Escuela de Doctorado
y Estudios de Posgrado
Universidad de La Laguna

Máster Universitario en
Ingeniería Industrial

Trabajo Fin de Máster

Diseño e implementación de un sistema SCADA para un prototipo físico para la simulación y detección de fugas en depósitos de combustible

Autor/a: Alexander Epifanio Corona Ledesma

Tutor/a: Marta Sigut Saavedra

Cotutor/a: Pedro Antonio Toledo Delgado

7 de julio de 2024

La publicación de este Trabajo Fin de Máster solo implica que el estudiante ha obtenido al menos la nota mínima exigida para superar la asignatura correspondiente, no presupone que su contenido sea correcto, aunque si aplicable. En este sentido, la ULL no posee ningún tipo de responsabilidad hacia terceros por la aplicación total o parcial de los resultados obtenidos en este trabajo. También pone en conocimiento del lector que, según la ley de protección intelectual, los resultados son propiedad intelectual del alumno, siempre y cuando se haya procedido a los registros de propiedad intelectual o solicitud de patentes correspondientes con fecha anterior a su publicación.

Índice General

I. MEMORIA

Índice

Índice de Figuras

Resumen

Este proyecto de Trabajo de Fin de Máster (TFM) se centra en el diseño e implementación de un sistema SCADA aplicado a un prototipo físico que simula una estación de servicio, con el objetivo de detectar fugas en depósitos de combustible. Este trabajo es una continuación de varios proyectos anteriores, donde se ha construido y mejorado el prototipo inicial.

El sistema desarrollado permite la supervisión en tiempo real del estado de los depósitos de combustible y proporciona una capacidad de respuesta ante el funcionamiento del sistema. Para lograrlo, se ha creado una interfaz de usuario utilizando la librería Dash en Python, que incluye un panel de control y un panel de monitorización. Estas interfaces facilitan la interacción y la visualización de datos tanto en tiempo real como históricos, mejorando la capacidad de análisis y toma de decisiones.

La comunicación entre el sistema SCADA y el prototipo físico se ha establecido mediante el microcontrolador Arduino del sistema, utilizando comunicación serial y el protocolo de red MQTT. Esta configuración ha demostrado ser efectiva para el propósito del prototipo, ofreciendo una solución robusta y flexible. Además, el sistema tiene un potencial uso didáctico significativo, permitiendo a estudiantes del área de la automatización y control adquirir conocimientos prácticos sobre la implementación y operación de sistemas SCADA.

El proyecto también incluye el desarrollo de mecanismos para el almacenamiento de datos, lo que permite la conservación y el análisis de datos a largo plazo. Esta funcionalidad es crucial para identificar patrones y tendencias significativas de este tipo de sistemas.

Abstract

This Master's Thesis Project (TFM) focuses on the design and implementation of a SCADA system applied to a prototype simulating a petrol station, with the aim of detecting leaks in fuel tanks. This project is a continuation of several previous projects, where the initial prototype has been built and improved.

The developed system allows real-time monitoring of the status of fuel tanks and provides a response capability to the system's operation. To achieve this, a user interface has been created using the Dash library in Python, which includes a control panel and a monitoring panel. These interfaces facilitate interaction and data visualization, both in real-time and historical data, enhancing analysis and decision-making capabilities.

Communication between the SCADA system and the physical prototype has been established through the system's Arduino controller, using serial communication and the MQTT network protocol. This configuration has proven to be effective for the prototype's purpose, offering a robust and flexible solution.

Additionally, the system has significant educational potential, allowing students in the field of automation and control to acquire practical knowledge about the implementation and operation of SCADA systems.

The project also includes the development of mechanisms for data storage, allowing for the long-term preservation and analysis of data. This functionality is crucial for identifying significant patterns and trends in such systems.

1. Introducción

1.1 Marco teórico

La detección de fugas en depósitos de combustible es fundamental para la operación segura y eficiente de estaciones de servicio y otras instalaciones que almacenan hidrocarburos. Las fugas pueden causar pérdidas económicas significativas, así como daños ambientales y riesgos para la salud pública. Existen varios métodos y tecnologías utilizados para identificar y gestionar estas fugas, cada uno con sus ventajas y limitaciones.

Los métodos de detección se pueden clasificar en directos e indirectos. Los métodos directos incluyen los Sistemas de Monitoreo Continuo (CMS), que utilizan sensores para vigilar la presencia de hidrocarburos dentro de los tanques en todo momento. Los métodos de inspección visual también son comunes, involucrando revisiones regulares de los depósitos y las áreas circundantes en busca de señales visibles de fugas, como manchas de combustible.

Por otro lado, los métodos indirectos comprenden técnicas basadas en la denominada ‘conciliación de inventario’, a través de los datos de inventario se monitorea las variaciones en el volumen de combustible dentro de los depósitos, comparándolas con las entradas y salidas registradas para identificar discrepancias que puedan indicar una fuga.

En el artículo perteneciente a la Universidad de La Laguna [1], los autores utilizan la conciliación de inventarios junto con la teoría de clasificación de patrones para identificar días operativos en estaciones de servicio como "día sin fugas" o "día con fugas".

En este artículo se estudia la aplicación de cuatro tipos diferentes de clasificadores (dos supervisados y dos no supervisados) para la detección de fugas de combustible en gasolineras.

La investigación avanzó con el artículo que propone el uso de ventanas temporales para mejorar la detección de fugas de combustible en estaciones de servicio [2]. En este estudio, los autores utilizaron clasificadores supervisados y no supervisados, evaluando su eficacia para detectar fugas mediante el análisis de variables extraídas de los libros de inventario.

El trabajo destaca la implementación de ventanas temporales de longitud variable, que permiten acumular y procesar información a lo largo de varios días. Esta técnica no solo facilita la identificación de fugas, sino que también distingue entre fugas ocurridas en un solo día y aquellas acumuladas con el tiempo. Este enfoque permite determinar con precisión la gravedad y urgencia de las fugas detectadas, mejorando significativamente la eficiencia del sistema de detección.

1.2. Antecedentes

Este proyecto se basa en trabajos previos de otros estudiantes enfocados en la detección de fugas de combustible en depósitos de estaciones de servicio.

- TFG "Diseño e implementación de un sistema autónomo para la simulación de fugas en depósitos" [3], del ingeniero Luis Arriaga Campos, donde presenta el diseño inicial de un prototipo de bajo coste de un sistema que simula el proceso de llenado y vaciado de depósitos en una estación de servicio, aunque sin implementación física.

- TFM "Diseño e implementación de un sistema autónomo para la detección de fugas en depósitos" [4], del ingeniero Nicolás Yanes Pérez, llevó a cabo el diseño de un simulador de dicha planta para la generación de datos de un inventario.

- Tanto el TFG "Simulación y detección de fugas en depósitos de combustible" [5], realizado por Fernando Rodríguez Herrera, como el TFG "Prototipo de bajo coste para la simulación de fugas en depósitos de una estación de servicio con fines docentes y de investigación", de Eduardo Miguel Gastón Quesada, llevaron a cabo la implementación física del diseño.

- El punto de partida para este proyecto, es el TFG “Puesta en marcha y mejora de usabilidad de un prototipo físico para la simulación y detección de fugas en depósitos de combustible” [6], de Gregorio José Medina León.

Antes del comienzo de este TFM, el prototipo se encuentra en una fase operativa avanzada gracias a la implementación y mejoras realizadas por los proyectos previos. El trabajo de Luis Arriaga y Nicolás Yanes proporcionó una base teórica y simulada, mientras que Fernando Rodríguez y Eduardo Miguel lograron la implementación física inicial. Finalmente, Gregorio José mejoró la usabilidad y la funcionalidad del prototipo, dejándolo en un estado adecuado para su integración en un sistema SCADA completo.

Este TFM se basa en este prototipo, con el objetivo de diseñar e implementar un sistema SCADA para la simulación y detección de fugas. El sistema permitirá controlar y monitorizar el prototipo, generando datos que se utilizarán como entrada para otros sistemas de detección basados en datos. Esto no solo mejorará la capacidad de detección de fugas, sino que también proporcionará una plataforma robusta para futuros desarrollos y estudios en este campo.

1.3. Objetivo del proyecto

El objetivo principal de este proyecto es diseñar e implementar un sistema SCADA (Supervisory Control and Data Acquisition) que se conecte a un prototipo físico de una estación de servicios para simular y detectar fugas de combustible. Este sistema permitirá supervisar el estado de los depósitos y los modos de funcionamiento en tiempo real.

Además, el sistema proporcionará acceso a datos históricos, permitiendo el análisis y la evaluación a largo plazo de los datos recopilados. Los usuarios también tendrán la capacidad de controlar el sistema desde el SCADA, ajustando parámetros y respondiendo a eventos en tiempo real.

1.4. Alcance del proyecto

El alcance del proyecto incluye los siguientes objetivos:

- **Diseño del sistema SCADA:** Definir la arquitectura del sistema, los componentes necesarios y sus interacciones.
- **Desarrollo de la interfaz de usuario:** Crear un panel de control y un panel de monitorización utilizando la librería Dash en Python.
- **Implementación de la comunicación:** Establecer el protocolo de comunicaciones entre el sistema SCADA y el prototipo físico.
- **Monitoreo y almacenamiento de datos:** Implementar mecanismos para la visualización y almacenamiento de las variables de interés. Esto proporciona un historial de datos y facilita la identificación de patrones y tendencias que puedan indicar problemas potenciales.
- **Pruebas y validación:** Realizar pruebas para asegurar el correcto funcionamiento del sistema y validar su desempeño.

1.5. Metodología de trabajo

Para el desarrollo de este proyecto, se ha seguido una metodología estructurada y sistemática que garantiza la correcta planificación, ejecución y evaluación de cada una de las etapas involucradas. La metodología adoptada se divide en cinco fases principales:

1. Investigación y Recolección de Información

Se estudiaron proyectos anteriores basados en el prototipo, evaluando su funcionamiento y recopilando información relevante para entender mejor los desafíos y soluciones previas. Esto incluyó la revisión detallada de trabajos de otros estudiantes que se centraron en la implementación del prototipo de detección de fugas en depósitos de combustible

2. Diseño del Sistema

En esta fase se definieron los requisitos del sistema, y se elaboró un diseño inicial del sistema SCADA. Esta etapa incluyó:

- **Diseño de Arquitectura**: Se definió la arquitectura del proyecto, identificando los elementos esenciales y su integración en el sistema. Esta fase incluyó la selección de componentes clave y la estructuración general del sistema, asegurando una cohesión y funcionalidad óptimas.

- **Diseño de la Interfaz de Usuario**: Desarrollo de maquetas y prototipos de la interfaz de usuario, teniendo en cuenta aspectos de usabilidad y ergonomía, para asegurar una experiencia de usuario eficiente e intuitiva.

3. Desarrollo e Implementación

En esta fase, se llevó a cabo la programación del panel de control y el panel de monitorización utilizando Dash en Python, así como la

integración de las comunicaciones entre el sistema SCADA y el prototipo físico. Las actividades incluyeron:

- **Desarrollo del Panel de Control**: Implementación de una interfaz de usuario para el control del sistema, permitiendo a los usuarios interactuar con el prototipo.
- **Desarrollo del Panel de Monitorización**: Creación de gráficos y visualizaciones tanto en tiempo real como históricos del sistema.
- **Implementación de Comunicaciones**: Configuración de los protocolos de comunicación entre el SCADA y el prototipo físico, asegurando una transmisión de datos confiable y eficiente.
- **Configuración de la Base de Datos**: Implementación de un sistema de almacenamiento de datos para registrar y gestionar la información relevante del sistema.

4. Pruebas y Validación

Esta fase se centró en la realización de pruebas exhaustivas para garantizar el correcto funcionamiento del sistema y validar su efectividad.

5. Documentación

La fase final consistió en la redacción de la documentación completa del proyecto, incluyendo memoria del proyecto y documentación del código desarrollado.

2. Sistemas SCADA

En la gran mayoría de las organizaciones industriales se encuentran sistemas tanto simples como complejos que deben ser supervisados y controlados continuamente por los operarios de la instalación. Para la automatización industrial se han desarrollado diversos sistemas que permiten, con toda precisión, la supervisión y control de la planta industrial en cuestión, estos sistemas son los denominados: sistemas SCADA.

SCADA es el acrónimo de los términos en inglés “Supervisory Control and Data Acquisition” cuya traducción es supervisión, control y adquisición de datos [7]. Estos términos describen a la perfección las diferentes etapas que se deben llevar a cabo para mantener un control correcto de cualquier proceso industrial, desde la monitorización en tiempo real hasta el control de los equipos presentes en dicho proceso.

Los sistemas SCADA son esenciales en la supervisión y control de procesos industriales y de infraestructura. Estos sistemas integran hardware y software para permitir la recopilación de datos en tiempo real, la supervisión y el control remoto de equipos, y el almacenamiento de datos históricos para posteriores análisis. Los componentes típicos de un sistema SCADA incluyen:

A) Sensores y actuadores

Los sensores y actuadores son componentes fundamentales en cualquier sistema industrial, ya que permiten la interacción directa con el entorno físico.

- **Sensores:** Son dispositivos encargados de recoger datos de campo, midiendo diversas variables que son cruciales para el monitoreo y control del proceso. Entre los tipos de sensores más comunes se incluyen sensores de temperatura, presión, nivel...

- **Actuadores:** Son dispositivos que ejecutan acciones de control basadas en instrucciones recibidas del sistema de control. Convierten las señales de control en acciones físicas. Algunos de los más importantes son: electroválvulas, motores eléctricos, bombas...

Tanto sensores como actuadores trabajan en conjunto para mantener las condiciones operativas dentro de los parámetros deseados, garantizando la eficiencia y seguridad de los sistemas.

B) Controladores Lógicos Programables (PLC)

Los Controladores Lógicos Programables (PLC) son procesadores industriales diseñados específicamente para manejar múltiples entradas y salidas en entornos de automatización. Su robustez y capacidad para operar en condiciones adversas los hacen ideales para aplicaciones industriales.

Los PLCs se utilizan para controlar una variedad de procesos industriales mediante la adquisición de datos de sensores, el procesamiento de esos datos según un conjunto de instrucciones programadas y el envío de comandos a los actuadores para realizar las acciones deseadas.

Características Principales de los PLCs:

- Entradas y Salidas (I/O): Los PLCs tienen una gran cantidad de entradas y salidas tanto digitales como analógicas para conectar sensores y actuadores de todo tipo.
- Procesamiento en Tiempo Real: Capacidad para procesar datos y ejecutar programas en tiempo real, lo cual es crucial para el control preciso de procesos industriales.
- Robustez: Diseñados para operar en condiciones industriales difíciles, incluyendo temperaturas extremas, humedad, vibraciones...

C) Interfaz de Operador (HMI)

La Interfaz de Operador (HMI, por sus siglas en inglés) es un componente crucial en los sistemas SCADA, ya que permite a la personas interactuar con el sistema para supervisar y controlar los procesos industriales. Esta interfaz proporciona una representación visual del estado del sistema y permite la entrada de comandos de control.

Funciones Principales:

- Visualización de Datos: Muestra datos en tiempo real, permitiendo a los operadores monitorear el estado del sistema.
- Interacción con el Sistema: Permite enviar comandos de control para ajustar parámetros del sistema, activar o desactivar modos de funcionamiento, y responder a alarmas y eventos.

D) Red de comunicaciones

La infraestructura de red es fundamental para permitir la transferencia de datos entre los diferentes componentes del sistema SCADA. En la industria, se utilizan protocolos robustos de comunicación como Modbus, Ethernet y otros protocolos industriales estándar que garantizan una comunicación fiable y segura. Estos protocolos permiten la integración de múltiples dispositivos y la transferencia eficiente de datos a través de la red.

3. Diseño del sistema SCADA

3.1. Descripción del prototipo físico

El prototipo físico para la simulación y detección de fugas en depósitos de combustible está compuesto por una serie de depósitos interconectados y dispositivos electrónicos montados en una estantería.

El conjunto simula el funcionamiento de una estación de servicio, permitiendo la visualización y el control del estado de los depósitos en tiempo real. Este sistema proporciona una plataforma integral para la monitorización y gestión de los depósitos, emulando condiciones reales de operación y facilitando la identificación y análisis de posibles fugas en un entorno controlado.

La estructura física incluye una controladora, sensores de nivel y presión, actuadores, tuberías y depósitos. Existe un sistema de control basado en Arduino que está conectado a la planta, el cual se comunica de manera serial con el SCADA. Este diseño permite una supervisión detallada y precisa, así como una respuesta rápida ante cualquier anomalía detectada. La integración de hardware y software en este prototipo asegura una representación fiel de las operaciones de una estación de servicio, proporcionando un entorno de prueba efectivo para la simulación y detección de fugas en los depósitos de combustible.

Este prototipo, representado en la figura 1, no solo sirve como una herramienta educativa, sino que también actúa como una plataforma de prueba para métodos de detección de fugas a partir de los datos generados, contribuyendo al avance y mejora continua de estos sistemas de detección.

Figura 1. Fotografía del prototipo

Los componentes principales (Figura 2) son los siguientes:

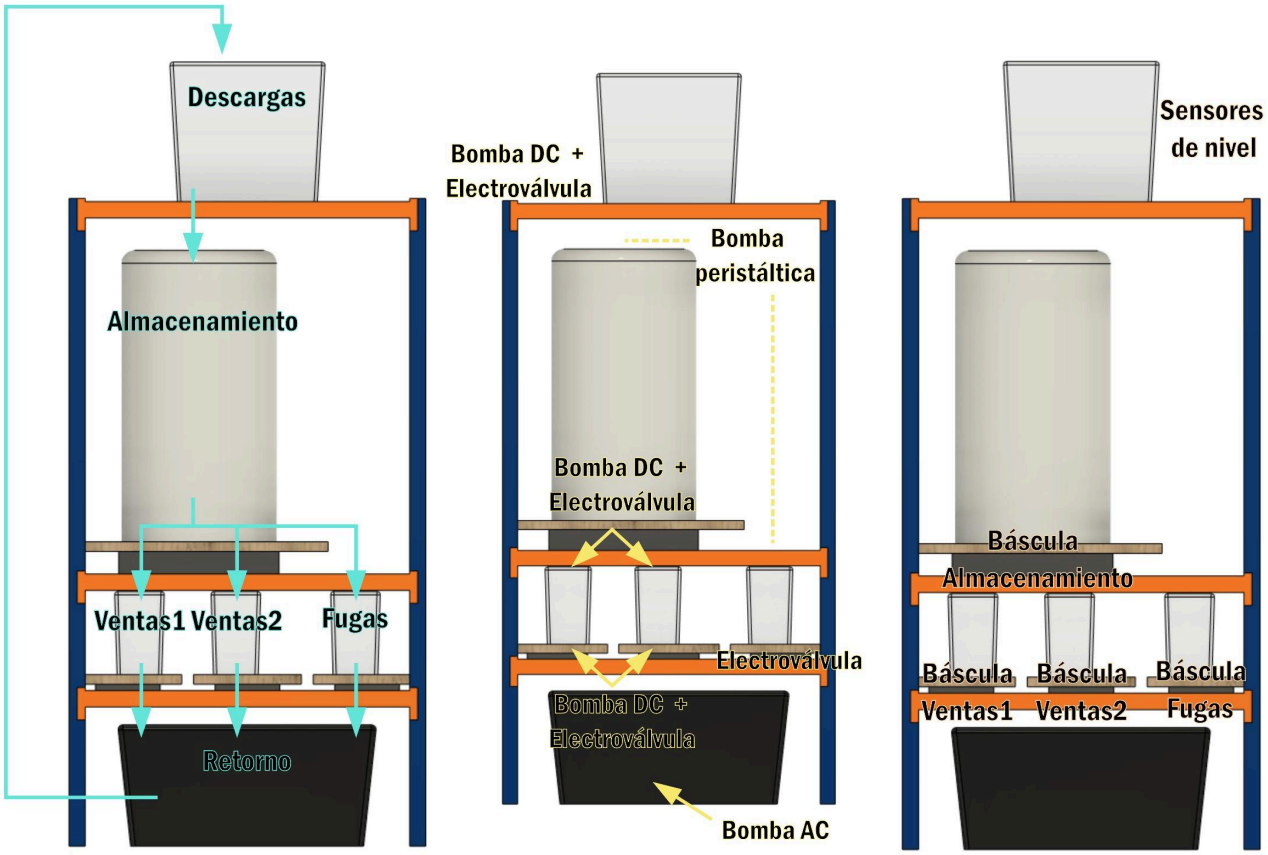


Figura 2. Disposición de los depósitos, actuadores y sensores del prototipo [6]

1. ***Depósitos de Agua***

- Depósito de Descargas: tiene una capacidad de 33 L, cumple la función de simular un camión cisterna que periódicamente efectúa descargas en el depósito de almacenamiento.

- Depósito de Almacenamiento o Principal: Con una capacidad de 125 L, representa el tanque subterráneo de las estaciones de servicio, desde donde se realizan las dispensaciones los depósitos de ventas y fugas.

- ❖ Depósitos de Ventas 1 y 2: Cada uno con una capacidad de 4 L, simulan los surtidores de combustible, llenándose y vaciándose cíclicamente.
- ❖ Depósito de Fugas: También de 4 L, almacena la fuga simulada desde el depósito principal.
- ❖ Depósito de Retorno: Con una capacidad de 150 L, devuelve el agua de los depósitos de Ventas y Fugas al depósito de Descargas.

2. ***Bombas y Válvulas***

- ❖ Bombas Peristálticas y Centrífugas: Utilizadas para mover el agua entre los diferentes depósitos. Las bombas centrífugas están equipadas con electroválvulas acopladas para controlar el flujo.

3. ***Sensores y Actuadores***

- ❖ Sensores: Dispositivos para conocer la cantidad de líquido en los depósitos para monitorizar y controlar el llenado y vaciado. Para saber la cantidad de líquido en cada depósito se usan básculas. Sin embargo en el depósito de descargas se usan también sensores de nivel.
- ❖ Actuadores: Incluyen las bombas y válvulas que realizan acciones de control basadas en las instrucciones del sistema.

4. ***Panel de Control Físico***

- ❖ Pulsadores y Conmutadores: Para iniciar, detener y controlar el proceso, incluyendo botones para bombeo y conmutadores para selección de modos.
- ❖ Potenciómetros y LEDs: Para ajustar parámetros de operación y visualizar el estado del sistema mediante indicadores luminosos.

5. ***Unidad de Control***

En este proyecto específico, se ha optado por utilizar un microcontrolador Arduino Mega 2560 (Figura 3) en lugar de un PLC tradicional. El Arduino actúa como la unidad central de control del sistema SCADA, y cumple funciones similares a las de un PLC.

- Entradas y Salidas: El Arduino tiene múltiples pines I/O que permiten la conexión de diversos sensores y actuadores.
- Procesamiento de Datos: Recoge datos de los sensores, los procesa en tiempo real y envía instrucciones a los actuadores.
- Flexibilidad y Simplicidad: El uso del Arduino proporciona flexibilidad y es adecuado para el prototipo debido a su simplicidad y facilidad de programación utilizando el entorno de desarrollo Arduino (IDE).

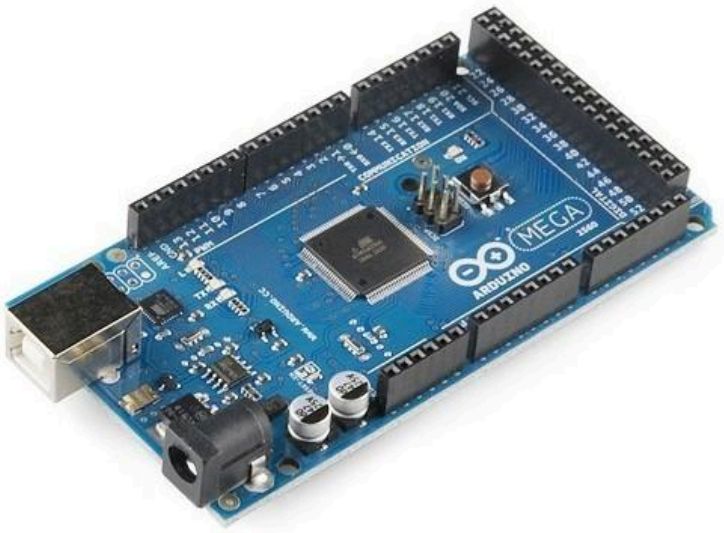


Figura 3. Controladora Arduino Mega

3.2. Arquitectura del sistema

La arquitectura del sistema está diseñada para asegurar la eficiencia y fiabilidad en la monitorización y control del prototipo de detección de fugas. La estructura del sistema se organiza en varias capas y componentes clave que interactúan entre sí para proporcionar una solución integral.

Unidad de control del sistema

El microcontrolador Arduino actúa como unidad de control, recibe datos de los sensores, los procesa según el programa predefinido y envía las instrucciones necesarias a los actuadores para mantener el control del sistema.

Dispositivos de Campo

Incluyen todos los sensores y actuadores distribuidos en el prototipo, encargados de la adquisición de datos y la ejecución de acciones.

Sistema SCADA

Para la configuración del SCADA, utilizamos un miniPC PIPO X10S (Figura 4) con el sistema operativo Windows 10 y un monitor externo utilizado como pantalla de monitorización. El PC Pipo es un dispositivo compacto y robusto, diseñado para entornos industriales y de laboratorio. Su tamaño reducido permite una fácil instalación en la estación de control central, ocupando poco espacio y ofreciendo alta portabilidad.

El PC PIPO actúa como el cerebro del SCADA, procesando y visualizando los datos. La pantalla de este PC es un panel táctil que se usará para el panel de control del sistema mientras que el monitor estándar conectado se usará como panel de monitorización.

En el Pipo se ejecuta el software de la interfaz de usuario, proporcionando una HMI (Human-Machine Interface) gráfica que permite

a los operadores interactuar con el sistema. Utilizando herramientas como Dash y Plotly, se diseñan paneles de control y monitorización que muestran datos en tiempo real y de manera histórica.

Figura 4. Mini PC PIPO X10S

Las interfaces de las que dispondrá el sistema son:

- **Interfaz de Control:** permite a los usuarios interactuar directamente con el sistema. A través de esta interfaz, los usuarios pueden enviar comandos al sistema para ejecutar diversas acciones, ajustar parámetros como las dimensiones de las fugas y proporcionar la capacidad de reaccionar de manera inmediata ante eventos o alertas.

- **Interfaz de Monitorización:** está diseñada para ofrecer una supervisión detallada y continua del sistema. Sus características incluyen la visualización en tiempo real, mostrando el estado actual del sistema y permitiendo a los usuarios monitorear continuamente las condiciones operativas. Además, permite el almacenamiento de datos históricos de operación, lo cual es fundamental para el análisis y diagnóstico de tendencias y patrones a lo largo del tiempo. Por último, facilita la visualización y análisis de los datos históricos.

Comunicación Arduino-SCADA

En un entorno industrial, se utilizan protocolos robustos y estandarizados como Modbus, Ethernet y otros protocolos industriales para asegurar una comunicación fiable y segura. Estos protocolos permiten la integración de múltiples dispositivos y la transferencia eficiente de datos.

Para este proyecto específico, se ha optado por una arquitectura que incorpora tanto el uso de MQTT como una comunicación serial simple. A continuación se detallan ambos métodos:

Comunicación Serial entre Arduino y SCADA

Para la comunicación entre el microcontrolador Arduino y los componentes del sistema SCADA, se ha optado por una comunicación serial simple, representada en la figura 5. Este tipo de comunicación es adecuada para aplicaciones donde se requiere una conexión directa y sencilla entre dispositivos. La comunicación serial es fácil de implementar y suficiente para las necesidades de este proyecto, proporcionando una transferencia de datos fiable y eficiente sin la complejidad de los protocolos industriales más avanzados..

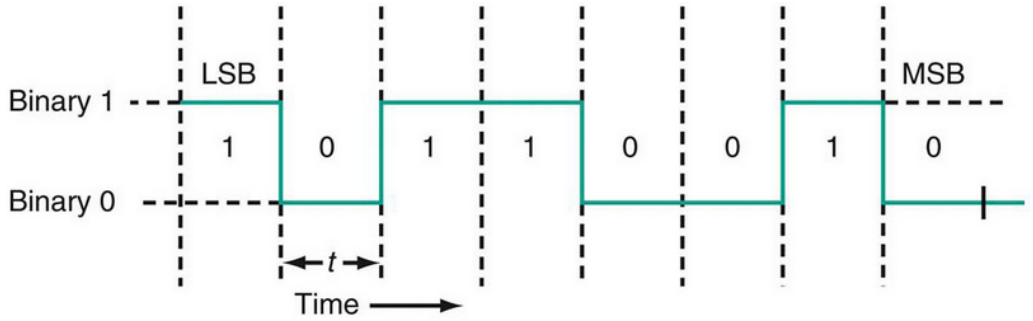


Figura 5. Comunicación binaria en serie [8]

Uso de MQTT para la Comunicación

Se ha optado por utilizar MQTT (Message Queuing Telemetry Transport), un protocolo de mensajería ligero diseñado para maximizar la eficiencia en la transmisión de datos. MQTT es ideal para la comunicación entre dispositivos IoT (Internet de las Cosas), ya que permite la publicación

y suscripción de mensajes entre los distintos componentes del sistema SCADA. Esto asegura una comunicación eficiente, fiable y escalable.

3.3. Diseño de la interfaz de usuario

El diseño de la interfaz de usuario es un componente crítico en el sistema SCADA, ya que permite a los operadores interactuar de manera efectiva con el sistema para supervisar y controlar los procesos. Esta interfaz debe ser intuitiva, fácil de usar y proporcionar información clara y precisa en tiempo real.

La distribución del sistema se ha diseñado con dos paneles principales: un panel de mando y otro panel de monitorización. Esta distribución asegura una gestión eficiente y eficaz del sistema, combinando control activo y monitorización detallada para mantener la operación segura y optimizada.

1. **Panel de Mando:** Este panel permite a los operadores interactuar activamente con el sistema, enviando comandos y ajustando parámetros operativos.
2. **Panel de Monitorización:** Este panel proporciona una visión detallada y continua del estado del sistema. Permite la visualización en tiempo real de las variables críticas, el acceso a datos históricos para análisis y diagnóstico, y la gestión de errores y alarmas, asegurando una supervisión completa y precisa del sistema.

3.3.1. Panel de control

El panel de control del sistema SCADA está diseñado para proporcionar a los operadores una interfaz intuitiva y eficiente que les permita interactuar directamente con el sistema. Utilizando herramientas como Dash y Plotly, se ha desarrollado una interfaz gráfica de usuario (GUI) que facilita la supervisión y el control del sistema en tiempo real.

La GUI del panel de control presenta un diseño intuitivo, con una disposición clara y accesible de los elementos interactivos, como botones, controles deslizantes y selectores. Estos elementos permiten a los operadores realizar ajustes y enviar comandos de manera rápida y eficiente.

Los indicadores de estado y alarmas proporcionan una visión inmediata del estado del sistema, permitiendo a los operadores conocer rápidamente cualquier condición anómala.

Inicialmente, se ha realizado un diseño preliminar del panel de control a implementar, este diseño se puede ver en la figura 6. Este diseño sirve para obtener una idea clara de cómo quedará el panel final y permite visualizar los elementos que lo compondrán. A través de este diseño, se pueden identificar y ajustar los componentes necesarios, asegurando que la disposición y funcionalidad del panel sean óptimas.

Sin embargo, tras la implementación, el diseño ha sido modificado ligeramente para mejorar su usabilidad y eficacia.

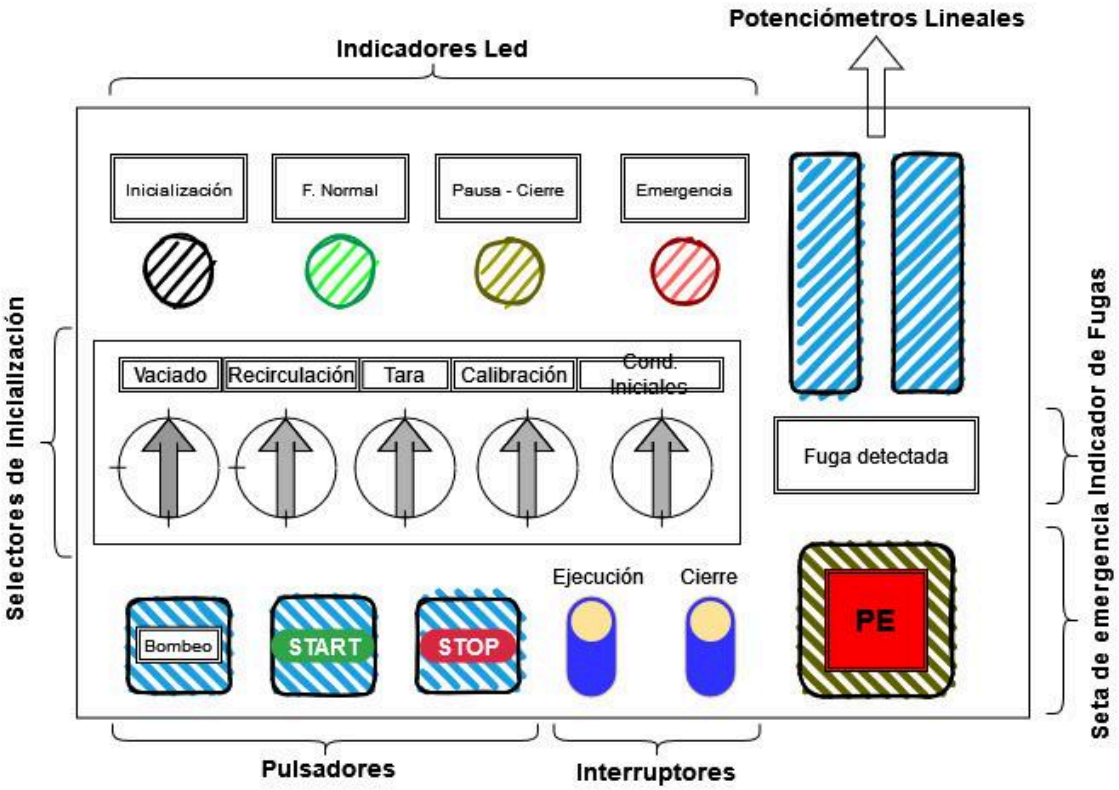


Figura 6. Diseño inicial del panel de mando

3.3.2. Panel de monitorización

El panel de monitorización del sistema SCADA está diseñado para proporcionar a los usuarios una visión detallada y en tiempo real del estado del sistema, así como acceso a datos históricos para análisis y diagnóstico. Este panel se implementa utilizando herramientas como Dash y Plotly, que permiten crear visualizaciones interactivas y ricas en datos. El panel de monitorización se encuentra organizado en 3 subpaneles.

Estado de la planta (Figura 7): muestra gráficos y tablas que representan datos en tiempo real de los sensores del sistema. Estos gráficos se actualizan continuamente, ofreciendo a los usuarios una visión precisa y actualizada del funcionamiento del sistema. La claridad de estas visualizaciones es crucial para detectar rápidamente cualquier anomalía.

Históricos (Figura 8): además de la visualización en tiempo real, el panel de monitorización almacena y muestra datos históricos. Esta

funcionalidad permite a los usuarios revisar el funcionamiento del sistema a lo largo del tiempo, identificar patrones y tendencias, y realizar análisis detallados. Los datos históricos son esenciales para el diagnóstico de problemas, la planificación de mantenimiento preventivo y la optimización del rendimiento de los sistemas.

Depuración, Alarmas y Alertas (Figura 9): otra característica destacada del panel de monitorización es la capacidad de mostrar mensajes de depuración en tiempo real. Estos mensajes se recogen de diferentes partes del código y se agrupan en cuatro conjuntos principales: mensajes del panel de control, del panel de monitorización, de la base de datos y de la comunicación serial. Estos mensajes se muestran en un cuadro de texto dedicado, proporcionando información crucial sobre el estado del sistema SCADA. Esta funcionalidad es necesaria para el diagnóstico rápido de problemas y para entender el comportamiento del sistema.

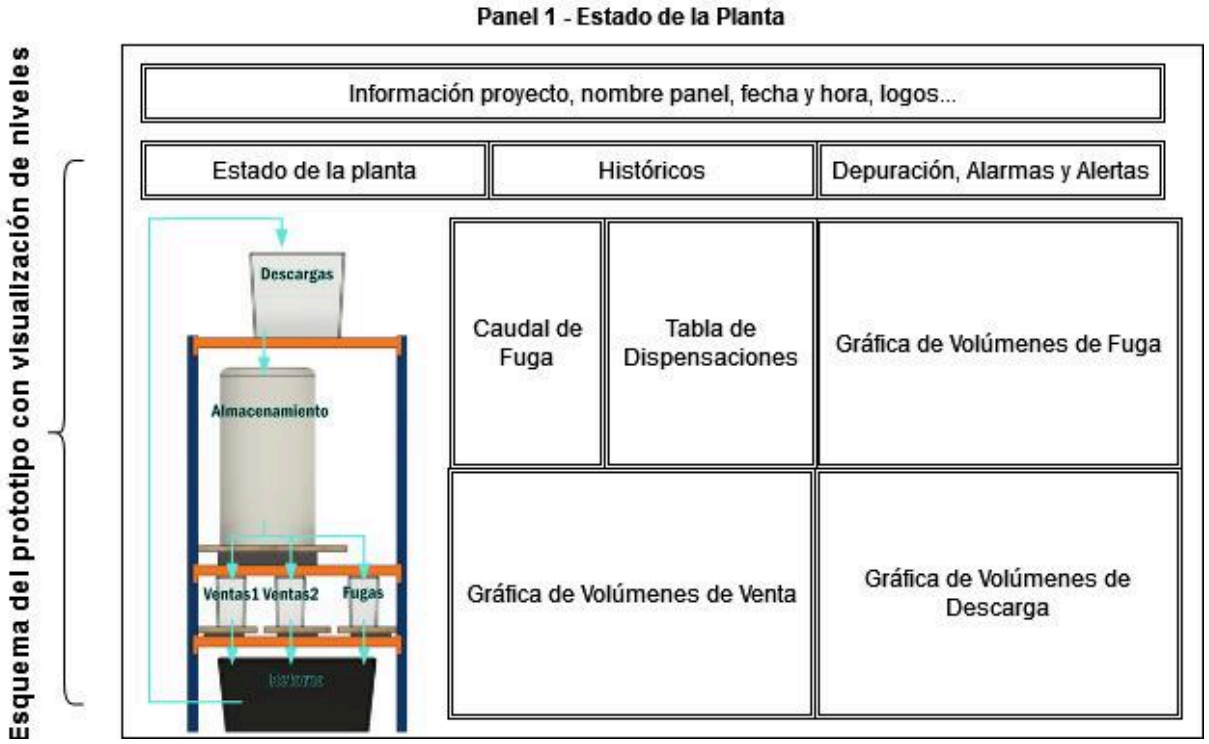


Figura 7. Diseño inicial del panel de monitorización - Estado de la planta

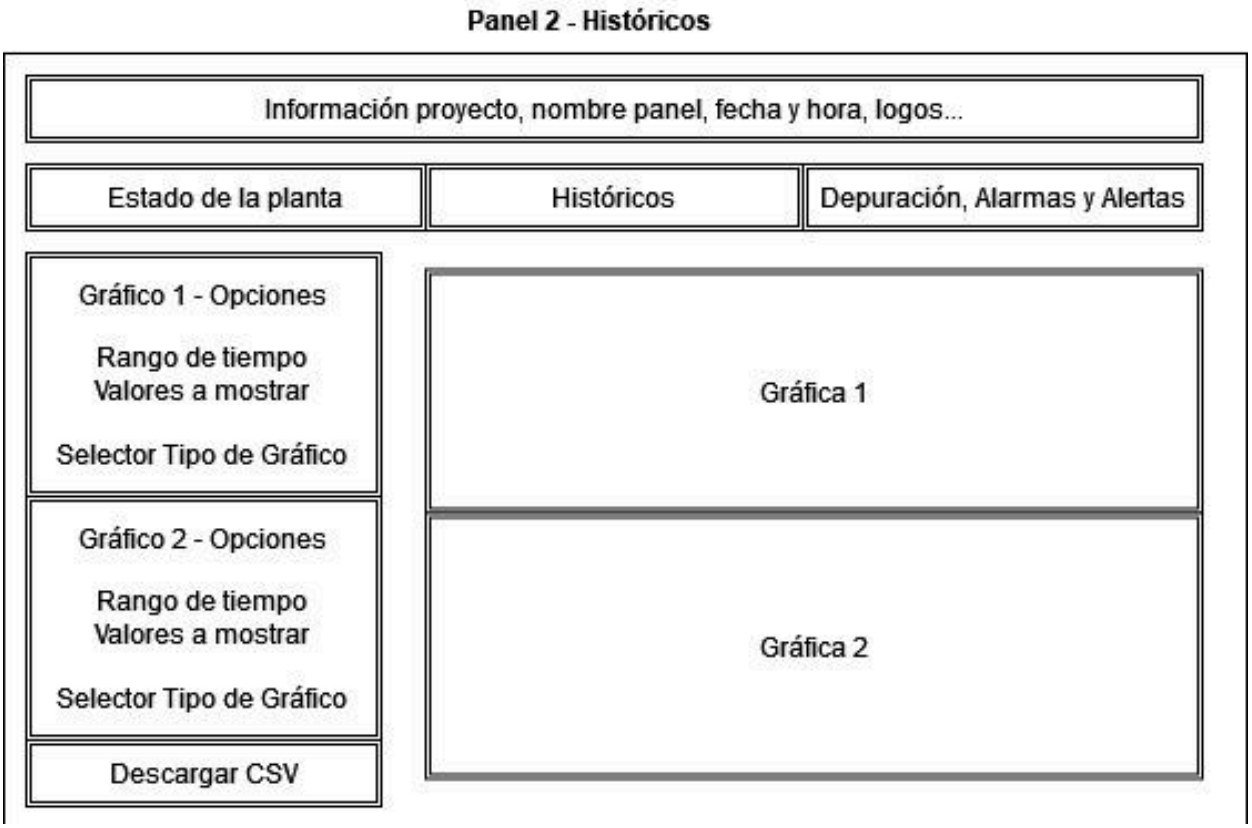


Figura 8. Diseño inicial del panel de monitorización - Históricos

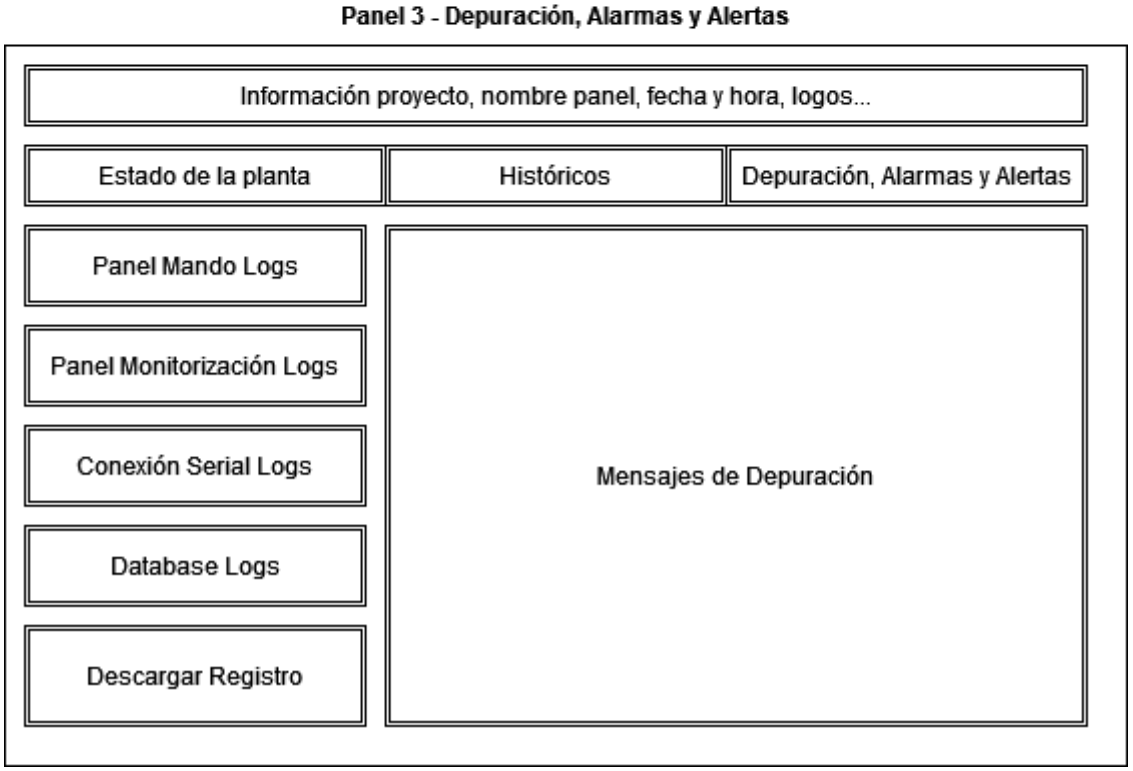


Figura 9. Diseño inicial del panel de monitorización - Depuración y Alarmas

4. Implementación del proyecto

El sistema SCADA desarrollado para la detección de fugas se ha implementado utilizando el lenguaje de programación Python.

Python se eligió por su versatilidad, facilidad de uso y la amplia disponibilidad de bibliotecas y herramientas para la creación de aplicaciones web interactivas, procesamiento de datos y comunicación con dispositivos de hardware.

Además, Python permite una rápida iteración y desarrollo, lo que facilita la implementación de mejoras y ajustes necesarios en el sistema. Python cada vez se usa más en el campo de la automatización, convirtiéndose en una opción preferida para el desarrollo de sistemas SCADA modernos y otros sistemas de control industrial.

4.1. Estructura del proyecto

El programa desarrollado para el sistema SCADA se organiza en una estructura de archivos clara y bien definida, ilustrado en la figura 10.

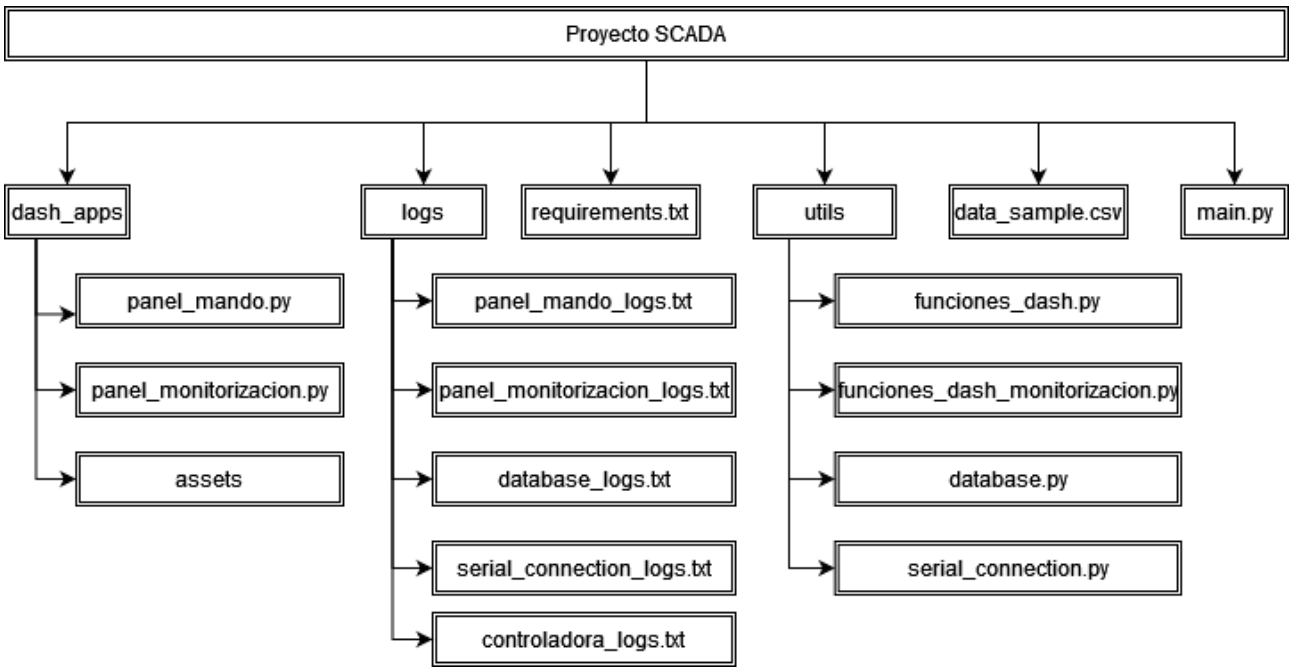


Figura 10. Estructura del proyecto

En el primer nivel (Figura 11) que encontramos en el proyecto tenemos:

1. `main.py`**:** Es el archivo principal que inicia la aplicación SCADA. Configura y ejecuta el servidor web que aloja los paneles de control y monitorización. Coordina la inicialización de las diferentes partes del sistema.

2. `data_sample.csv`**:**: Este archivo actúa como la base de datos donde se almacena toda la información recibida del microcontrolador Arduino.

3. `requirements.txt:` Lista de todas las dependencias y paquetes necesarios para ejecutar la aplicación SCADA. Incluye librerías como Dash, Plotly, Pandas...

4. **Carpeta dash_apps**: Contiene los archivos relacionados con las aplicaciones Dash que forman los paneles de control y monitorización.

5. **Carpeta utils:** Contiene módulos de utilidades y funciones auxiliares necesarias para el funcionamiento del sistema SCADA.

6. **Carpeta logs:** Almacena archivos de registro que contienen información sobre el funcionamiento del sistema y eventos importantes.

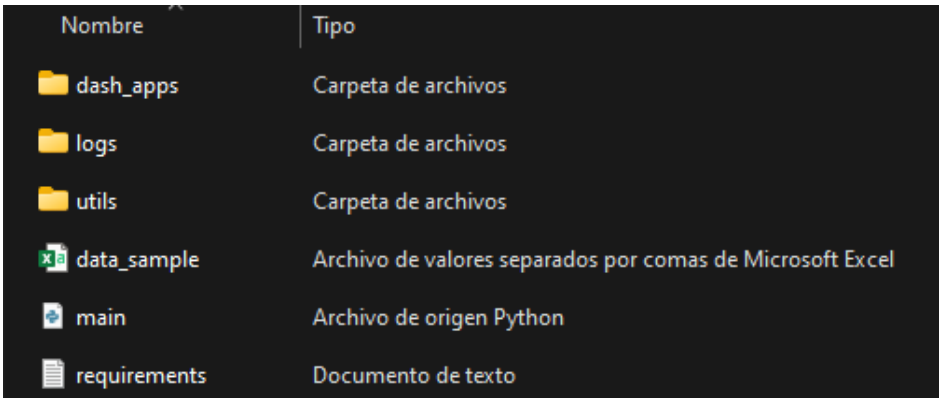


Figura 11. Archivos principales del proyecto

A continuación se explicarán los archivos que se encuentran dentro de cada carpeta.

Carpeta dash_apps (Figura 12)

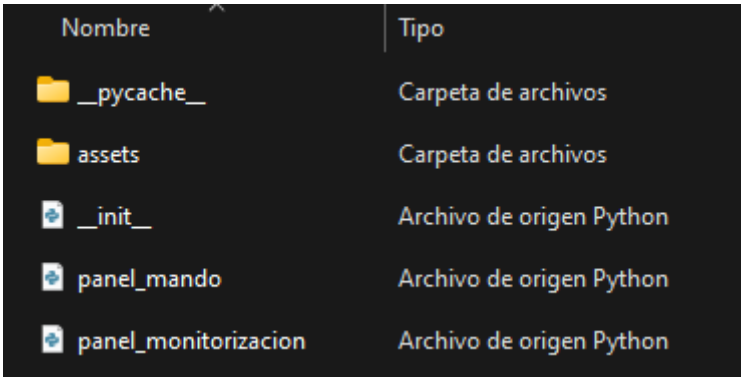


Figura 12. Contenido carpeta *dash_apps*

- `panel_mando.py`**:** Define la estructura y funcionalidad del panel de control. Incluye las interfaces de usuario, gráficos y controles interactivos necesarios para el manejo del sistema.
- `panel_monitorizacion.py`**:** Define la estructura y funcionalidad del panel de monitorización. Incluye visualizaciones de datos en tiempo real, gráficos históricos, y el sistema de alertas y depuración.
- `init.py`: Inicializa el paquete dash_apps.
- **pycache**: Carpeta generada automáticamente que contiene archivos de caché de Python para optimizar el rendimiento.
- **assets:** carpeta que incluye las imágenes que importamos en el proyecto, tienen formato de gráficos vectoriales escalables (.svg). Entre estas imágenes se encuentran el diseño del prototipo, de los selectores y de la parada de emergencia.

Carpeta utils (Figura 13)

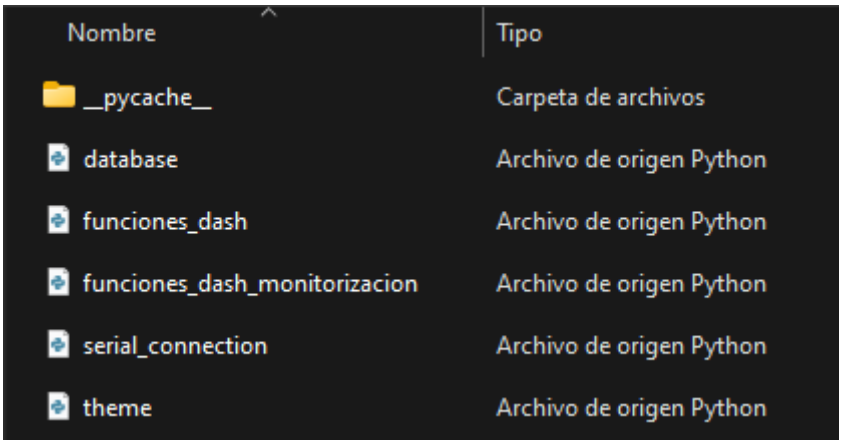


Figura 13. Contenido carpeta *utils*

- `database.py`**:** Maneja las operaciones de la base de datos, incluyendo la lectura y escritura de datos históricos.
- `funciones_dash.py`**:** Contiene funciones específicas utilizadas en el panel de control para la definición de elementos como indicadores, selectores, interruptores o controles deslizantes.
- `funciones_dash_monitorizacion.py`**:** Contiene funciones específicas utilizadas en el panel de monitorización, incluyendo la definición de elementos como los tanques o pulsadores, la creación de funciones como el selector de fechas o las check-list y la creación de gráficas para las visualizaciones de valores históricos.
- `serial_connection.py`**:** Gestiona tanto la comunicación serial entre el microcontrolador Arduino y el PC Pipo como el protocolo MQTT para actuar tanto como suscriptor como publicador. Incluye funciones para el envío y recepción de datos.
- `theme.py`**:** Define el tema visual y estilos utilizados en las aplicaciones Dash para asegurar una apariencia coherente y profesional.

Carpeta logs (Figura 14)

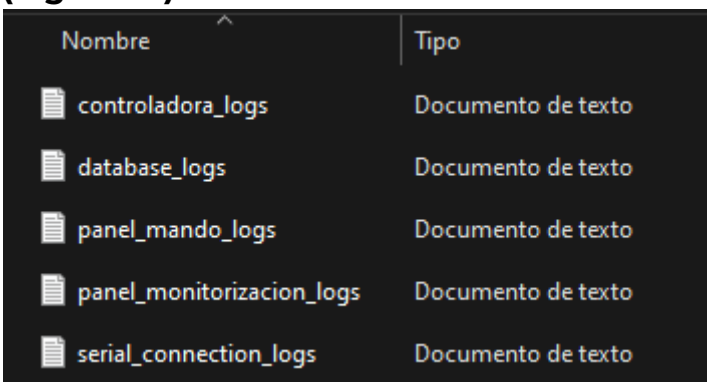


Figura 14. Contenido carpeta *logs*

- `panel_mando_logs.txt`**:** Registra eventos y errores específicos del panel de control.
- `panel_monitorizacion_logs.txt`**:** Registra eventos y errores específicos del panel de monitorización.
- `database_logs.txt`**:** Registra eventos y errores relacionados con las operaciones de la base de datos.
- `serial_connection_logs.txt`**:** Registra eventos y errores relacionados con la comunicación serial Arduino-SCADA.
- `controladora_logs.txt`**:** Registra eventos y mensajes de depuración relacionados con la controladora arduino. Sirve para saber en qué estado se encuentran las máquinas de estado tanto de la controladora como de cada uno de los depósitos del sistema.

Como se puede apreciar, la estructura organizada del programa asegura una clara separación de responsabilidades y facilita el mantenimiento y la expansión del sistema SCADA. Cada archivo y carpeta tiene un propósito específico que contribuye al funcionamiento integral del sistema, desde la interfaz de usuario hasta la comunicación con los dispositivos de campo. Esta organización permite una gestión eficiente del código y asegura la escalabilidad del proyecto.

4.2. Comunicaciones entre SCADA y prototipo

La comunicación entre el sistema SCADA y la planta es crucial para asegurar el flujo continuo de datos entre los sensores y actuadores en el campo y el sistema central de control y monitoreo. En este proyecto, se emplea una combinación de métodos y protocolos para asegurar una comunicación efectiva y fiable.

En el prototipo desarrollado, se utiliza la comunicación serial para transmitir datos entre el microcontrolador Arduino y el PC Pipo. La comunicación serial es una forma sencilla y eficiente de conectar dispositivos electrónicos, permitiendo el intercambio de datos mediante una conexión directa. A continuación se describen los aspectos clave de la comunicación serial implementada en este proyecto:

- **Protocolo Utilizado**: Se emplea el protocolo USB-Serial, que permite la comunicación serie a través de una conexión USB. Este protocolo facilita la transmisión de datos entre el Arduino y el PC Pipo. Para que la conexión con el puerto serie se realice de manera regular en segundo plano se ha utilizado un objeto Thread que representa una determinada operación que se ejecuta como un subproceso independiente, es decir, representa a un hilo.

- **Conexión Física**: La comunicación se realiza a través de un cable USB tipo B que conecta el puerto USB del Arduino con el puerto USB del PC Pipo.

- **Velocidad de Transmisión**: La velocidad de transmisión (baud rate) se ha configurado a 9600 baudios, una tasa común que equilibra la velocidad de transferencia de datos y la fiabilidad de la comunicación.

- **Formato de Datos**: Los datos se transmiten en paquetes que incluyen un bit de inicio, 8 bits de datos, y un bit de parada. Este

formato asegura que los datos se interpreten correctamente en ambos extremos de la comunicación.

Además, se implementa en el SCADA el protocolo de comunicación MQTT (Figura 15) con el fin de poder usar paneles de monitorización de manera remota. Las principales características de MQTT son:

- **Publicación/Suscripción:** MQTT opera bajo un modelo de publicación/suscripción que desacopla los productores de datos (publicadores) de los consumidores de datos (suscriptores). Un broker MQTT gestiona las conexiones y las comunicaciones entre estos clientes. El broker utilizado es `broker.hivemq.com`.
- **Topic:** En este modelo, los mensajes se organizan en "topics" o temas, que actúan como canales para la comunicación. Los publicadores envían mensajes a un topic específico, y los suscriptores reciben mensajes de los topics a los que están suscritos. Por ejemplo, el SCADA podría publicar datos en el topic `scada/monitorizacion`, y un sistema de monitoreo podría suscribirse a este topic para recibir y procesar esos datos.
- **Ligero:** Está diseñado para ser un protocolo ligero con sobrecarga mínima, lo que lo hace ideal para dispositivos con recursos limitados, como sensores y microcontroladores.
- **Persistencia de Mensajes:** MQTT puede almacenar mensajes en el broker hasta que los suscriptores estén disponibles para recibirlos, asegurando que no se pierdan datos críticos.
- **Escalabilidad:** Permite la conexión de miles de dispositivos simultáneamente, gestionando eficientemente la comunicación entre ellos.

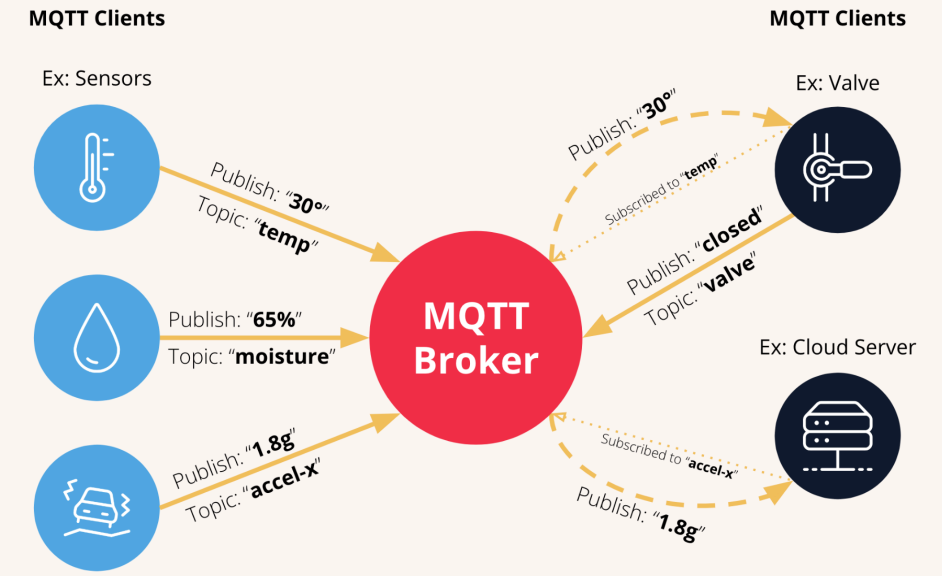


Figura 15. Protocolo MQTT [9]

En el proyecto, el sistema SCADA conectado al Arduino actúa como publicador de los datos del sistema. Esto significa que los datos recopilados por el Arduino se publican en el topic definido, como `scada/monitorizacion`. Cualquier sistema de monitoreo que conozca la estructura de los datos podría suscribirse a este topic para recibir y procesar esos datos. Esta arquitectura permite que el panel de monitorización acceda a estos datos desde cualquier lugar con conexión a Internet, proporcionando una visión en tiempo real del estado de la estación de servicio.

Además, en futuras expansiones del proyecto, se podría configurar el sistema SCADA para que funcione también como suscriptor. Esto permitiría recibir comandos remotos para el control del sistema, abriendo la posibilidad de gestionar y ajustar parámetros del sistema SCADA desde ubicaciones remotas. Por ejemplo, un usuario podría enviar comandos a través de un topic específico como `comandos/control` para ajustar la configuración del sistema. Esta capacidad mejoraría significativamente la flexibilidad y el alcance del sistema, permitiendo una gestión más dinámica y reactiva del prototipo.

4.2.1 Implementación en el sistema SCADA

La implementación de la comunicación serial en el sistema SCADA se realiza a través del fichero `serial_connection.py`, que gestiona todas las operaciones de envío y recepción de datos. A continuación se detalla cómo se lleva a cabo esta implementación:

- **Configuración de la Conexión**: El script configura los parámetros de la comunicación serial (puerto, velocidad de transmisión, formato de datos) y ejecuta la conexión entre el Arduino y el PC Pipo. También establece los parámetros y la conexión con el broker del protocolo MQTT.
- **Envío de Datos**: Los datos recopilados por los sensores conectados al Arduino se envían al PC Pipo a través de la conexión serial. El Arduino empaqueta los datos en el formato adecuado y los transmite de manera continua.
- **Recepción de Datos**: El PC Pipo recibe los datos enviados por el Arduino, los procesa y los almacena en la base de datos. Estos datos son utilizados para actualizar las visualizaciones en tiempo real en los paneles de control y monitorización. A su vez, esta información se publica en el topic correspondiente del protocolo MQTT.
- **Manejo de Errores**: El script incluye mecanismos para detectar y manejar errores de comunicación, asegurando que los datos se transmitan de manera fiable. Los errores comunes, como la pérdida de paquetes o la interferencia, se gestionan mediante reintentos y validación de datos.

4.2.2 Beneficios de la comunicación serial

La comunicación serial ofrece una serie de beneficios que la hacen una opción atractiva para diversas aplicaciones. Uno de sus principales beneficios es su simplicidad, es fácil de implementar y configurar, lo que la convierte en una opción ideal para prototipos y sistemas de pequeña escala como es el caso.

Otro beneficio significativo de la comunicación serial es su bajo costo. Este tipo de comunicación requiere un hardware mínimo, un cable entre el arduino y el PC PIPO. Con una configuración adecuada, este tipo de comunicación puede proporcionar una transmisión de datos fiable y continua. Esto es crucial para aplicaciones que requieren una transferencia de datos constante y sin interrupciones, asegurando que la información llegue de manera precisa y en el momento adecuado.

4.2.3 Consideraciones para futuras expansiones

Aunque la comunicación serial es adecuada para el prototipo, las aplicaciones industriales a gran escala pueden requerir protocolos de comunicación más robustos y versátiles. En estos casos, se consideran alternativas como Modbus, Ethernet/IP, y otros protocolos industriales que ofrecen mayores velocidades de transmisión, mejor gestión de errores y capacidad para conectar múltiples dispositivos en una red compleja.

4.3. Almacenamiento de datos

El almacenamiento de datos es un componente crucial del sistema SCADA, ya que permite la recopilación, almacenamiento y acceso a la información histórica y en tiempo real generada por los sensores y dispositivos del sistema. En este proyecto, se utiliza un archivo CSV para manejar los datos de manera eficiente y organizada.

El archivo CSV (`data_sample.csv`) actúa como la base de datos principal donde se almacena toda la información recibida del microcontrolador Arduino. Este archivo facilita el desarrollo, pruebas y operación del sistema SCADA. A continuación se describen las características y el uso del archivo CSV:

- El archivo CSV almacena los datos en un formato tabular, con cada fila representando un registro de datos y cada columna representando un atributo específico (por ejemplo, instante actual, volumen de los depósitos o caudal de fuga). Este formato es ampliamente compatible y fácil de manipular utilizando bibliotecas de Python como `pandas`.

- Las operaciones de lectura y escritura en el archivo CSV se gestionan mediante el módulo `database.py`. Este módulo proporciona funciones para insertar nuevos registros y recuperar datos para su visualización y análisis. Utilizando `pandas`, se pueden realizar estas operaciones de manera eficiente, asegurando que los datos estén siempre actualizados y accesibles.

- Los datos recibidos del Arduino se escriben en el archivo CSV en tiempo real. Esto asegura que toda la información relevante se almacene de manera continua y esté disponible para su posterior análisis

- Aunque se utiliza un archivo CSV en lugar de una base de datos relacional, se implementan mecanismos para asegurar la integridad y seguridad de los datos almacenados. Esto incluye la validación de datos antes de su inserción.

Los datos almacenados en el archivo CSV se integran directamente con el panel de monitorización, permitiendo la visualización en tiempo real y el acceso a registros históricos.

5. Desarrollo del proyecto

La creación de los paneles de control y monitorización es esencial para el sistema SCADA, ya que ofrece a los usuarios las herramientas indispensables para supervisar y gestionar los procesos en tiempo real. En este proyecto, se han empleado diversas herramientas y bibliotecas para desarrollar interfaces de usuario que sean tanto intuitivas como efectivas.

5.1. Herramientas utilizadas para la visualización

Para el desarrollo de los paneles de control y monitorización del sistema SCADA, se han empleado varias herramientas y bibliotecas que facilitan la creación de interfaces de usuario interactivas y eficientes. A continuación, se describen las principales herramientas utilizadas:

- **Dash:** es un framework de Python que permite construir aplicaciones web analíticas e interactivas de manera sencilla. Es especialmente útil para crear interfaces gráficas que integran gráficos y tablas dinámicas, proporcionando una visualización detallada de los datos en tiempo real. Dash se basa en Flask, Plotly.js y React.js, lo que lo hace muy potente y flexible para aplicaciones de visualización de datos.

- **Plotly:** es una biblioteca de gráficos interactivos que se integra perfectamente con Dash. Permite crear una amplia variedad de visualizaciones, como gráficos de líneas, barras, dispersión y mapas de calor. Las visualizaciones generadas con Plotly son altamente interactivas y pueden actualizarse en tiempo real, lo que es crucial para el monitoreo continuo.

- **Pandas:** es una biblioteca de Python para la manipulación y análisis de datos. Se utiliza para manejar los datos almacenados en el archivo CSV (`data_sample.csv`), facilitando la lectura, escritura y procesamiento de grandes volúmenes de datos de manera eficiente.

Pandas permite realizar operaciones complejas de filtrado, agregación y transformación de datos, que son esenciales para preparar la información antes de su visualización.

- **PySerial**: esta biblioteca permite la comunicación serial con dispositivos externos, como el Arduino. Se utiliza para establecer y gestionar la conexión serial a través de USB, facilitando el envío y recepción de datos entre el Arduino y el PC Pipo. PySerial permite configurar parámetros como la velocidad de transmisión (baud rate) y el puerto serial, y proporciona funciones para leer y escribir datos.

- **MQTT:** Message Queuing Telemetry Transport es un protocolo de mensajería ligero que se utiliza para la comunicación entre dispositivos IoT. En nuestro proyecto, MQTT permite la publicación y suscripción de mensajes entre los distintos componentes, asegurando una comunicación eficiente y fiable. Utilizamos el broker `broker.hivemq.com` para gestionar las comunicaciones. MQTT es fundamental para la transmisión en tiempo real de los datos recopilados por el SCADA, facilitando su visualización desde cualquier lugar. En el futuro, se puede expandir su uso para permitir el control remoto del sistema SCADA.

- **Threads**: el uso de threads o hilos es crucial para la gestión de tareas concurrentes. En este proyecto, los threads permiten la ejecución simultánea de múltiples tareas, como son cada una de las instancias dash (panel de control y de monitorización) y la comunicación serial. La implementación de threads asegura que el sistema pueda manejar múltiples operaciones a la vez, mejorando así la eficiencia y la capacidad de respuesta del sistema SCADA.

- **HTML y CSS:** Se utilizan para dar estilo y estructura a las aplicaciones Dash, asegurando que las interfaces de usuario sean visualmente atractivas y fáciles de navegar.

5.2. Funcionamiento de dashboard

La construcción tanto del panel de control como del panel de monitorización comienza con la definición de la estructura de la aplicación. Esto se realiza utilizando el módulo `dash.html`, que proporciona clases para todas las etiquetas HTML (como `html.Div`), utilizadas para crear contenedores en la página web. Estos contenedores pueden alojar componentes de Dash, incluyendo gráficos, tablas y campos de entrada. La disposición de estos elementos en la página se gestiona mediante el uso de CSS, asignando estilos a cada contenedor `Div`.

Para actualizar la información visualizada, se emplea la funcionalidad de callbacks. Un callback es una función que se ejecuta en respuesta a un cambio en el estado de la aplicación, como la interacción del usuario con un elemento de entrada o la llegada de nuevos datos. Los callbacks se definen utilizando la función `app.callback`, que vincula un componente de entrada a un componente de salida. Así, cuando cambia el valor del componente de entrada, el callback se activa y devuelve la salida correspondiente permitiendo la actualización automática de los paneles dash.

La interacción fluida y la capacidad de respuesta en tiempo real de los paneles se logran gracias a esta arquitectura basada en callbacks, asegurando que los usuarios siempre vean la información más reciente y relevante en sus pantallas.

5.3. Desarrollo del panel de control

El panel de control es la interfaz principal que permite a los usuarios interactuar directamente con el sistema. En este panel, los usuarios pueden observar el estado del sistema a través de indicadores luminosos, que proporcionan una visualización clara y rápida de las condiciones operativas actuales. Además, se incluyen selectores que permiten cambiar el modo de funcionamiento, ofreciendo flexibilidad y control sobre los diferentes parámetros y modos operativos del sistema.

5.3.1 Diseño de la interfaz

Utilizando Dash y elementos de Dash DAQ, se ha creado una interfaz gráfica de usuario que incluye diversos elementos interactivos, como botones, controles deslizantes y selectores. La interfaz se ha diseñado para ser intuitiva y fácil de usar, asegurando que los usuarios puedan acceder a las funciones necesarias sin complicaciones. El diseño final del panel de mando se observa en la figura 16 y la figura 17.

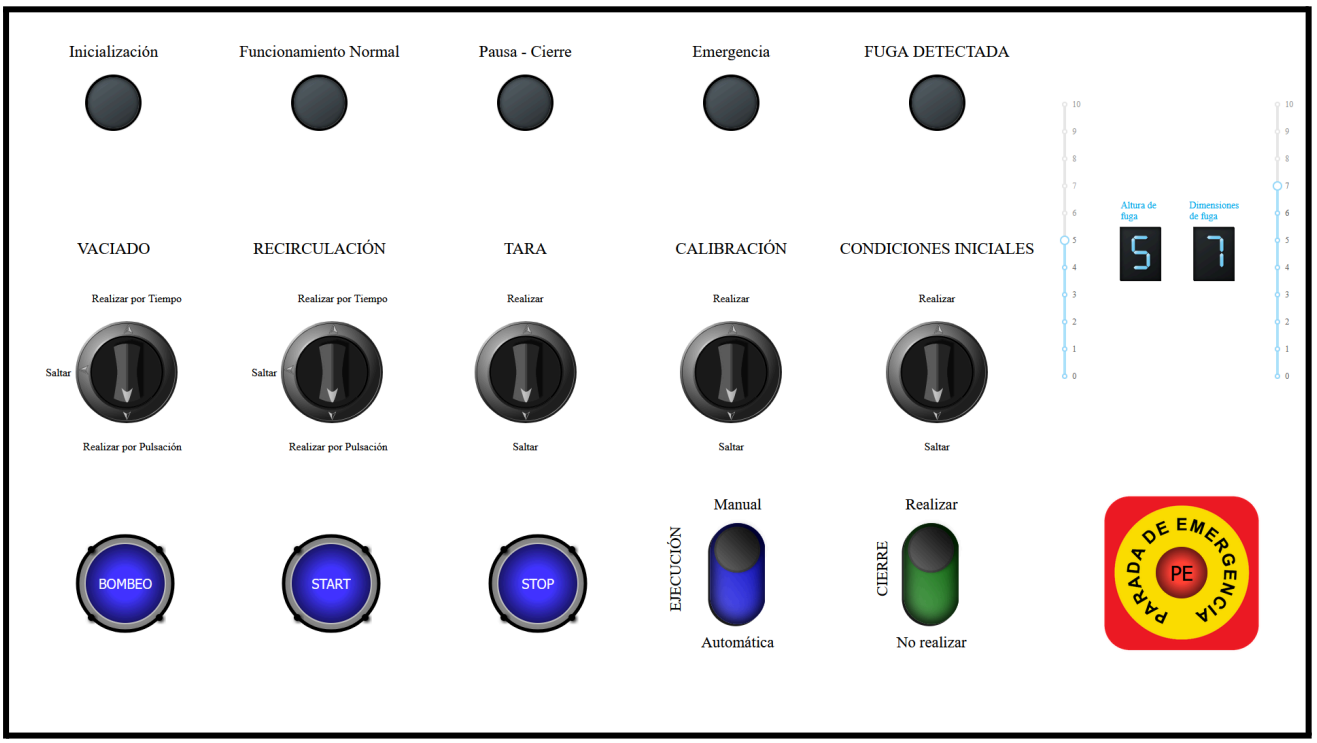


Figura 16. Diseño final del panel de mando

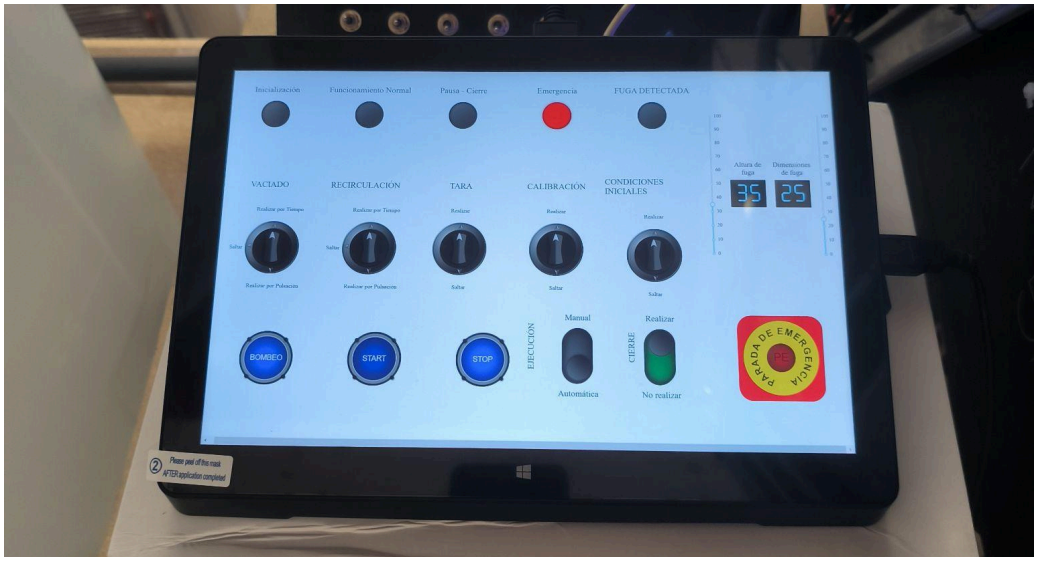


Figura 17. Diseño final del panel de mando en PC PIPO

La disposición principal incluye dos secciones verticales. La primera sección vertical, que ocupa el 70% del ancho de la pantalla, a su vez se distribuye en tres filas horizontales con diferentes elementos:

- **Primera fila:** incluye varios indicadores luminosos (Figura 18) que muestran el estado del sistema, como "Inicialización", "Funcionamiento Normal", "Pausa - Cierre", "Emergencia" y "FUGA DETECTADA".

  Estos indicadores permiten a los usuarios conocer rápidamente el estado actual del sistema.

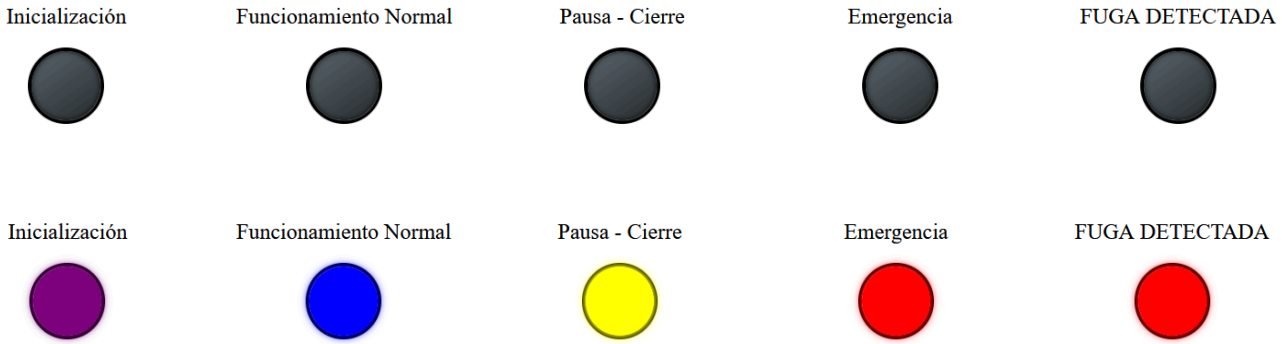


Figura 18. Indicadores luminosos del panel de mando

- **Segunda fila:** en esta fila hay selectores esenciales para ajustar el comportamiento del sistema según las necesidades operativas (Figura 19). Los selectores son "VACIADO", "RECIRCULACIÓN", "TARA", "CALIBRACIÓN" y "CONDICIONES INICIALES". Cada uno de estos selectores tienen diferentes opciones de configuración.

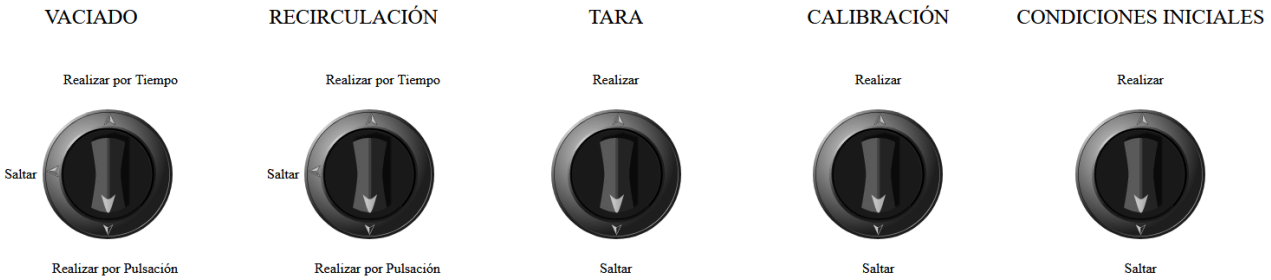


Figura 19. Selectores del panel de mando

- **Tercera fila:** como se observa en la figura 20, esta fila incluye botones para acciones específicas como "BOMBEO", "START" y "STOP". También hay interruptores para cambiar entre modos de ejecución ("Manual" y "Automática") y para realizar acciones de cierre. Cuando se pulsa estos botones tienen un cambio visual que permite conocer el estado en el que se encuentran.

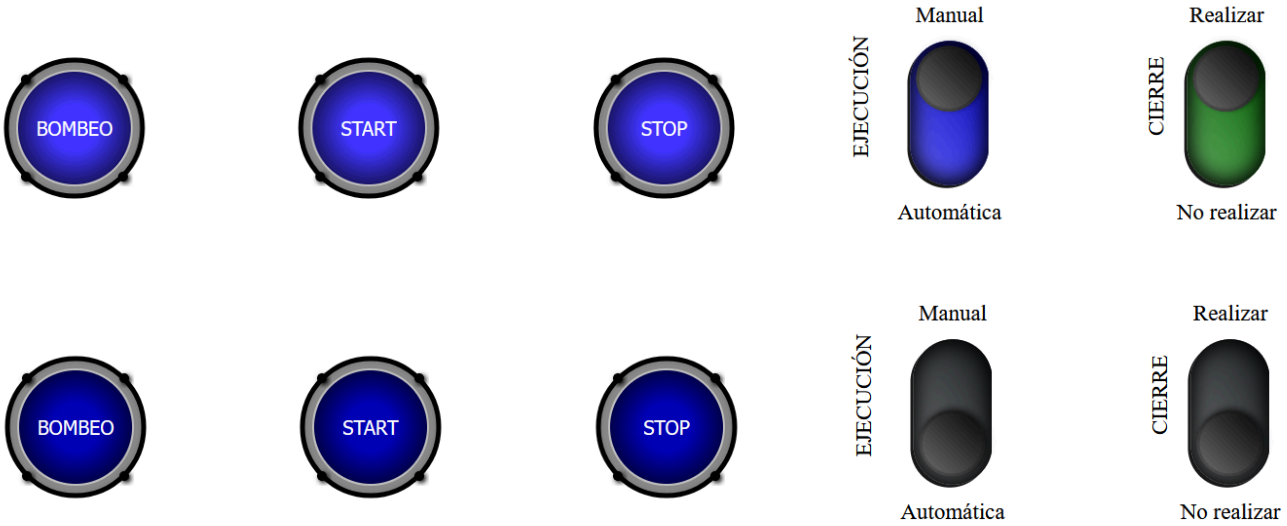


Figura 20. Interruptores del panel de mando

Por otro lado tenemos la segunda sección vertical que ocupa el 30% del ancho, los elementos que se encuentran en esta división son:

- **Sliders de configuración de fugas (Figura 21):** se trata de dos controles deslizantes con sus respectivos displays que simulan dos potenciómetros lineales, estos controles permiten ajustar parámetros específicos como la "Altura de fuga" y las "Dimensiones de fuga". La altura de fuga indica la altura física en un depósito en la que se produce un agujero por el que se fuga el combustible y la dimensión de fuga representa el tamaño del agujero por el que se fuga el combustible.

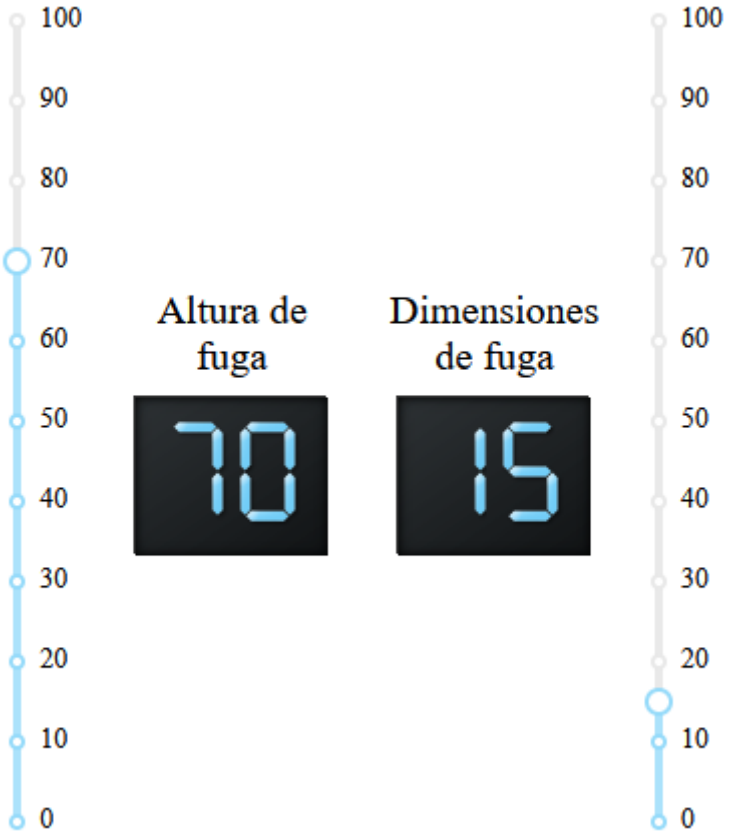


Figura 21. Controles deslizantes del panel de mando

- **Parada de Emergencia (Figura 22):** en la parte baja se encuentra una seta de emergencia que sirve para detener inmediatamente todas las operaciones en caso de una situación crítica.

Figura 22. Parada de emergencia del panel de mando (sin accionar y accionada)

5.3.2 Funcionalidad y actualización en tiempo real

El panel de control se actualiza en tiempo real gracias a un componente de intervalo (`dcc.Interval`) que llama a funciones de actualización a intervalos de tiempo regulares. La función de callback (`update_data`) se encarga de recibir datos del microcontrolador Arduino a través de una conexión serial utilizando la biblioteca `pyserial`. Los datos recibidos se procesan y se utilizan para actualizar los indicadores LED.

5.3.3 Manejo de eventos y acciones

Diversos callbacks de Dash gestionan las interacciones del usuario con los elementos de la interfaz, algunos elementos cambian su apariencia tras ser pulsados:

- **Selectores y Botones**: Las funciones de callback permiten rotar selectores y cambiar estados de botones, enviando comandos específicos al Arduino para modificar el comportamiento del sistema.

- **Parada de Emergencia**: Cambia de color para indicar si está activada o no, y envía una señal al Arduino para detener todas las operaciones en caso de emergencia.

5.4. Desarrollo del panel de monitorización

El panel de monitorización es la interfaz del sistema SCADA, diseñada para proporcionar a los usuarios una visión detallada y en tiempo real del estado del sistema, así como acceso a datos históricos para análisis y diagnóstico. Además, incluye herramientas de depuración y alertas.

El desarrollo de este panel se llevó a cabo utilizando herramientas avanzadas como Dash y Plotly, que permiten la creación de visualizaciones interactivas y dinámicas. La interfaz del panel de monitorización se llevó a cabo con un enfoque en la claridad y la usabilidad. Se crearon componentes visuales que permiten una interacción intuitiva y eficiente. La disposición general incluye tres paneles principales, cada uno de estos paneles se enfoca en diferentes aspectos del monitoreo y control del sistema, estos son:

- Panel 1 - Estado de la planta
- Panel 2 - Históricos
- Panel 3 - Depuración, Alarmas y Alertas

5.4.1 Panel 1 - Estado de la planta

El panel de "Estado de la Planta", que se muestra en la figura 23, ofrece una visualización en tiempo real de los parámetros operativos del sistema.

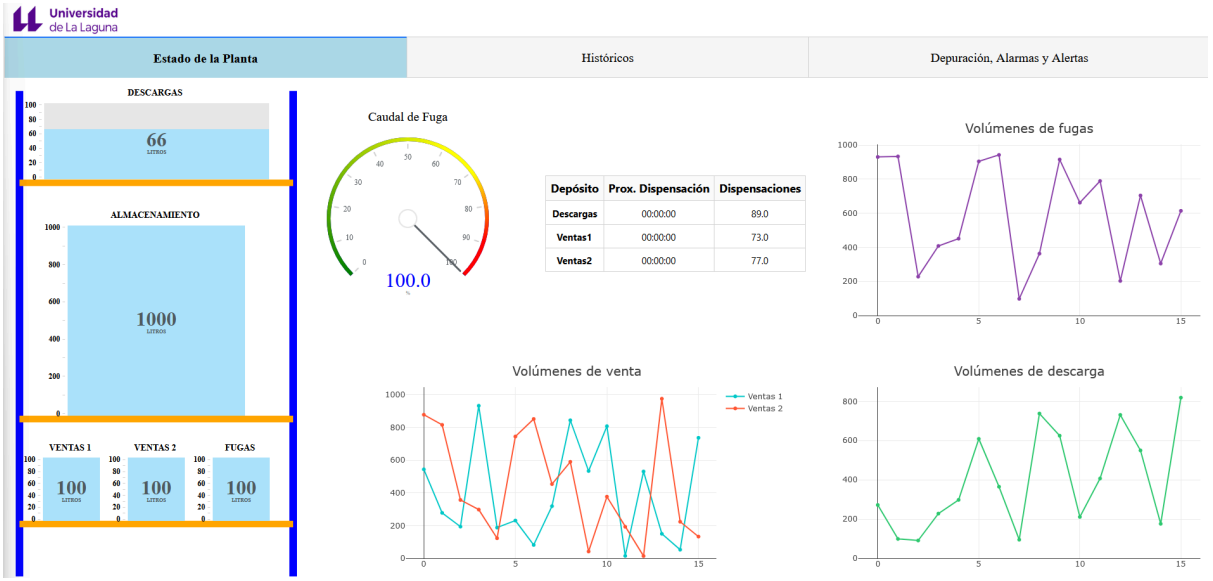


Figura 23. Diseño final del panel de monitorización. Panel 1 - Estado de la planta

La disposición de este panel incluye:

1. **Representación del prototipo:** como se observa en la figura 24, se representan los diferentes depósitos del prototipo (descargas, almacenamiento, ventas 1, ventas 2 y fugas). Estos elementos (`daq.Tank`) permiten a los usuarios ver de un vistazo los niveles actuales de cada depósito.

   En el panel físico, había LEDs asociados a cada depósito para indicar diferentes estados. Por ejemplo, durante el vaciado, el LED encendido correspondía al depósito que se estaba vaciando. En la representación digital, el color del depósito cambia para indicar estas acciones, facilitando la identificación rápida del estado de cada depósito.

Figura 24. Representación del prototipo (depósitos)

2. **Visualización del caudal de fuga (Figura 25):** se incluye un medidor de caudal de fuga que muestra visualmente la magnitud porcentual de las fugas. Este medidor se implementa utilizando el componente `daq.Gauge` de Dash.

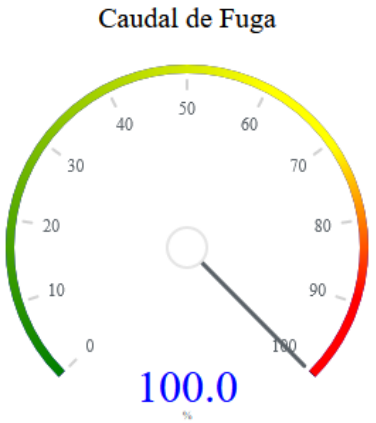


Figura 25. Representación del caudal de fuga

3. **Gráficos en Tiempo Real:** en las figuras 26, 27 y 28 se pueden observar los gráficos de volúmenes de descarga, fugas y ventas que se actualizan continuamente, proporcionando una visión dinámica del comportamiento del sistema. Estos gráficos ayudan a los usuarios a identificar rápidamente cambios en las condiciones operativas. (`dcc.Graph`).

Figura 26. Representación en tiempo real de los volúmenes de descarga

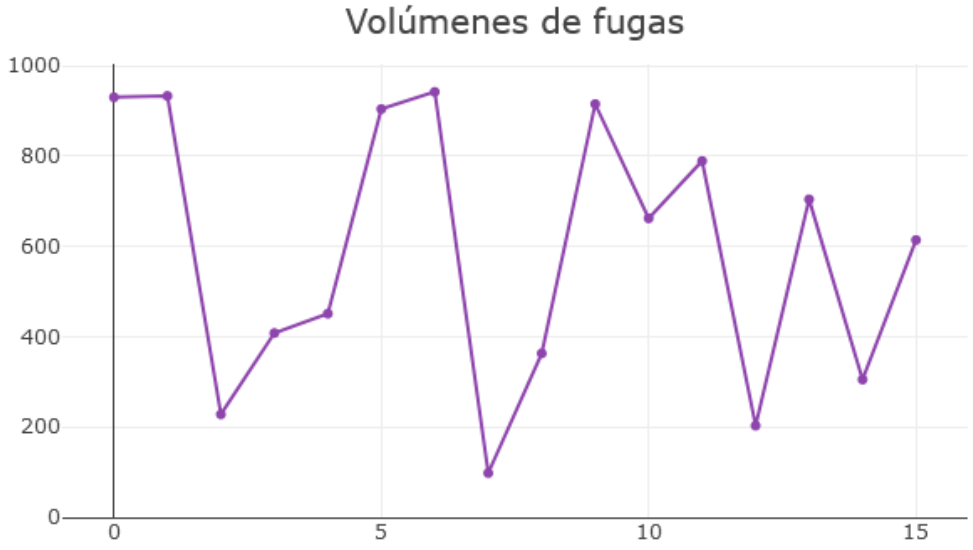


Figura 27. Representación en tiempo real de los volúmenes de fuga

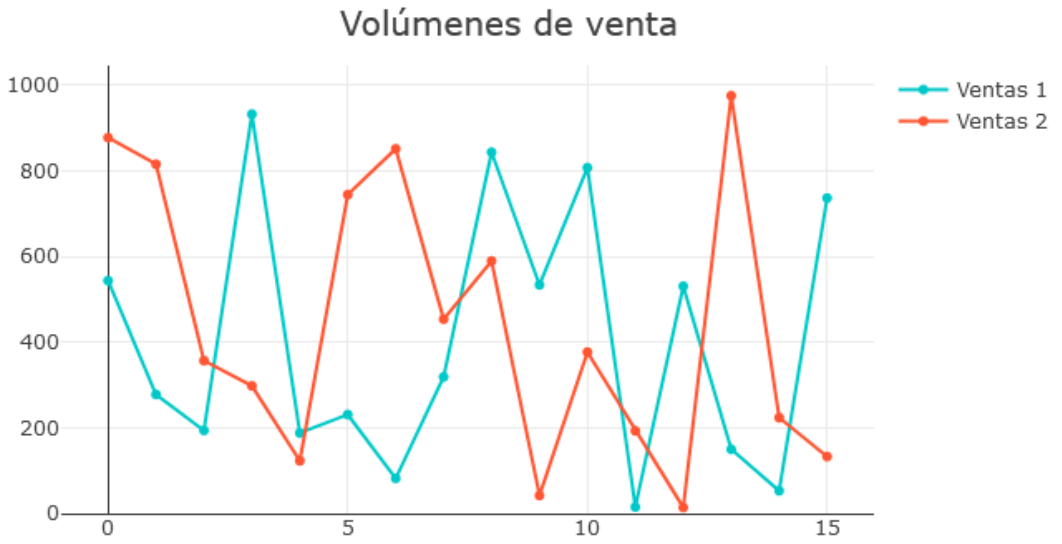


Figura 28. Representación en tiempo real de los volúmenes de venta

4. **Tabla de Próximas Dispensaciones:** la tabla ilustrada en la figura 29 muestra información sobre las próximas dispensaciones, incluyendo el tiempo restante y el número de dispensaciones realizadas. El elemento usado es (`dash_table.DataTable`)

Depósito	Prox. Dispensación	Dispensaciones
Descargas	00:00:00	89.0
Ventas1	00:00:00	73.0
Ventas2	00:00:00	77.0

Figura 29. Tabla de dispensaciones

5.4.2 Panel 2 - Históricos

El panel de "Históricos", mostrado en la figura 30, está diseñado para proporcionar acceso a registros históricos del sistema, permitiendo análisis detallados y generación de informes. Este panel es fundamental para la revisión y evaluación del rendimiento del sistema a lo largo del tiempo.

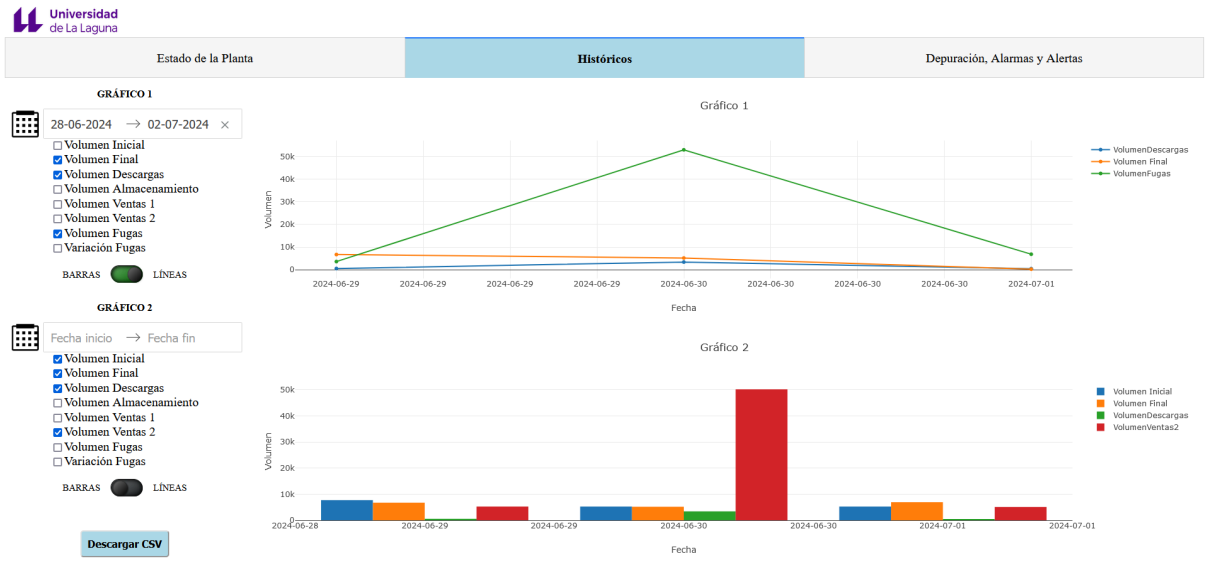


Figura 30. Diseño final del panel de monitorización. Panel 2 - Históricos

La disposición de este panel se basa en dos gráficos principales (Figura 33) que muestran los datos históricos seleccionados. Estos gráficos se actualizan dinámicamente en función de las opciones de filtro y selección realizadas por el usuario. La visualización clara y detallada de estos gráficos permite un análisis profundo de las tendencias y patrones de comportamiento del sistema. Para cada gráfico existe un panel de configuración.

A) **Opciones de filtro de fecha:** los usuarios pueden seleccionar rangos de fechas para filtrar los datos históricos utilizando selectores de fecha (`dcc.DatePickerRange`). Este filtro permite analizar datos específicos dentro de un período determinado La selección de fechas es intuitiva y se realiza a través de un calendario interactivo.

B) Selección de variables y tipo de gráfico (Figura 31): se proporcionan listas de verificación (`dcc.Checklist`) para que los usuarios seleccionen las variables que desean visualizar en los gráficos históricos. Además, hay un interruptor (`daq.BooleanSwitch`) que permite cambiar entre gráficos de líneas y gráficos de barras, ofreciendo flexibilidad en la presentación de los datos.

Figura 31. Elementos de configuración para los gráficos históricos

Para la selección de fechas se utiliza el calendario interactivo mostrado en la figura 32.

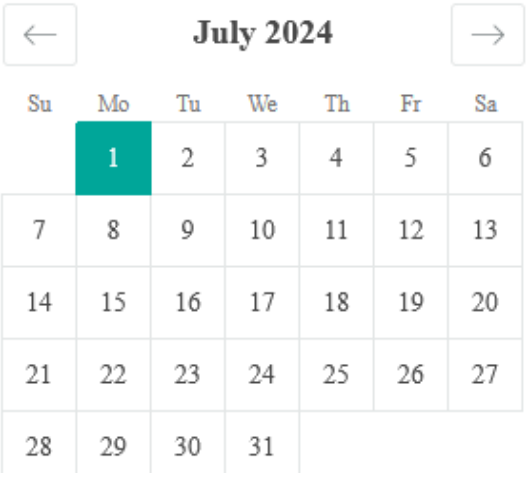


Figura 32. Calendario interactivo para la selección de fechas

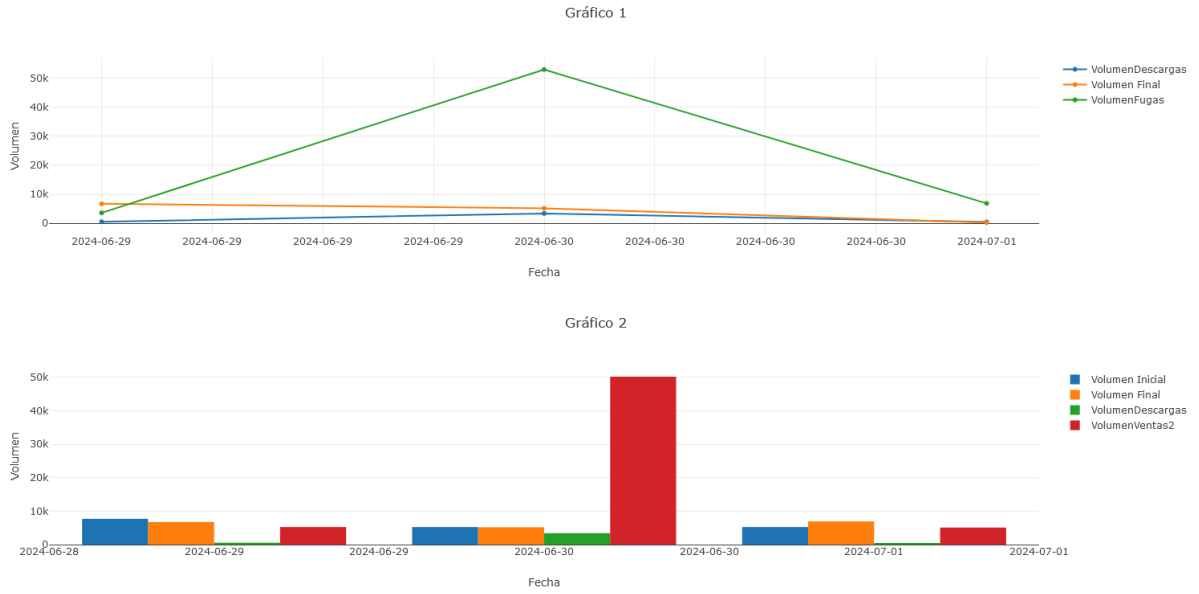


Figura 33. Gráficos de valores históricos

Además, se incluye un botón que permite descargar los datos históricos en formato CSV, representado en la figura 34. Esta función es crucial para el almacenamiento, análisis posterior y compartición de datos con otros equipos o para la generación de informes detallados. Al hacer clic en el botón de descarga, los usuarios pueden obtener un archivo CSV con los datos del sistema.

Figura 34. Botón para descargar el archivo CSV

5.4.3 Panel 3 - Depuración, alarmas y alertas

El panel de "Depuración" mostrado en la figura 35, está diseñado para proporcionar acceso a los registros detallados de eventos y operaciones de los diferentes componentes del sistema, permitiendo un análisis exhaustivo y la resolución de problemas. Este panel es esencial para identificar y solucionar posibles fallos o anomalías en el funcionamiento del sistema

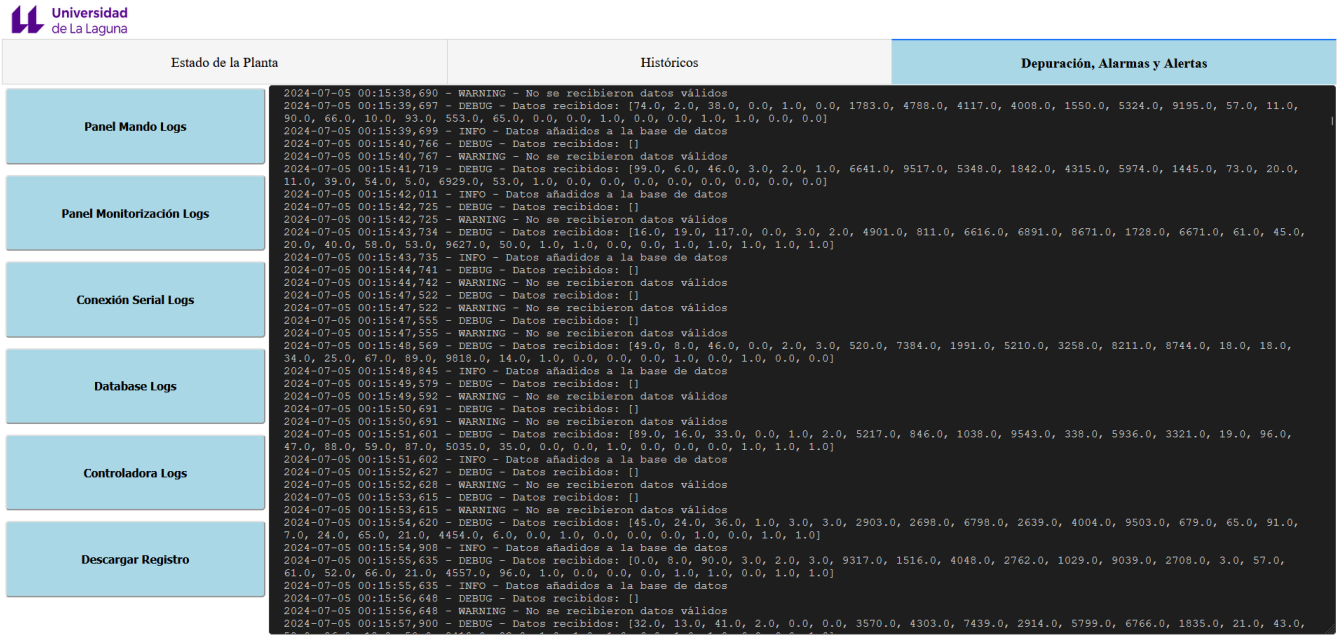


Figura 35. Diseño final del panel de monitorización. Panel 3 - Depuración, Alarmas y Alertas

La disposición de este panel se basa en dos secciones diferenciadas: en la zona izquierda, hay diferentes botones que permiten seleccionar la información que se desea mostrar, como registros del panel de mando, monitorización, conexión serial, base de datos y controladora, así como la opción de descargar todos los registros. La sección de la derecha es un cuadro de texto modificado con apariencia de consola, donde se muestra la información detallada seleccionada.

1. Panel Mando Logs: este botón permite acceder a los registros del panel de mando. Aquí se almacenan todos los eventos y comandos enviados desde el panel de control, lo que ayuda a rastrear acciones de los usuarios y el estado de los comandos.

2. Panel Monitorización Logs: accede a los registros del panel de monitorización. Incluye todos los eventos y datos recopilados por el sistema de monitorización, permitiendo un seguimiento detallado de las condiciones operativas y cualquier cambio en los parámetros del sistema.

3. Conexión Serial Logs: proporciona acceso a los registros de la conexión serial. Estos logs son cruciales para diagnosticar problemas de comunicación entre el sistema SCADA y los dispositivos de hardware conectados. Incluyen información sobre errores de transmisión y recepción de datos.

4. Database Logs: muestra los registros relacionados con la base de datos del sistema. Aquí se pueden encontrar detalles sobre las operaciones de lectura y escritura de datos, errores de base de datos y transacciones importantes que pueden afectar el funcionamiento del sistema.

5. Controladora Logs: permite mostrar información crucial sobre el funcionamiento de la unidad de control principal, incluyendo errores, advertencias y eventos críticos que pueden influir en la estabilidad y eficiencia del sistema. Es muy importante para poder conocer el estado actual tanto de la controladora como de los diferentes depósitos.

6. Descargar Registro: este botón permite descargar todos los registros. Es una función útil para realizar análisis detallados offline, compartir los logs con otros equipos de soporte o archivar registros históricos para futuros análisis.

5.5. Comunicación arduino - SCADA

La comunicación entre el microcontrolador Arduino y el sistema SCADA es un aspecto crítico para el correcto funcionamiento del sistema. En este proyecto, se ha implementado una comunicación serial a través de una conexión USB. Esta sección describe la implementación y funcionamiento de la comunicación entre el Arduino y el sistema SCADA.

Para establecer la conexión serial, se configura el puerto de comunicación (en este caso, `COM5`) y se define una velocidad de transmisión de 9600 baudios. Se utiliza la biblioteca `pyserial` de Python para gestionar la comunicación serial.

La función `start_serial_connection` (Figura 36) intenta establecer una conexión con el puerto serial especificado. En caso de fallar, lo intenta nuevamente después de un segundo. Esto asegura que el sistema continúe intentando conectarse hasta que la conexión se establezca correctamente.

```python
def start_serial_connection():
    puerto_com = 'COM5'
    ser = None
    while ser is None:
        try:
            ser = serial.Serial(puerto_com, 9600, timeout=1)
            logger.debug("Conexión serial establecida en %s", puerto_com)
        except Exception as e:
            logger.error("Error en start_serial_connection: %s", e)
            time.sleep(1)  # Espera un segundo antes de intentar nuevamente
    return ser
```

Figura 36. Definición de función para la conexión serial

La función `read_from_serial` se encarga de leer los datos enviados por el Arduino. Los datos se leen línea por línea y se procesan para convertirlos en un formato numérico adecuado. Para asegurar que los datos se leen continuamente del puerto serial, se utiliza un hilo separado.

La función `continuously_read_serial_data` lee los datos en un bucle infinito y actualiza la variable global `data`. Este hilo se ejecuta en segundo plano, permitiendo que la aplicación principal siga funcionando sin interrupciones.

Configuración MQTT

Para establecer la conexión MQTT, se utilizó el broker público de HiveMQ, configurando la dirección del broker, el puerto, el tópico de suscripción y las credenciales de usuario. Estos parámetros son esenciales para asegurar que el cliente pueda conectarse correctamente y comunicarse con el broker.

- **Broker MQTT:** broker.hivemq.com
- **Puerto:** 1883
- **Tópico de Suscripción:** proyecto/monitorizacion
- **Usuario y Contraseña:** Se definieron credenciales específicas para el acceso seguro.

Una vez configurado, el cliente se conecta al broker y se inicia un bucle que mantiene la conexión activa y gestiona el envío y la recepción continua de mensajes.

Se define una funcionalidad específica para publicar mensajes en el tópico MQTT. Esta capacidad permite que el sistema SCADA envíe datos y comandos a otros componentes de la red IoT, facilitando el control y monitoreo del sistema.

Además, desde el panel de monitorización se configura otro cliente que cada vez que se recibe un mensaje en el tópico suscrito, una función se activa para procesar el mensaje. Esta función actualiza los datos del panel de monitorización y de la base de datos, permitiendo así la visualización y manejo de los datos en tiempo real.

5.6. Registros

El registro de eventos y errores (logging) es una parte crucial de cualquier sistema informático, ya que permite rastrear y diagnosticar problemas, monitorear el estado del programa y realizar seguimiento de eventos. En este proyecto, se han implementado varias funcionalidades de logging utilizando la biblioteca estándar de Python `logging`.

Se ha configurado un logger para cada componente crítico del sistema SCADA, incluyendo la comunicación serial, los paneles de control y monitorización, y la base de datos. Cada logger se encarga de registrar eventos específicos de su componente, facilitando la identificación y solución de problemas.

Para cada componente, se configura un logger específico que escribe los registros en archivos de texto separados. A continuación, en la figura 37, se muestra como ejemplo de configuración el logger correspondiente a la base de datos:

```
log_file = os.path.join('logs', 'database_logs.txt')

# Configurar el logger
logger = logging.getLogger('database_logger')
logger.setLevel(logging.INFO)
file_handler = logging.FileHandler(log_file)
formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
file_handler.setFormatter(formatter)
logger.addHandler(file_handler)
```

Figura 37. Configuración del logger de la base de datos

El formato de los mensajes de registro es crucial para asegurar que los logs sean claros y útiles para el diagnóstico y análisis. Los mensajes de registro se formatean para incluir la fecha y hora del evento, el nivel de registro y el mensaje en sí. Esto se configura mediante el formatter del logger.

El proceso de logging incluye el uso de distintos niveles de registro según la gravedad del evento a registrar, los niveles utilizados son:

- **DEBUG (Figura 38):** Mensajes detallados que son útiles para la depuración del sistema.

```
logger.debug(f"Datos recibidos: {data}")
```

Figura 38. Ejemplo registro tipo *debug*

- **INFO (Figura 39)**: Mensajes informativos generales sobre el funcionamiento normal del sistema

```
logger.info("Conexión serial establecida en %s", puerto_com)
```

Figura 39. Ejemplo registro tipo *info*

- **WARNING (Figura 40)**: Mensajes que indican la presencia de posibles problemas que no impiden del todo el funcionamiento del sistema, pero que deben ser revisados ya que afectan a funcionalidades específicas.

```
logger.warning("No se puede convertir los datos a float. Datos: %s", data)
```

Figura 40. Ejemplo registro tipo *warning*

- **ERROR (Figura 41)**: Mensajes que reportan errores críticos que afectan el funcionamiento del sistema.

```
logger.error("Error en start_serial_connection: %s", e)
```

Figura 41. Ejemplo registro tipo *error*

Todos estos registros se almacenan en diferentes archivos de texto según el componente al que pertenecen. Es importante destacar el archivo `controladora_logs.txt`, que es fundamental ya que muestra la depuración del propio Arduino Mega, registrando los estados y sus cambios, así como los estados de los depósitos.

6. Pruebas y validación

6.1. Metodología de pruebas

Para asegurar la fiabilidad y eficacia del sistema SCADA desarrollado se implementó una metodología de pruebas exhaustiva. Inicialmente, se realizaron pruebas unitarias a nivel de código para verificar la funcionalidad de cada elemento y de cada módulo individualmente, abarcando funciones de lectura de sensores, envío de comandos y almacenamiento de datos. Posteriormente, los módulos validados se integraron y se realizaron pruebas de integración para asegurar que trabajen correctamente en conjunto.

Las pruebas de sistema se llevaron a cabo evaluando el sistema completo bajo condiciones operativas reales simuladas, verificando el correcto funcionamiento de las diferentes funciones del sistema, incluyendo el panel de control y el panel de monitorización.

Para las pruebas de funcionamiento del panel de monitorización sistema SCADA, se utilizó un Arduino UNO programado para enviar datos de manera aleatoria con el mismo formato y características que el Arduino Mega del prototipo, de esta manera se puede comprobar que el sistema SCADA propuesto recibe la información, la muestra y la actualiza de manera correcta. A su vez, para probar los diferentes elementos del panel de control, se conectaron diferentes actuadores al arduino de pruebas como pueden ser leds o zumbadores, de manera que se activen según el elemento a probar en el panel de control.

Una vez probado el sistema en condiciones simuladas, se procede al comienzo de pruebas con el sistema real. Para ello, lo primero que se hizo fue adaptar el código original de la controladora Arduino Mega para el nuevo proyecto. De manera resumida, las adaptaciones que hubo que realizar fueron:

- Modificar la función que envía los datos por el puerto serie para incluir más cantidad de información relevante como pueden ser los estados de los indicadores luminosos o los estados de los propios depósitos.
- Realizar un cambio en las entradas del sistema, en vez de leer los pines del panel de control físico ya existente se leen datos de una nueva variable tipo struct que almacena el estado de cada elemento del panel de control. Esta variable tipo struct se actualiza constantemente con la información recibida por el puerto serie.
- Se añadieron nuevos mensajes de depuración en las funciones correspondientes para poder enviar información al SCADA acerca de las máquinas de estado tanto de la controladora principal como de los depósitos.

6.2. Resultados de las pruebas

Los resultados de las pruebas fueron analizados para identificar cualquier problema y realizar las correcciones necesarias. En las pruebas unitarias, se identificaron y corrigieron los errores que iban surgiendo en el código, mejorando la estabilidad y precisión de las funciones básicas. Las pruebas de integración de componentes concluyeron en que los módulos se comunican correctamente y funcionan en conjunto sin problemas significativos.

Utilizar el Arduino UNO para las pruebas de funcionamiento del sistema SCADA de manera simulada permitió el desarrollo del sistema SCADA de la mejor manera posible, configurando los diferentes elementos de manera progresiva y escalable.

El sistema completo funcionó correctamente bajo condiciones operativas simuladas, con todas las funcionalidades tanto de control como de monitorización operando como se esperaba.

Para implementar el sistema SCADA con la controladora del prototipo, fue necesario adaptar ciertas características de su programación, tal y como se ha comentado anteriormente. Este proceso resultó ser complejo y dificultó la integración del sistema Dash con el prototipo.

A pesar de realizar diversas pruebas, tanto de control como de monitorización, el sistema no logró funcionar completamente de manera óptima. Con el fin de mejorar la integración, se ha trabajado en la depuración de errores y en el seguimiento de los estados de la controladora a través del panel Dash, permitiendo así un conocimiento más preciso de las condiciones y estados del prototipo.

Estas pruebas consistieron en tratar de controlar el prototipo desde el panel de control y a la vez monitorizarlo desde el monitor externo. La monitorización se realizó de manera correcta, el panel de estado de la planta se actualizaba constantemente y la información se registraba de manera adecuada en la base de datos para su posterior representación en el panel de históricos. Cabe destacar que parece que se necesita una correcta calibración del prototipo ya que pese a que los valores enviados por la controladora se mostraban correctamente, estos no eran consistentes y no reflejaban la realidad.

Por otro lado, para el panel de control se trató de realizar un proceso completo de funcionamiento. Las órdenes de control se enviaban de manera correcta a la controladora y luego se almacenaban en una variable de tipo struct que es la que se leerá en lugar de los pines físicos de la controladora. Sin embargo, no se logró realizar el proceso completo de funcionamiento seguramente debido a un problema al integrar las órdenes del SCADA con las funciones existentes en el arduino mega. Algunos elementos como la parada de emergencia funcionan perfectamente mientras que otros funcionan de manera intermitente.

7. Presupuesto

Tabla 1. Presupuesto de ejecución material (PEM)

PRESUPUESTO DE EJECUCIÓN MATERIAL (PEM)					
1. Material					
Código	**Descripción**	**Cantidad**	**Precio**	**Subtotal**	**Importe total**
01.01	Lenovo D27-40 LED	1	130,00 €	130,00 €	130,00 €
2. Mano de Obra					
Horas			**Coste/horas (€/h)**	**Total**	
300			48,00 €	14.400,00 €	
PEM (sin aplicar impuestos)				14.530,00 €	

Tabla 2. Presupuesto total

PRESUPUESTO TOTAL	
Descripción	**Importe**
Hardware	130,00 €
Mano de obra	14.400,00 €
PEM	**14.530,00 €**
Costes Indirectos (13%)	1.888,90 €
Beneficio industrial (6%)	871,80 €
Subtotal	**17.290,70 €**
IGIC (7%)	1.210,35 €
TOTAL	**18.501,05 €**

8. Conclusiones y trabajos futuros

8.1. Conclusiones

El desarrollo del sistema SCADA para la detección de fugas en una estación de servicio ha logrado cumplir con gran parte de los objetivos planteados inicialmente. Las principales conclusiones del proyecto son:

El sistema SCADA implementado ha demostrado ser eficaz en la supervisión de variables e información de funcionamiento del sistema. La capacidad de monitorear en tiempo real y registrar información histórica ha sido validada a través de pruebas exhaustivas.

Las interfaces de usuario desarrolladas con Dash facilitan la interacción con el sistema, permitiendo a los usuarios supervisar y controlar las operaciones de manera intuitiva y eficiente. La visualización de datos en tiempo real y la capacidad de acceso a datos históricos han mejorado significativamente la obtención de información.

La comunicación serial ha proporcionado una solución flexible y de bajo costo para el prototipo. Aunque esta solución es adecuada para el alcance del proyecto, se ha identificado la necesidad de considerar alternativas más robustas para aplicaciones industriales de mayor escala.

El almacenamiento de datos en archivos CSV ha sido suficiente para el desarrollo y validación inicial del sistema. No obstante, para mejorar la eficiencia y seguridad en el manejo de grandes volúmenes de datos, se recomienda migrar a una base de datos relacional en futuros desarrollos.

Además, es importante destacar que los datos generados son fundamentales como entrada para otros sistemas avanzados, como mecanismos de inteligencia artificial (IA). Estos sistemas pueden utilizar los datos históricos y en tiempo real para identificar patrones y realizar

predicciones precisas sobre posibles fugas, mejorando aún más la capacidad de respuesta y prevención de este tipo de sistemas.

Las pruebas realizadas con datos generados de manera aleatoria como entrada del SCADA han confirmado el correcto funcionamiento del sistema, tanto del panel de mando como del panel de monitorización. Validar el sistema de esta forma ha sido fundamental para asegurar los primeros pasos para el uso del SCADA en el prototipo con condiciones operativas reales.

Sin embargo, en cuanto al panel de mando, las pruebas del SCADA con el prototipo real no se han podido realizar correctamente debido a problemas con la integración completa del sistema SCADA y la programación de la controladora.

A pesar de estas dificultades, los resultados obtenidos con datos simulados son prometedores y proporcionan una base sólida para futuras pruebas y mejoras. Gracias a la monitorización de estados del sistema a través del SCADA, es posible identificar y abordar estos problemas. Es crucial que en fases posteriores se solucionen estos problemas, permitiendo así una validación completa del sistema en condiciones operativas reales.

Las siguientes etapas deben enfocarse en resolver las limitaciones actuales, realizar los ajustes necesarios y llevar a cabo pruebas exhaustivas con el prototipo real para garantizar la robustez y fiabilidad del sistema SCADA.

Para una revisión detallada del proyecto, incluyendo código fuente, documentación y vídeos del funcionamiento del SCADA con los datos simulados, todo el contenido se encuentra disponible en el repositorio de GitHub del proyecto [10].

8.2. Conclusions

The development of the SCADA system for leak detection at the service station has accomplished most of the initially set objectives. The main conclusions of the project are:

The SCADA system implemented has proven to be effective in monitoring variables and system operation information. The ability to monitor in real-time and record historical information has been validated through extensive testing.

The user interfaces developed with Dash facilitate interaction with the system, allowing users to monitor and control operations intuitively and efficiently. Real-time data visualization and the ability to access historical data have significantly improved information retrieval.

Serial communication has provided a flexible and low-cost solution for the prototype. While this solution is adequate for the scope of the project, the need to consider more robust alternatives for larger-scale industrial applications has been identified.

Data storage in CSV files has been sufficient for the initial development and validation of the system. However, to improve efficiency and security in handling large volumes of data, it is recommended to migrate to a relational database in future developments.

In addition, it is important to note that the data generated is essential as input for other advanced systems, such as artificial intelligence (AI) mechanisms. These systems can use historical and real-time data to identify patterns and make accurate predictions about potential leaks, further improving the responsiveness and prevention of such systems.

Tests using randomly generated data as input to the SCADA have confirmed the correct operation of the system, both the control panel and the monitoring panel. Validating the system in this way has been

fundamental to ensure the first steps for the use of the SCADA in the prototype under real operating conditions.

However, as far as the control panel is concerned, the SCADA tests with the real prototype could not be carried out correctly due to problems with the complete integration of the SCADA system and the programming of the controller.

Despite these difficulties, the results obtained with simulated data are promising and provide a solid basis for further testing and improvements. By monitoring system states through SCADA, it is possible to identify and address these problems. These problems must be solved in subsequent phases, allowing a full validation of the system under real operating conditions.

The next stages should focus on resolving the current limitations, making the necessary adjustments, and carrying out extensive testing with the real prototype to ensure the robustness and reliability of the SCADA system.

All the project's content, including source code, documentation, and videos of SCADA operation with simulated data, can be found on the project's GitHub repository [10].

8.3. Limitaciones del proyecto

El proyecto presenta algunas limitaciones. La comunicación serial utilizada, aunque suficiente para el prototipo, podría no ser adecuada para aplicaciones industriales de mayor escala que requieran una transmisión de datos más robusta y rápida.

Además, el almacenamiento de datos en archivos CSV, si bien es práctico para el desarrollo inicial, podría beneficiarse de una migración a una base de datos relacional para un manejo más eficiente y seguro de grandes volúmenes de datos.

8.4. Propuestas de mejoras y trabajos futuros

A continuación se detallan las propuestas de mejora y trabajos futuros identificados para optimizar y ampliar las capacidades del sistema SCADA desarrollado:

1. ***Resolución de problemas de integración***

Abordar y solucionar los problemas de integración entre el SCADA y el prototipo real. Esto incluye finalizar la implementación y optimización del funcionamiento del panel de mando con el prototipo.

2. ***Mejora de la comunicación serial***

Considerar la adopción de protocolos que ofrecen mayores velocidades de transmisión, mejor gestión de errores y capacidad para conectar múltiples dispositivos en una red compleja. Esto permitirá una comunicación más fiable y eficiente entre el SCADA y el prototipo.

3. ***Migración a una base de datos relacional***

La migración de los datos almacenados en archivos CSV a una base de datos relacional (como MySQL, PostgreSQL o SQLite) mejorará la eficiencia y seguridad en el manejo de grandes volúmenes de datos. Las bases de datos relacionales ofrecen mejores capacidades de consulta, integridad de datos y escalabilidad, lo que facilitará el análisis y gestión de la información a largo plazo.

4. ***Añadir un panel adicional en el panel de monitorización***

Incorporar un panel adicional en el panel de monitorización que muestre las diferentes máquinas de estado del sistema y de cada depósito. Este panel proporcionará una representación visual clara del estado actual del sistema, facilitando la identificación rápida de cualquier cambio o anomalía. Los usuarios podrán ver de un vistazo en qué estado se encuentran las distintas partes del sistema y si existen problemas entre cambios de estado.

5. ***Implementación de un sistema de monitorización y control remoto***

Desarrollar un sistema para la monitorización y control remoto del sistema SCADA. Esto permitirá a los usuarios acceder y gestionar el sistema desde ubicaciones remotas, aumentando la flexibilidad y capacidad de respuesta ante eventos. Para el correcto uso de manera remota del sistema, será necesario añadir cámaras para proporcionar una visión en tiempo real del entorno físico del prototipo. Las cámaras pueden integrarse en el sistema SCADA de este proyecto, permitiendo a los usuarios visualizar el estado del prototipo y las condiciones operativas de manera remota.

9. Bibliografía

[1] Sigut, M., Alayón, S., & Hernández, E. (2014). Applying pattern classification techniques to the early detection of fuel leaks in petrol stations. *Journal of Cleaner Production, 80,* 262-270. https://doi.org/10.1016/j.jclepro.2014.05.070

[2] Alayón, S., Sigut, M., Arnay, R., & Toledo, P. (2020). Time windows: The key to improving the early detection of fuel leaks in petrol stations. Safety Science, 130, 104874. https://doi.org/10.1016/j.ssci.2020.104874

[3] Arriaga Campos, L. (2020). Diseño e implementación de un sistema autónomo para la simulación de fugas en depósitos (Trabajo Final de Grado). Repositorio institucional de la Universidad de La Laguna. Recuperado de http://riull.ull.es/xmlui/handle/915/20640

[4] Yanes Perez, N. (2022). Diseño e implementación de un sistema autónomo para la detección de fugas en depósitos (Trabajo Final de Máster, Máster Universitario en Ingeniería Industrial). Repositorio institucional de la Universidad de La Laguna. Recuperado de http://riull.ull.es/xmlui/handle/915/31568

[5] Rodríguez Herrera, F. (2022). Simulación y detección de fugas en depósitos de combustible (Trabajo Final de Grado, Grado en Ingeniería Electrónica Industrial y Automática). Repositorio institucional de la Universidad de La Laguna. Recuperado de http://riull.ull.es/xmlui/handle/915/30329

[6] Medina León, G. (2023). Detección de fugas en depósitos de combustible [Repositorio de GitHub]. Recuperado de https://github.com/GregorioML/TFG-DeteccionFugas

[7] Copadata. (2023). ¿Qué es SCADA? Recuperado en julio de 2024, de https://www.copadata.com/es/productos/zenon-software-platform/visualizacion-control/que-es-scada/

[8] Proyecto Arduino. (2019). Uso del puerto serie con Arduino. Recuperado en julio de 2024, de https://proyectoarduino.com/arduino-puerto-serie/

[9] Goebel, T. (2024, 26 de enero). What is MQTT? Twilio. Recuperado en julio de 2024, de https://www.twilio.com/en-us/blog/what-is-mqtt

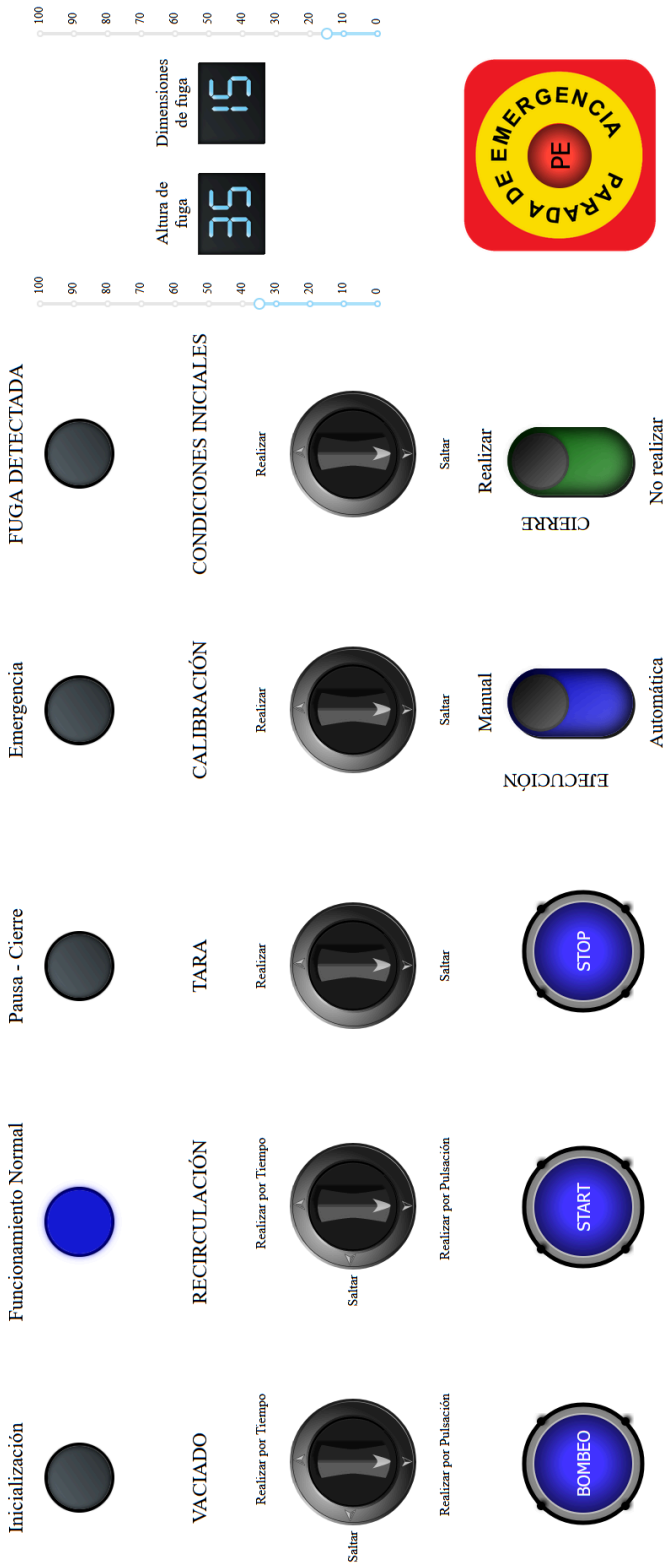
[10] Corona Ledesma, A (2024). *TFM-SCADA-DeteccionFugas*. GitHub. Recuperado en julio de 2024, de https://github.com/AlexCoronaLedesma/TFM-SCADA-DeteccionFugas

II. ANEXO

Índice

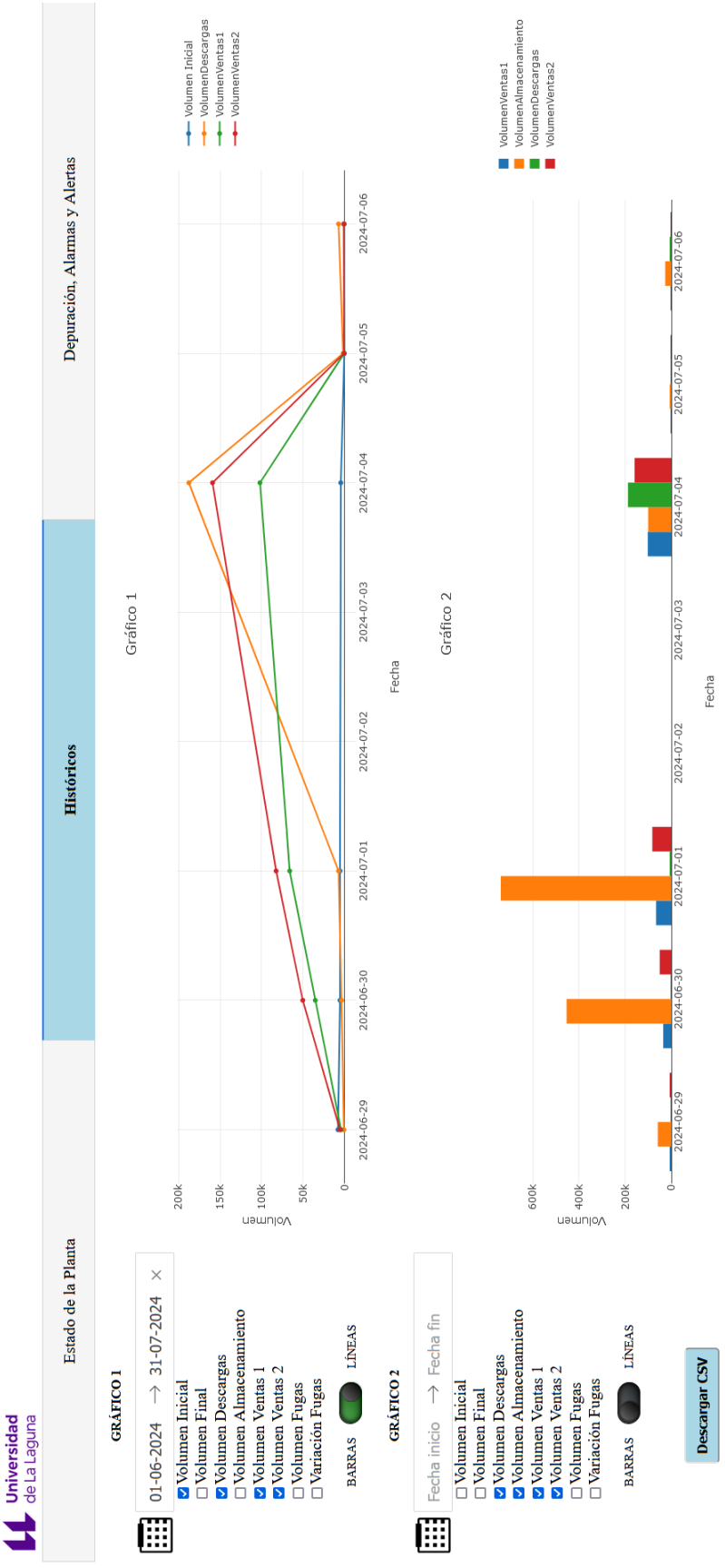
Anexo

1. Panel de Mando



2. Panel de Monitorización - Estado de la planta

3. Panel de Monitorización - Históricos



4. Panel de Monitorización - Depuración, Alarmas y Alertas