
- **Alarma de Flujo Bajo:** < 10 L/min
- **Alarma de Nivel Alto:** > 95% (Detener llenado, abrir salida si es necesario)
- **Alarma de Nivel Bajo:** < 5% (Iniciar llenado, cerrar salida si es necesario)
- **Control de Válvulas:** El usuario podrá abrir/cerrar las válvulas de entrada y salida de cada tanque desde el HMI.

2.2 Hojas de Datos de Instrumentos

A continuación, se detallan las especificaciones para los sensores y actuadores del sistema de Agua Potable. Estos datos servirán como base para la simulación y la representación en el P&ID.

a. Sensores de Presión (PS/PT)

- **Identificadores:** PS-101, PT-101 (Tanque 1); PS-102, PT-102 (Tanque 2); PS-103, PT-103 (Tanque 3)
- **Descripción:** Mide la presión del líquido dentro del tanque.
- **Rango de Medición:** 0 a 10 bar
- **Unidades:** bar
- **Punto de Ajuste (Setpoint):** Presión máxima para alarma (ej. 8 bar)
- **Precisión:** ±0.5% del fondo de escala
- **Conexión:** Modbus/TCP Holding Register

b. Sensores de Temperatura (TS)

- **Identificadores:** TS-101 (Tanque 1); TS-102 (Tanque 2); TS-103 (Tanque 3)
- **Descripción:** Mide la temperatura del líquido dentro del tanque.
- **Rango de Medición:** 0 a 100 °C
- **Unidades:** °C
- **Punto de Ajuste (Setpoint):** Temperatura mínima/máxima para alarma (ej. 5 °C, 90 °C)
- **Precisión:** ±1°C
- **Conexión:** Modbus/TCP Holding Register

c. Sensores de Flujo (FS)

- **Identificadores:** FS-101 (Tanque 1); FS-102 (Tanque 2); FS-103 (Tanque 3)
- **Descripción:** Mide el caudal de entrada o salida del tanque.
- **Rango de Medición:** 0 a 100 L/min
- **Unidades:** L/min
- **Punto de Ajuste (Setpoint):** Caudal mínimo/máximo para alarma (ej. 10 L/min, 95 L/min)
- **Precisión:** ±1% del fondo de escala
- **Conexión:** Modbus/TCP Holding Register

d. Sensores de Nivel (LS)

- **Identificadores:** LS-101 (Tanque 1); LS-102 (Tanque 2); LS-103 (Tanque 3)
- **Descripción:** Mide el nivel de llenado del tanque.
- **Rango de Medición:** 0% a 100%
- **Unidades:** Porcentaje (%)
- **Punto de Ajuste (Setpoint):** Nivel mínimo/máximo para alarma (ej. 5%, 95%)
- **Precisión:** ±1%

- **Conexión:** Modbus/TCP Holding Register

e. Válvulas de Entrada (XV)

- **Identificadores:** XV-101A (Tanque 1); XV-102A (Tanque 2); XV-103A (Tanque 3)
- **Descripción:** Actuador On/Off que controla el flujo de entrada al tanque.
- **Estado:** Abierta (True) / Cerrada (False)
- **Tiempo de Apertura/Cierre:** Instantáneo para simulación (en la realidad, podría ser milisegundos).
- **Conexión:** Modbus/TCP Coil

f. Válvulas de Salida (XV)

- **Identificadores:** XV-101B (Tanque 1); XV-102B (Tanque 2); XV-103B (Tanque 3)
- **Descripción:** Actuador On/Off que controla el flujo de salida del tanque.
- **Estado:** Abierta (True) / Cerrada (False)
- **Tiempo de Apertura/Cierre:** Instantáneo para simulación.
- **Conexión:** Modbus/TCP Coil

g. Tanques de Agua Potable

- **Identificadores:** T-101 (Tanque 1); T-102 (Tanque 2); T-103 (Tanque 3)
- **Descripción:** Depósitos de almacenamiento de agua potable.
- **Volumen/Capacidad:** Se definirá en el archivo de configuración por sector (ej. 1000 Litros).

2.3 Componentes

Sistema de Control (SCADA/PLC)

El módulo de Agua Potable en SCADA23 se encargará de:

- **Adquisición de Datos:** Leerá los valores de los sensores (presión, temperatura, flujo, nivel) de los PLCs (simulados o reales) a través de Modbus/TCP (Holding Registers).
- **Visualización:** Los valores adquiridos se mostrarán en tiempo real en el HMI (P&ID) de forma gráfica. Se utilizarán indicadores visuales para el estado de las válvulas y el nivel de los tanques.
- **Control:** Enviará comandos a los actuadores (válvulas de entrada y salida) a los PLCs (simulados o reales) a través de Modbus/TCP (Coils) para cambiar su estado (abrir/cerrar).
- **Alertas:** Detectará y mostrará alarmas cuando los valores de los sensores superen o caigan por debajo de los puntos de ajuste predefinidos.
- **Registro de Datos:** La plataforma debería tener la capacidad básica de registrar los datos de los sensores a lo largo del tiempo, aunque esto podría ser una mejora futura.
- **Configuración:** Los rangos de medición, unidades y puntos de ajuste se cargarán desde el archivo de configuración del sector, permitiendo una fácil adaptación.

3 Planificación del Desarrollo (Metodología SCRUM)

Se detalla la planificación para el desarrollo de la plataforma SCADA23, siguiendo una metodología ágil SCRUM. El equipo de desarrollo estará compuesto por tres personas. La fase inicial se centrará en el módulo de Agua Potable y la comunicación Modbus/TCP.

4.1. Roles del Equipo

- **Product Owner:** Rafael Ausejo Prieto (contacto: rafael.ausejo@confianza23.es) - Responsable de la visión del producto y la priorización del Backlog.
- **Scrum Master:** [Nombre del Scrum Master] - Facilitador, elimina impedimentos y asegura la adherencia a la metodología SCRUM. (Rol asignado a este interlocutor para la generación de este documento)
- **Equipo de Desarrollo:**
  - Desarrollador 1: Rafael Ausejo Prieto
  - Desarrollador 2: J.S
  - Desarrollador 3:

4.2. Backlog del Producto

El Backlog del Producto será gestionado y priorizado por el Product Owner. A continuación, se listan algunos ítems iniciales para la primera fase:

- Como usuario, quiero que la plataforma simule el comportamiento de PLCs para pruebas.
- Como usuario, quiero que la plataforma pueda capturar y analizar tráfico Modbus/TCP de PLCs reales.
- Como usuario, quiero una interfaz gráfica (HMI) para supervisar y controlar sensores y actuadores del sistema de agua potable.
- Como usuario, quiero que la plataforma soporte completamente el protocolo Modbus/TCP.
- Como administrador, quiero que la plataforma cargue por defecto el módulo de "Agua Potable" y sea configurable para otros sectores.
- Como desarrollador, quiero que el código sea modular y esté documentado.
- Como usuario, quiero que la HMI muestre claramente el estado de los sensores y actuadores del sistema de agua potable.
- Como usuario, quiero poder controlar las válvulas de entrada y salida de los tanques desde el HMI.
- Como desarrollador, quiero que la plataforma sea desarrollada en Python 3.13.4 y se ejecute en Windows 11, utilizando Scapy y tcl/tk.
- Como usuario, quiero que se visualicen los datos de los sensores (presión, temperatura, flujo, nivel) en tiempo real en el HMI.
- Como usuario, quiero que se detecten y muestren alarmas cuando los valores de los sensores superen o caigan por debajo de los puntos de ajuste.
- Como desarrollador, quiero que los rangos de medición y puntos de ajuste se carguen desde un archivo de configuración.
- Como usuario, quiero simular el llenado y vaciado de tres tanques de agua potable.

4.3. Hitos de Desarrollo (Fase 1: Módulo Agua Potable)

Se proponen los siguientes hitos para la fase inicial del proyecto:

- **Hito 1 (Sprint 1-2): Configuración y Simulación Básica de PLC**
  - Definición del archivo de configuración inicial para Agua Potable.
  - Desarrollo del esqueleto del PLC simulado con Modbus/TCP y manejo de Holding Registers y Coils.
  - Simulación de lecturas básicas de sensores (presión, temperatura, flujo, nivel) y control de válvulas (On/Off).
  - Validación de la comunicación Modbus/TCP entre PLC simulado y un cliente de prueba.
- **Hito 2 (Sprint 3-4): HMI Básico y Visualización de Datos**
  - Implementación de la estructura básica del HMI con tcl/tk.
  - Conexión del HMI con el PLC simulado para la adquisición de datos de sensores.
  - Visualización en tiempo real de los valores de los sensores en el HMI.
  - Representación visual básica del estado de las válvulas y el nivel de los tanques.
- **Hito 3 (Sprint 5-6): Control y Alarmas**
  - Implementación de la funcionalidad de control de válvulas desde el HMI.
  - Desarrollo del módulo de detección y visualización de alarmas (presión, temperatura, flujo, nivel).
  - Integración de los puntos de ajuste y rangos operativos desde el archivo de configuración.
  - Mejora de la interactividad del HMI.
- **Hito 4 (Sprint 7): Refinamiento y Documentación**
  - Refinamiento del código y aseguramiento de la modularidad y documentación (PEP 8).
  - Pruebas de rendimiento y optimización para baja latencia.
  - Revisión de requisitos no funcionales (escalabilidad, seguridad).
  - Generación de la documentación técnica y de usuario inicial.

4.4. Tareas Detalladas y Nombres de Ficheros Python (.py)

A continuación, se presenta un desglose de tareas por componentes, incluyendo los nombres de ficheros Python sugeridos.

4.4.1. Módulo de Configuración y Utilidades

- **Tarea:** Definir la estructura y cargar el archivo de configuración (config.json o config.ini).
  - **Ficheros:** config_manager.py (Clase para cargar/manejar la configuración).
- **Tarea:** Implementar funciones utilitarias generales.
  - **Ficheros:** utils.py

4.4.2. PLC Simulado (Simulador Modbus/TCP)

Este componente será la base para la simulación de los tanques de agua.

- **Tarea:** Crear la clase base para un dispositivo Modbus/TCP (manejo de Holding Registers y Coils).
  - **Ficheros:** modbus_device.py
- **Tarea:** Implementar la lógica específica de los tanques de agua (T-101, T-102, T-103) con sus sensores y actuadores.
  - **Ficheros:** tank_simulator.py (Contendrá la lógica de llenado/vaciado, actualización de valores de sensores).
- **Tarea:** Crear scripts para simular la inyección de tráfico Modbus/TCP utilizando Scapy.
  - **Ficheros:** plc_emulator.py (Orquestador de los tanques simulados y el servidor Modbus/TCP).
  - sensor_data_generator.py (Módulo para generar datos realistas de sensores).

4.4.3. Consola SCADA (Cliente Modbus/TCP y HMI)

Este componente será la interfaz de usuario para supervisión y control.

- **Tarea:** Desarrollar el cliente Modbus/TCP para leer Holding Registers y escribir Coils.
  - **Ficheros:** modbus_client.py
- **Tarea:** Implementar la interfaz gráfica de usuario (HMI) utilizando tcl/tk.
  - **Ficheros:** scada_hmi.py (Contendrá la lógica principal de la interfaz gráfica).
  - hmi_elements.py (Clases para los elementos gráficos: tanques, válvulas, indicadores de sensores).
  - alarm_manager.py (Lógica para detectar y mostrar alarmas).
- **Tarea:** Integrar la adquisición de datos y el control de actuadores en el HMI.
  - **Ficheros:** data_acquisition.py (Módulo para gestionar la lectura periódica de datos del PLC).
  - control_logic.py (Módulo para gestionar el envío de comandos al PLC).

- **Tarea:** Lógica para monitorizar y analizar tráfico de PLCs reales (captura no intrusiva). (Considerado para futuras iteraciones en detalle, pero la base estará en el cliente).
    - **Ficheros:** traffic_monitor.py (Utilizará Scapy para la captura y análisis).

4.4.4. Punto de Entrada Principal

- **Tarea:** Crear los scripts principales para iniciar la simulación del PLC y la consola SCADA.
    - **Ficheros:** main_plc_simulator.py (Ejecuta el PLC simulado).
    - main_scada_console.py (Ejecuta la consola SCADA).

4.5. Consideraciones Adicionales

- **Control de Versiones:** Se utilizará Git para el control de versiones
- **Pruebas:** Se implementarán pruebas unitarias y de integración para asegurar la calidad del código.
- **Reuniones SCRUM:**
    - **Daily Stand-ups:** Reuniones diarias de 15 minutos para sincronización.
    - **Sprint Planning:** Al inicio de cada Sprint para planificar las tareas.
    - **Sprint Review:** Al final de cada Sprint para revisar el incremento.
    - **Sprint Retrospective:** Al final de cada Sprint para identificar mejoras en el proceso.
- **Herramientas:** Se utilizarán entornos de desarrollo compatibles con Python (e.g., VS Code, PyCharm).
- **Documentación:** La documentación del código se realizará de forma continua, siguiendo las mejores prácticas de Python.

4 Otros Sectores Documento de Ingeniería de Detalle

Tras la implementación del primer sector de agua potable, se impementarán el resto de sectores soportados:

Categoría	Código de Sector	Descripción del Sector
Energía	SAC01	Energía (General)
	SAC01a	Electricidad
	SAC01b	Sistemas urbanos de calefacción y refrigeración
	SAC01c	Crudo
	SAC01d	Gas
	SAC01e	Hidrógeno
Transporte	SAC02	Transporte (General)
	SAC02a	Transporte aéreo
	SAC02b	Transporte por ferrocarril
	SAC02c	Transporte marítimo y fluvial
	SAC02d	Transporte por carretera
Banca	SAC03	Banca
Finanzas	SAC04	Infraestructura de los mercados financieros
Sanidad	SAC05	Sector sanitario
Agua Potable	SAC06	Agua potable (Módulo inicial implementado por defecto)
Aguas Residuales	SAC07	Aguas residuales
Infraestructura digital	SAC08	Infraestructura digital
Servicios TIC	SAC09	Gestión de servicios de TIC (de empresa a empresa)
AGE	SAC10	Entidades de la Administración pública (excl. judicial, parlamentos y bancos centrales)
Espacio	SAC11	Espacio
Nuclear	SAC12	Industria Nuclear
Otros Sectores	OTR01	Servicios postales y de mensajería
	OTR02	Gestión de residuos

	OTR03	Fabricación, producción y distribución de sustancias y mezclas químicas
	OTR04	Producción, transformación y distribución de alimentos
Fabricación	OTR05	Fabricación (General)
	OTR5a	Fabricación de productos sanitarios y productos sanitarios para diagnóstico in vitro
	OTR5b	Fabricación de productos informáticos, electrónicos y ópticos
	OTR5c	Fabricación de material eléctrico
	OTR5d	Fabricación de maquinaria y equipo n.c.o.p.
	OTR5e	Fabricación de vehículos de motor, remolques y semirremolques
	OTR5f	Fabricación de otro material de transporte
	OTR06	Proveedores de servicios digitales
	OTR07	Investigación
	OTR08	Seguridad Privada

5 Fichero README.md en Github

SCADA23: Plataforma de Inteligencia y Seguridad Industrial 🛡️

  <img
src="https://github.com/rausejop/SCADA23/blob/main/docs/img/confianza23_logo.png?raw=true" alt="CONFIANZA 23 Logo" width="200"/>

📄 Introducción

Este repositorio contiene la **Plataforma SCADA23**, un entorno de laboratorio y simulación diseñado para la supervisión, control y adquisición de datos en sistemas SCADA/PLC. Su propósito principal es simular y emular el comportamiento de PLCs, monitorizar el tráfico de PLCs reales y funcionar como un sistema SCADA completo.

La fase inicial de desarrollo se enfoca en el soporte del protocolo **Modbus/TCP** y el sector de **Agua Potable**, con una arquitectura extensible para futuras integraciones. El primer módulo implementado simula un sistema de tres tanques (Tank1, Tank2, Tank3) con sus respectivos sensores y actuadores.

Este proyecto es de **código abierto** y se fomenta la colaboración. Si deseas contribuir, por favor, contacta con la coordinación del proyecto para alinear tus aportaciones.

✨ Características Principales (Fase 1)

* **Simulación y Emulación de PLCs**: Capacidad para simular y emular el comportamiento de PLCs, inyectando tráfico Modbus/TCP para lecturas de sensores y respuestas a actuadores utilizando la librería Scapy.
* **Monitorización de Tráfico PLC Real**: Captura y análisis no intrusivo del tráfico Modbus/TCP de PLCs reales, con visualización significativa de los datos.
* **Sistema SCADA con HMI**: Interfaz gráfica de usuario (HMI) desarrollada con `tcl/tk` para la supervisión, control y adquisición de datos, representando un Diagrama de Tuberías e Instrumentación (P&ID).
* **Soporte Modbus/TCP**: Implementación completa del protocolo Modbus/TCP para la comunicación con PLCs simulados y reales.
* **Configuración de Sectores Industriales**: Carga por defecto del módulo de "Agua Potable" y capacidad de configurar otros sectores predefinidos a través de un archivo de configuración.
* **Desarrollo en Python**: Construido íntegramente en Python 3.13.4 y diseñado para ejecutarse en Windows 11.

🚀 Puesta en Marcha (Fase 1: Módulo Agua Potable)

📋 Prerrequisitos

* Python 3.13.4
* Sistema Operativo: Windows 11
* Librerías Python: `Scapy`, `tcl/tk` (asegúrate de que `tcl/tk` esté disponible con tu instalación de Python).

⚙️ Instalación

1. Clona el repositorio:
   ```bash
   git clone [https://github.com/rausejop/SCADA23.git](https://github.com/rausejop/SCADA23.git)
   cd SCADA23
   ```
   El estado actual del proyecto se encuentra en `https://github.com/rausejop/SCADA23`.
2. Crea un entorno virtual (opcional pero recomendado):
   ```bash
   python -m venv venv
   .\venv\Scripts\activate  # En Windows
   source venv/bin/activate # En Linux/macOS
   ```
3. Instala las dependencias:
   ```bash
   pip install scapy
   # Puede que necesites instalar tk/tcl por separado si no viene con tu distribución de Python
   ```

🏃 Ejecución

Para iniciar la simulación y la consola SCADA, ejecuta los siguientes scripts desde la raíz del proyecto:

1. Iniciar el PLC simulado:
   ```bash
   python src/main_plc_simulator.py
   ```
2. Iniciar la consola SCADA (HMI):
   ```bash
   python src/main_scada_console.py
   ```

```

## 📂 Estructura de Ficheros del Proyecto

La estructura del proyecto está diseñada para ser modular y extensible.

Me disculpo por el error. El código `` es un marcador interno que no debería ser visible en la salida final. Ha sido un fallo en la generación del formato.

El objetivo era aplicar las pautas de citación internas del modelo, pero de forma incorrecta, se incluyó ese marcador visible.

He corregido el README.md para eliminar cualquier instancia de `` o marcadores de citación inesperados, presentando el contenido limpio como se espera en un archivo README de GitHub. Aquí tienes la versión corregida:

Markdown

# <p align="center">SCADA23: Plataforma de Inteligencia y Seguridad Industrial 🛡️</p>

<p align="center">
  <img
src="https://github.com/rausejop/SCADA23/blob/main/docs/img/confianza23_logo.png?raw=true" alt="CONFIANZA 23 Logo" width="200"/>
</p>

## 📄 Introducción

Este repositorio contiene la **Plataforma SCADA23**, un entorno de laboratorio y simulación diseñado para la supervisión, control y adquisición de datos en sistemas SCADA/PLC. Su propósito principal es simular y emular el comportamiento de PLCs, monitorizar el tráfico de PLCs reales y funcionar como un sistema SCADA completo.

La fase inicial de desarrollo se enfoca en el soporte del protocolo **Modbus/TCP** y el sector de **Agua Potable**, con una arquitectura extensible para futuras integraciones. El primer módulo implementado simula un sistema de tres tanques (Tank1, Tank2, Tank3) con sus respectivos sensores y actuadores.

Este proyecto es de **código abierto** y se fomenta la colaboración. Si deseas contribuir, por favor, contacta con la coordinación del proyecto para alinear tus aportaciones.

## ✨ Características Principales (Fase 1)
```

* **Simulación y Emulación de PLCs**: Capacidad para simular y emular el comportamiento de PLCs, inyectando tráfico Modbus/TCP para lecturas de sensores y respuestas a actuadores utilizando la librería Scapy.
* **Monitorización de Tráfico PLC Real**: Captura y análisis no intrusivo del tráfico Modbus/TCP de PLCs reales, con visualización significativa de los datos.
* **Sistema SCADA con HMI**: Interfaz gráfica de usuario (HMI) desarrollada con `tcl/tk` para la supervisión, control y adquisición de datos, representando un Diagrama de Tuberías e Instrumentación (P&ID).
* **Soporte Modbus/TCP**: Implementación completa del protocolo Modbus/TCP para la comunicación con PLCs simulados y reales.
* **Configuración de Sectores Industriales**: Carga por defecto del módulo de "Agua Potable" y capacidad de configurar otros sectores predefinidos a través de un archivo de configuración.
* **Desarrollo en Python**: Construido íntegramente en Python 3.13.4 y diseñado para ejecutarse en Windows 11.

🚀 Puesta en Marcha (Fase 1: Módulo Agua Potable)

📋 Prerrequisitos

* Python 3.13.4
* Sistema Operativo: Windows 11
* Librerías Python: `Scapy`, `tcl/tk` (asegúrate de que `tcl/tk` esté disponible con tu instalación de Python).

⚙️ Instalación

1. Clona el repositorio:
   ```bash
   git clone [https://github.com/rausejop/SCADA23.git](https://github.com/rausejop/SCADA23.git)
   cd SCADA23
   ```
   El estado actual del proyecto se encuentra en `https://github.com/rausejop/SCADA23`.
2. Crea un entorno virtual (opcional pero recomendado):
   ```bash
   python -m venv venv
   .\venv\Scripts\activate  # En Windows
   source venv/bin/activate # En Linux/macOS
   ```
3. Instala las dependencias:
   ```bash
   pip install scapy
   ```

Puede que necesites instalar tk/tcl por separado si no viene con tu distribución de Python
```

### 🏃 Ejecución

Para iniciar la simulación y la consola SCADA, ejecuta los siguientes scripts desde la raíz del proyecto:

1. Iniciar el PLC simulado:
   ```bash
   python src/main_plc_simulator.py
   ```
2. Iniciar la consola SCADA (HMI):
   ```bash
   python src/main_scada_console.py
   ```

📂 Estructura de Ficheros del Proyecto

La estructura del proyecto está diseñada para ser modular y extensible.

SCADA23/
├── src/
│ ├── config_manager.py # 🔧 Clase para cargar y manejar la configuración del sistema (e.g., config.json).
│ ├── utils.py # 🛠️ Funciones utilitarias generales para el proyecto.
│ │
│ ├── modbus_device.py # 🔌 Clase base para un dispositivo Modbus/TCP (manejo de Holding Registers y Coils).
│ ├── tank_simulator.py # 💧 Lógica específica de los tanques de agua (T-101, T-102, T-103), incluyendo llenado/vaciado y actualización de sensores.
│ ├── plc_emulator.py # 🤖 Orquestador de los tanques simulados y el servidor Modbus/TCP para emulación de PLC.
│ ├── sensor_data_generator.py # 📊 Módulo para generar datos realistas de sensores para la simulación.
│ │
│ ├── modbus_client.py # 📡 Cliente Modbus/TCP para leer Holding Registers y escribir Coils en PLCs (simulados o reales).
│ ├── scada_hmi.py # 🖥️ Lógica principal de la interfaz gráfica de usuario (HMI) construida con tcl/tk.
   ```

```
│   ├── hmi_elements.py       # 🎨 Clases para los elementos gráficos de la HMI: tanques, válvulas, indicadores de sensores.
│   ├── alarm_manager.py      # 🚨 Lógica para detectar y mostrar alarmas.
│   ├── data_acquisition.py   # 📈 Módulo para gestionar la lectura periódica de datos del PLC.
│   ├── control_logic.py      # 🕹️ Módulo para gestionar el envío de comandos al PLC.
│   ├── traffic_monitor.py    # 🕵️ Módulo para la captura y análisis no intrusivo de tráfico Modbus/TCP de PLCs reales (utiliza Scapy).
│   │
│   ├── main_plc_simulator.py # ▶️ Script principal para iniciar el PLC simulado.
│   └── main_scada_console.py # ▶️ Script principal para iniciar la consola SCADA.
│
├── docs/                     # 📚 Carpeta para la documentación del proyecto.
│   └── img/                  # 🖼️ Imágenes utilizadas en la documentación (e.g., logos).
│
├── config/
│   └── water_system.json     # ⚙️ Archivo de configuración por defecto para el módulo de Agua Potable.
│
└── README.md                 # 📖 Este mismo archivo.
```

🤝 Colaboración

¡Agradecemos cualquier contribución! Si tienes ideas o quieres participar, por favor:

1. Ponte en contacto con el Product Owner: **Rafael Ausejo Prieto** (rafael.ausejo@confianza23.es).
2. Revisa el Backlog del Producto para las tareas priorizadas.
3. Sigue las mejores prácticas de codificación Python (PEP 8) y asegura una buena documentación del código.

Utilizamos **Git** para el control de versiones y seguimos una metodología **SCRUM** con Daily Stand-ups, Sprint Planning, Sprint Review y Sprint Retrospective.

📞 Contacto

Rafael Ausejo Prieto
* **Email**: rafael.ausejo@confianza23.es
* **Organización**: CONFIANZA 23 Inteligencia y Seguridad

📜 Licencia

Este proyecto es de código abierto.

6 Siguientes pasos

- Generar el fichero main_plc_simulator.py
- Generar el fichero main_scada_console.py