UNIVERSITAT DE BARCELONA

Trabajo de fin de grado

GRADO DE INGENIERÍA INFORMÁTICA

Facultad de Matemáticas e Informática
Universidad de Barcelona

DESARROLLO DE UNA PLATAFORMA SOFTWARE PARA LA ADQUISICIÓN Y CONTROL DE SENSORES DE GASES

Pau Chamarro López

Director: Dr. Manuel López de Miguel
Realitzat a: Departament de
Ingeniería Electrónica

Barcelona, Junio 2025

Resumen

Este trabajo se centra en el desarrollo de una aplicación software destinada al control y gestión de una estación experimental de medida de gases, recientemente instalada en el Departamento de Ingeniería Electrónica y Biomédica de la Universidad de Barcelona. Esta estación permite analizar, en condiciones controladas, el comportamiento y la respuesta de distintos sensores de gases, fundamentales para estudios medioambientales.

El principal objetivo del proyecto consiste en desarrollar una herramienta intuitiva y flexible que facilite al investigador la adquisición de datos, el control de dispositivos físicos y la automatización de experimentos con diferentes combinaciones gaseosas. Para ello, se ha desarrollado una plataforma en Python basada en el patrón arquitectónico Modelo-Vista-Controlador (MVC), lo que garantiza una estructura modular y escalable.

Esta estación representa una mejora significativa respecto a soluciones anteriores, tanto en términos de la arquitectura del software como de funcionalidad operativa, lo cual proporciona una base sólida para futuras investigaciones y estudio de nuevos materiales para la detección de gases. En definitiva, esta estación experimental se configura como una herramienta estratégica para avanzar en el conocimiento sobre sensores y el análisis de emisiones gaseosas en entornos controlados.

Resum

Aquest treball se centra en el desenvolupament d'una aplicació de programari destinada al control i gestió d'una estació experimental de mesura de gasos, recentment instal·lada al Departament d'Enginyeria Electrònica i Biomèdica de la Universitat de Barcelona. Aquesta estació permet analitzar, en condicions controlades, el comportament i la resposta de diferents sensors de gasos, fonamentals per a estudis de caràcter mediambiental.

L'objectiu principal del projecte consisteix en desenvolupar una eina intuïtiva i flexible que faciliti a l'investigador l'adquisició de dades, el control de dispositius físics i l'automatització d'experiments amb diferents combinacions gasoses. Per a això, s'ha implementat una plataforma en Python estructurada segons el patró arquitectònic Model-Vista-Controlador (MVC), la qual cosa garanteix una organització modular i escalable del codi.

Aquesta estació representa una millora significativa respecte a solucions anteriors, tant pel que fa a l'arquitectura del programari com a la funcionalitat operativa, proporcionant així una base sòlida per a futures investigacions i estudi de nous materials per a la detecció de gasos. En definitiva, aquesta estació experimental es configura com una eina estratègica per avançar en el coneixement sobre sensors i l'anàlisi d'emissions gasoses en entorns controlats.

Abstract

This work is focused on the development of a software application intended for the control and management of an experimental gas measurement station, recently installed at the Department of Electronic and Biomedical Engineering at the University of Barcelona. The station allows for the analysis of the behaviour and response of various gas sensors under controlled conditions, which are essential for environmental research.

The primary objective of the project is to develop an intuitive and flexible tool that supports researchers in data acquisition, physical device control, and the automation of experiments involving different gas mixtures. To achieve this, a Python-based platform has been implemented using the Model-View-Controller (MVC) architectural pattern, ensuring a modular and scalable software structure.

This station constitutes a significant advancement over previous solutions, both in terms of software architecture and operational capabilities, providing a robust foundation for future research, and the exploration of new materials for gas detection. Ultimately, this experimental setup is positioned as a strategic instrument to further knowledge on sensors and the analysis of gaseous emissions within controlled environments.

Agradecimientos

Quiero agradecer al Departamento de Ingeniería Electrónica y Biomédica de la Universidad de Barcelona por facilitar los recursos necesarios para llevar a cabo este proyecto. También agradezco especialmente al profesor Manel López de Miguel por su orientación y apoyo durante todo el desarrollo. Por último, agradezco a mi familia y pareja por su ánimo constante.

Índice

Capítulo 1: Introducción

1.1 Contexto

Este Trabajo de Final de Grado se desarrolla en el Departamento de Ingeniería Electrónica y Biomédica de la Universidad de Barcelona (UB), bajo la dirección del Dr. Manuel López. La propuesta surge como una continuación de las prácticas externas, en las cuales trabajé en el desarrollo de soluciones basadas en microcontroladores ESP32 [1], implementación de modelos de predicción y comunicación mediante el protocolo MQTT [2].

Durante este período de prácticas, pude trabajar con el ecosistema de sistemas y la interacción entre sensores, controladores y plataformas de análisis de datos. Al finalizar las prácticas, el Dr. López me propuso participar en el rediseño del sistema de control de una estación de medida de gases, con el objetivo de sustituir la arquitectura existente por una solución más modular, flexible y orientada al uso de software libre.

El proyecto representa una evolución importante respecto a la implementación anterior, basada en LabView, que presentaba limitaciones significativas en cuanto a escalabilidad, mantenibilidad y reutilización del código. En cambio, el nuevo sistema ha sido desarrollado en Python, utilizando programación orientada a objetos (POO) y el patrón arquitectónico Model-View-Controller (MVC), lo que permite una separación clara entre la lógica de negocio, la interfaz de usuario y la gestión de los datos. Esta estructura facilita la integración de nuevos módulos, la colaboración entre desarrolladores y la portabilidad del software a otros entornos de investigación.

Desde el punto de vista académico y profesional, este TFG me ha permitido aplicar conocimientos adquiridos a lo largo del grado, especialmente en el diseño de arquitecturas software, automatización y comunicaciones. Además, me ha ofrecido la oportunidad de contribuir al desarrollo de una plataforma robusta, extensible y alineada con las necesidades reales de la investigación científica.

El sistema resultante no solo mejora las capacidades de adquisición y control de la estación experimental, sino que también sienta las bases para futuras ampliaciones y aplicaciones en el estudio del comportamiento de mezclas gaseosas.

1.2 Motivación del problema

Como bien se ha comentado, este proyecto surge del desarrollo de una nueva estación de medida de gases en la Universidad de Barcelona (UB), que pretende superar las limitaciones detectadas en las instalaciones anteriores. La motivación principal radica en la oportunidad de integrar conocimientos científicos junto con herramientas tecnológicas modernas, especialmente en el ámbito del software, para facilitar la adquisición, control y análisis de datos sobre el comportamiento de gases en condiciones ambientales controladas.

Una de las bases fundamentales de esta nueva estación es el desarrollo de un sistema de control modular, basado en el patrón arquitectónico **Model-View-Controller (MVC)**. Esta estructura permite separar de forma clara la lógica de negocio (mediciones y control de gases), la interfaz de usuario y la gestión de los datos, lo cual no solo facilita el mantenimiento y la escalabilidad del código, sino que también favorece a su reutilización en otros proyectos de mayor o menor envergadura. Al trabajar con un diseño desacoplado, es posible sustituir o actualizar componentes individuales sin tener que reescribir el sistema completo.

Una de las ventajas más importantes de este proyecto se encuentra en la adopción de herramientas basadas en software libre. Para el desarrollo de la aplicación y la comunicación con los distintos módulos, se ha utilizado el lenguaje de programación Python, haciendo uso de librerías proporcionadas por los propios fabricantes de los dispositivos. Además, la programación de los microcontroladores empleados (PIC32MX [3] y PIC18F [4]) se ha llevado a cabo mediante el entorno de desarrollo MPLAB X [5], junto con compiladores de código abierto como GCC y bibliotecas oficiales.

Este enfoque no solo contribuye a la reducción de costes, sino que también fomenta la transparencia, la adaptabilidad del sistema y la posibilidad de colaboración, tanto en contextos académicos como en entornos profesionales.

Desde el punto de vista de la interacción hardware-software, el sistema permite integrar distintos **protocolos de comunicación**, como Ethernet, Flow-Bus o CAN, gestionando de forma automatizada la recopilación de datos y el tratado de estos. También hemos trabajado en garantizar que no existan fugas de gases, mediante la detección de posibles anomalías en la presión o fugas en el flujo. Esto es crucial debido a que algunas de las combinaciones gaseosas pueden resultar tóxicas en concentraciones elevadas.

El paso de tecnologías analógicas a dispositivos digitales, como los Mass Flow Controllers (MFCs), ha permitido mejorar la precisión y fiabilidad de las mediciones, así como automatizar procesos que antes requerían intervención manual. Todo esto se ha logrado sin renunciar a la flexibilidad, gracias al diseño modular y al uso de drivers y herramientas de código abierto.

Tal como indican Hernández Gómez y Tuma, “**los datos experimentales son esenciales para validar y refinar modelos predictivos del comportamiento de los gases, especialmente en condiciones no ideales**” [6]. Esta afirmación argumenta la necesidad de completar los estudios teóricos con observaciones empíricas controladas. En esta misma línea, organismos como la Agencia de Protección Ambiental de los Estados Unidos (EPA), destacan que el desarrollo de metodologías experimentales para medir gases en condiciones controladas es clave para evaluar el impacto ambiental, modelar emisiones y diseñar políticas eficaces de mitigación [7].

En este contexto, el laboratorio de detección de gases de la Facultad de Física de la Universidad de Barcelona está llevando a cabo el desarrollo de sensores capaces de detectar la presencia de determinados gases. Uno de los aspectos más significativos, es que es capaz de discriminar entre compuestos con propiedades similares, como el monóxido de carbono (CO) y el dióxido de carbono (CO_2). Para poder estudiar el comportamiento de estos sensores, se ha desarrollado un sistema experimental que permite inyectar distintas combinaciones de gases y analizar la respuesta del sensor bajo condiciones controladas.

El principio de funcionamiento de este sistema experimental se basa en la ley de Ohm. Lo que hacemos es aplicar una tensión conocida al sensor y se mide la corriente que circula a través del mismo, con la presencia o no de la combinación de gases. La evolución de dicha corriente, y, por tanto, de la impedancia del sensor, nos permite determinar la base para la discriminación selectiva entre gases con características similares.

La capacidad de observar, modelar y predecir con precisión el comportamiento de sensores de gases ofrece aplicaciones de gran relevancia en los ámbitos científico, industrial y medioambiental. En primer lugar, permite optimizar la eficiencia energética de procesos térmicos como la combustión y la refrigeración, mediante un ajuste más preciso de las condiciones previas. Un estudio reciente de *The Times* ha identificado que los sistemas de calefacción a gas en Londres son responsables del 72 % de las emisiones de óxidos de nitrógeno (NOx) en el centro de la ciudad, lo que destaca la importancia de comprender y controlar las emisiones en estos procesos para mejorar la calidad del aire urbano.

En definitiva, este proyecto no solo contribuye a la comprensión experimental del comportamiento de los sensores de gases, sino que también demuestra cómo un enfoque de ingeniería del software, bien estructurado y orientado a la reutilización y la escalabilidad, puede facilitar el trabajo de investigadores. Esta estación pretende ser una plataforma de referencia para el estudio de combinaciones gaseosas, el modelado de emisiones y la validación de sensores.

1.3 Definición de objetivos

El **objetivo principal** de este proyecto es desarrollar un software de control para una estación experimental de sensores de gases recientemente instalada en el Departamento de Ingeniería Electrónica y Biomédica de la Universidad de Barcelona. Esta estación ha sido diseñada para estudiar y caracterizar el comportamiento de materiales empleados como sensores de gases bajo condiciones ambientales controladas.

El software desarrollado permitirá gestionar hasta siete líneas de gases diferentes (como O_2, CO, CO_2, entre otros), aplicar estímulos eléctricos en sensores de gases y monitorizar en tiempo real la respuesta del mismo. Gracias a esta herramienta, será posible obtener datos experimentales precisos y reproducibles, esenciales para la validación de modelos predictivos y el desarrollo de nuevas aplicaciones tanto en investigación.

Para alcanzar el objetivo general de este proyecto, se han planteado los siguientes **objetivos específicos**:

1. **Implementar los protocolos de comunicación** necesarios para garantizar una interacción robusta y eficiente entre el software y el hardware de la estación de gases. Estos protocolos incluyen:
    - Ethernet
    - Flow-Bus
    - Puerto Serie
    - CAN

    Este conjunto de protocolos serán integrados en un sistema unificado, transparente para el usuario, que facilite la gestión centralizada de todo el conjunto de dispositivos.
2. **Desarrollar el software de control**, que permita configurar, ejecutar y monitorizar los experimentos desde una única aplicación. Este software deberá cubrir funcionalidades clave como:
    - Aplicación de estímulos eléctricos sobre los sensores.
    - Selección y combinación de hasta seis líneas de gases distintas.
    - Adquisición, procesamiento y almacenamiento automatizado de datos experimentales.
3. **Diseñar una interfaz gráfica de usuario (GUI)** modular, clara e intuitiva, basada en el patrón de diseño Model-View-Controller, que permita:
    - Configurar todos los parámetros del experimento de forma precisa.
    - Visualizar en tiempo real los resultados obtenidos mediante gráficos dinámicos.
    - Exportar automáticamente los datos registrados en archivos Excel.
    - Mejorar la trazabilidad y reducir errores operativos durante la ejecución de medidas.

1.4 Planificación y metodología de trabajo

1.4.1 Metodología de trabajo

Para alcanzar los objetivos definidos en este proyecto, se ha adoptado una **metodología de desarrollo incremental**, orientada a la práctica y a la validación continua de cada componente del sistema. Este enfoque ha permitido integrar progresivamente las distintas funcionalidades, asegurando la estabilidad del sistema y facilitando la detección temprana de errores.

El punto de partida fue el **análisis de una solución anterior** desarrollada por National Instruments [9], basada en el entorno gráfico LabVIEW [10]. Aunque esta plataforma ofrecía una integración ágil con los dispositivos, presentaba importantes limitaciones en términos de escalabilidad, flexibilidad del código y adaptación a futuras ampliaciones.

Ante estas restricciones, se decidió desarrollar una solución propia en Python, un lenguaje que permite una programación más modular, flexible y mantenible. Esta elección facilitó la integración de múltiples protocolos de comunicación, el control directo de hardware diverso y la creación de una interfaz gráfica personalizable.

1.4.2 Visión general de la estación de gases

La estación experimental tiene como finalidad analizar cómo varían las propiedades eléctricas de distintos sensores al ser expuestos a diferentes gases. Este análisis es crucial, ya que un buen sensor no solo debe detectar un gas concreto, sino también ser **selectivo**, es decir, reaccionar de manera diferenciada según el gas presente.

Los sensores bajo estudio son **estimulados eléctricamente,** aplicando una tensión o corriente, y midiendo la respuesta correspondiente (corriente o tensión, respectivamente). Las variaciones de estas medidas en función del gas aplicado permiten evaluar la sensibilidad y selectividad del material.

La **Figura 1** ilustra la arquitectura general de la estación de gases, incluyendo las líneas de suministro de gases, los sensores, y los dispositivos de control y adquisición.

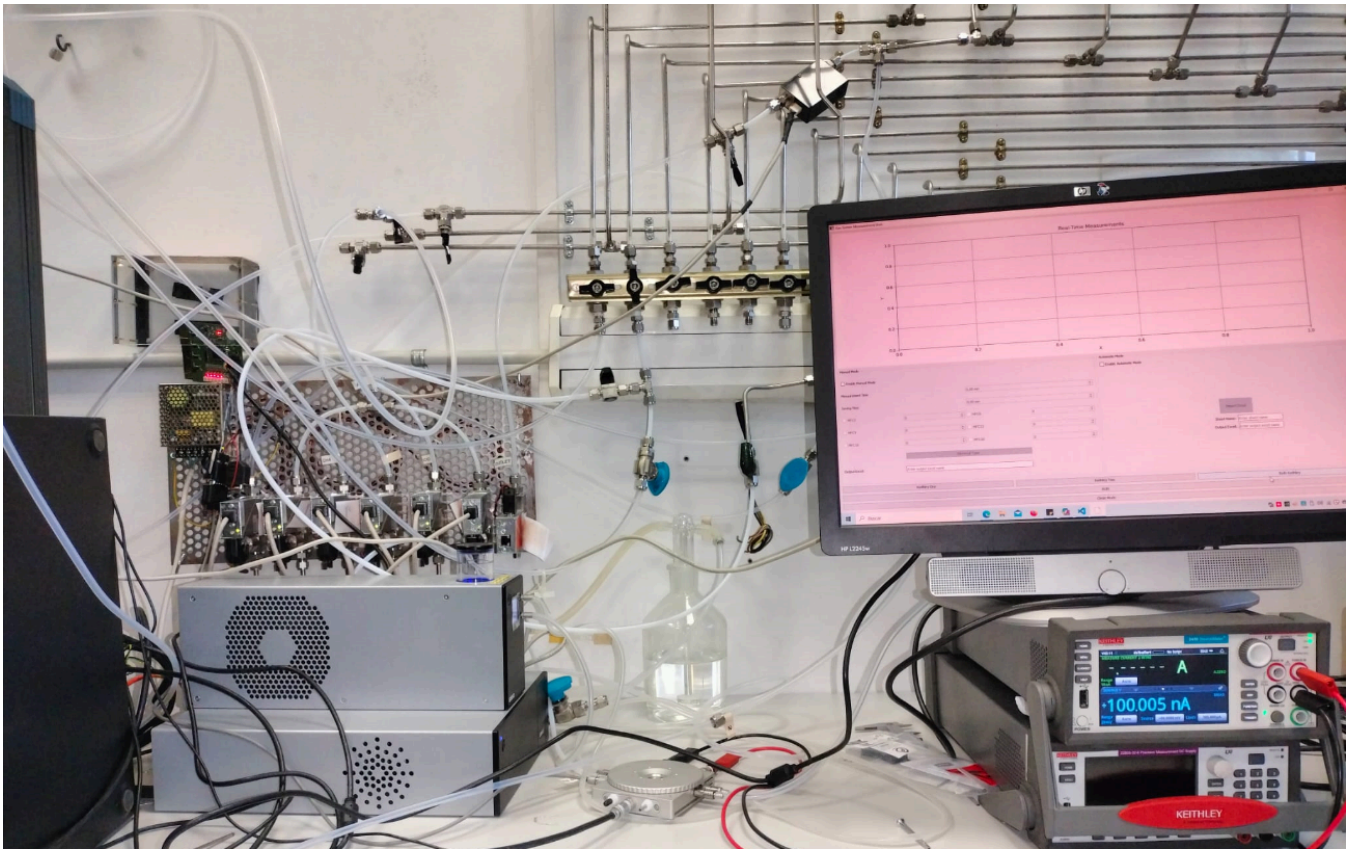

Figura 1: Arquitectura general de la nueva estación de gases.

1.4.3 Etapas de desarrollo

El desarrollo del sistema se estructuró en las siguientes fases:

1. **Desarrollo inicial con una única Keithley.** Se inició con la implementación del control básico de un multímetro Keithley, validando la comunicación por Ethernet, la adquisición de medidas y la estructura general del software.

2. **Ampliación a doble Keithley.** Una vez verificado el funcionamiento con un único dispositivo, se adaptó el sistema para gestionar dos unidades simultáneamente. Esta etapa requirió una reorganización del código y la incorporación de procesamiento en paralelo (multithreading) para evitar condiciones de carrera y asegurar una adquisición de datos fiable.

3. **Diseño de la interfaz gráfica (GUI).** Superada la fase de pruebas en terminal, se desarrolló una interfaz gráfica intuitiva y modular, basada en el patrón de diseño Model-View-Controller (MVC). Esta interfaz permite al usuario configurar los experimentos, visualizar en tiempo real los datos mediante gráficos dinámicos y exportar automáticamente los resultados a archivos Excel.

4. **Documentación y redacción de la memoria.** Una vez finalizado el desarrollo técnico, se llevó a cabo la documentación detallada del proyecto, estructurando la memoria sobre los fundamentos teóricos, la arquitectura del sistema y las decisiones técnicas adoptadas.

1.4.3 Planificación de tareas

Para alcanzar los objetivos mencionados, es fundamental desglosar y planificar las tareas a desarrollar.

- **Análisis y definición inicial del sistema:**
    1. Estudio del sistema previo basado en LabVIEW.
    2. Identificación de las limitaciones del enfoque anterior.
    3. Investigación de dispositivos, protocolos de comunicación y requisitos experimentales
- **Desarrollo e implementación:**
    1. Integración de los MFCs, la cámara sensorizada y la placa de control.
    2. Control y comunicación con una Keithley.
    3. Exportación y gestión de los datos de medición.
    4. Adaptación de los sistemas para el uso de dos Keithley en paralelo.
    5. Diseño e implementación de la GUI.

- **Documentación:**

    1. Estructuración y redacción de la memoria del Trabajo de Fin de Grado.
    2. Revisión del documento, corrección de errores y adaptación según la retroalimentación.

1.4.4 Diagrama de Gantt

El diagrama de Gantt ofrece una representación visual de la duración y secuencia de las tareas de un proyecto, distribuidas a lo largo de un período de tiempo. Se trata de una herramienta útil para planificar, organizar y hacer seguimiento del progreso del proyecto, ayudando a garantizar que se cumplan los plazos establecidos.

En este caso, se ha elaborado un diagrama de Gantt **(Figura 2)**, que refleja la duración real de cada una de las tareas realizadas a lo largo del desarrollo del proyecto. Aunque no se definió un diagrama inicial de planificación detallado, este cronograma permite visualizar cómo se distribuyeron las tareas en el tiempo, ofreciendo una visión clara del flujo de trabajo seguido hasta la finalización del proyecto.

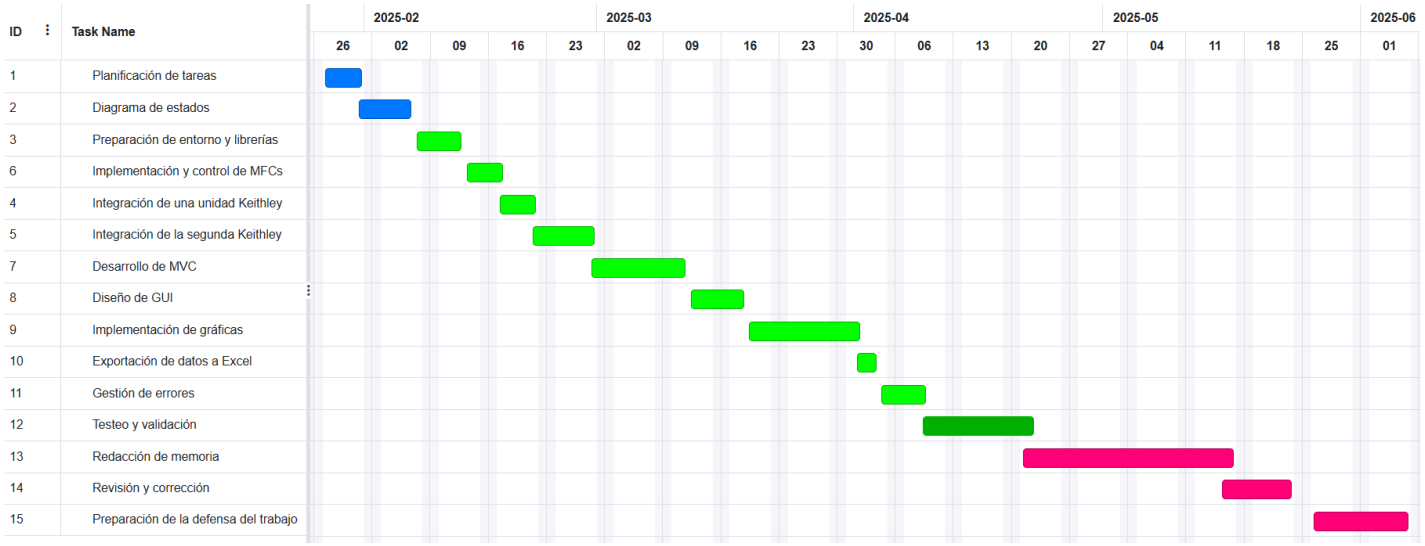

Figura 2: Diagrama de Gantt.

Capítulo 2: Antecedentes

Antes del desarrollo del sistema de medida y control de gases, las instalaciones de la Universidad de Barcelona (UB) empleaban una solución basada en el entorno de programación gráfica LabVIEW. Esta herramienta ofrecía una interfaz visual intuitiva para configurar y visualizar señales, pero presentaba importantes limitaciones estructurales y de mantenimiento, especialmente al escalar o modificar funcionalidades específicas del sistema.

Figura 3: Arquitectura general de la antigua estación de gases.

Una de las principales dificultades del sistema anterior era la falta de modularidad. En LabVIEW, gran parte de la lógica de control estaba incrustada en bloques gráficos interdependientes, lo cual dificulta la reutilización de componentes y el aislamiento de funcionalidades. Esta situación implicaba que fuera una arquitectura poco flexible, ya que cualquier modificación en un módulo específico requería revisar y rehacer conexiones con otras partes del sistema. Además, la duplicación de lógica o el acoplamiento eran errores comunes, aumentando el riesgo de errores y complicando las tareas de depuración.

Por otra banda, el sistema desarrollado en este proyecto implementa el patrón arquitectónico Model-View-Controller (MVC), utilizado por sus ventajas en cuanto a organización, mantenimiento y escalabilidad del código. Esta estructura permite dividir el sistema en tres componentes independientes: el modelo (donde se encuentra la lógica de negocio y el acceso a los datos), la vista (encargada de la interfaz de usuario) y el controlador (que gestiona la comunicación entre el modelo y la vista). Gracias a esta separación de responsabilidades, es posible trabajar en cada componente de manera aislada, facilitando la incorporación de nuevas funcionalidades o el reemplazo de partes del sistema sin necesidad de modificar el resto del código.

Además del rediseño arquitectónico y la mejora en la estructura del software, el nuevo sistema de medida incorpora una plataforma electrónica adicional desarrollada específicamente para la monitorización de temperatura y humedad en tiempo real dentro de la cámara del sensor. Esta tarjeta electrónica está equipada con un microcontrolador PIC18F, responsable de adquirir las medidas ambientales previamente mencionadas mediante un bus I²C [11], y transmitirlas hacia el exterior utilizando un protocolo robusto y fiable como el bus CAN. Este sistema de comunicación se ha implementado siguiendo el bus altamente fiable CAN (Controller Area Network), utilizado en entornos de automoción e instrumentación por su capacidad para gestionar múltiples nodos y garantizar la integridad de los datos incluso en condiciones exigentes.

En este caso, la comunicación se establece entre la tarjeta interna y una tarjeta master ubicada fuera de la cámara, equipada con un microcontrolador PIC32MX795L, también programado en lenguaje C mediante el entorno MPLAB X del fabricante. Esta arquitectura no solo permite una monitorización precisa del entorno de medida, sino que ofrece una visión de futuro escalable, en la que es posible integrar múltiples cámaras con sensores independientes.

La mayor parte del proyecto ha sido desarrollado en Python, un lenguaje de programación de propósito general. Esta elección no solo permite aprovechar una gran variedad de librerías especializadas y herramientas de código abierto, sino que también facilita el desarrollo colaborativo, la portabilidad del sistema y su integración con otras plataformas. En cambio, el uso de LabVIEW implicaba una dependencia de licencias comerciales y una menor interoperabilidad con otras tecnologías de software libre.

Otra mejora significativa respecto al sistema anterior es la integración de programación orientada a objetos (POO). Esta metodología permite representar los distintos elementos del sistema (como sensores o controladores) como clases con atributos y métodos propios, ayudando a mantener la encapsulación, reutilización y extensibilidad del código. En el sistema anterior, basado en bloques funcionales sin una estructura orientada a objetos, era más complicado gestionar múltiples dispositivos o protocolos sin generar código redundante y difícil de mantener.

Un claro ejemplo de estas mejoras es la gestión de los Mass Flow Controllers (MFCs). En la versión anterior la conexión a un solo MFC ya resultaba compleja y poco flexible, pero el nuevo sistema permite conectar y controlar hasta siete MFCs simultáneamente. El diseño modular y orientado facilita su configuración, control y monitorización. Si se desea integrar un nuevo MFC en otro proyecto, basta con importarlo en el archivo correspondiente, sin necesidad de realizar modificaciones profundas en el resto del sistema.

El uso de software libre permite eliminar barreras económicas para el uso, modificación y distribución del sistema. Esto no solo reduce los costes, sino que también promueve la transparencia y la auditoría del código y potencia la innovación abierta.

En resumen, el desarrollo de esta nueva estación de medida supone una mejora significativa en términos de diseño, eficiencia y mantenimiento del software. La transición desde un sistema cerrado hacia una solución estructurada en MVC proporciona una base sólida y escalable para futuras investigaciones.

Capítulo 3: Ingeniería de la concepción

3.1 Componentes del sistema experimental

En esta sección describiremos en detalle los distintos componentes físicos que componen el sistema experimental. Cada uno de estos elementos desempeñan un papel clave en la adquisición de datos, el control del entorno o la gestión de señales. La correcta integración y funcionamiento de estos dispositivos es esencial para poder garantizar la fiabilidad de los resultados obtenidos y la reproducibilidad de los experimentos.

A continuación, analizaremos las funcionalidades de los instrumentos utilizados y su importancia dentro del sistema. Además, incluiremos consideraciones sobre sus conexiones físicas y protocolos de comunicación asociados.

3.1.1 Keithley 2450

La **Keithley 2450 SourceMete**r (SMU) [12] **(Figura 4)** es un instrumento versátil que combina en un solo equipo la capacidad de generar señales eléctricas de tensión o corriente y, al mismo tiempo, medir con alta precisión la respuesta eléctrica del dispositivo bajo prueba. Esta integración la convierte en una herramienta fundamental para la caracterización eléctrica de sensores, materiales funcionales y dispositivos electrónicos, especialmente en contextos de desarrollo experimental.

En el sistema experimental desarrollado, la Keithley 2450 se utiliza para **a**plicar una señal eléctrica controlada, ya sea voltaje o corriente, a un sensor ubicado en una cámara de ensayo, y registrar la respuesta eléctrica en tiempo real. Esta información es clave para analizar cómo varía el comportamiento eléctrico del sensor en función del tipo de gas presente en la atmósfera de la cámara, lo que permite estudiar tanto la sensibilidad como la selectividad del material sensor.

Al inicio, el desarrollo del software se centró en la comunicación con una única unidad Keithley, lo que permitió validar la arquitectura base del sistema de medida y comprobar la correcta implementación de los comandos de control. Una vez el control de una Keithley funcionó correctamente, se amplió la funcionalidad para permitir el uso de dos Keithley de forma simultánea, lo que implicó una reestructuración del código para incorporar una gestión concurrente robusta mediante programación paralela (threading). El uso de hilos independientes garantiza que ambas unidades operen sin interferencias ni bloqueos, permitiendo una adquisición de datos fluida y paralela.

La comunicación entre el PC y las unidades Keithley se realiza mediante el protocolo **Ethernet** (TCP/IP) [13], lo que proporciona una conexión estable, rápida y escalable, ideal para entornos de laboratorio. Esta interfaz no solo facilita la transferencia de comandos y datos, sino que también permite el control remoto de los instrumentos desde diferentes entornos de desarrollo. La instrumentación se controla mediante el estándar **SCPI** (Standard Commands for Programmable Instruments) [14], ampliamente adoptado en el ámbito de la instrumentación electrónica avanzada.

Figura 4: Aspecto físico de una unidad Keithley 2450.

3.1.2 Mass Flow Controllers (MFCs)

Los **Mass Flow Controllers** [15] **(Figura 5)** son dispositivos clave en sistemas experimentales donde se requiere una regulación precisa del flujo de gases. Su función principal es controlar de forma activa y estable el flujo de gas, ajustándose al valor de consigna establecido, independientemente de posibles variaciones en la presión o temperatura que puedan presentarse en el sistema.

En nuestro sistema experimental se utilizan MFCs de la marca **Bronkhorst** [16], ampliamente reconocidos en entornos científicos e industriales por su precisión, fiabilidad y capacidad de integración con sistemas automatizados. En concreto, el sistema cuenta con seis MFCs calibrados individualmente para diferentes tipos de gas y rangos específicos de flujo. Esta configuración permite la generación precisa de mezclas gaseosas controladas. Además, se ha incorporado un séptimo MFC que actúa como limitador de flujo total, siendo configurado para un flujo máximo de 200 sccm. Esta medida garantiza un funcionamiento seguro del sistema, limitando el flujo global en cualquier configuración de mezcla.

Durante el desarrollo de los experimentos, el flujo de los MFCs puede ajustarse dinámicamente desde el software, lo que permite modificar en tiempo real la composición gaseosa de la cámara a lo largo de diferentes medidas. Esta flexibilidad es fundamental para simular distintas condiciones ambientales y analizar cómo responde el sensor frente a variaciones controladas en la concentración y composición del gas.

La comunicación entre el PC y los MFCs se establece mediante el protocolo **Flow-Bus** [17], implementado en Python. Esta solución permite el control digital individualizado de cada unidad desde el software principal del sistema. Flow-Bus es un protocolo de comunicación industrial diseñado específicamente para dispositivos de control de flujo, que ofrece una alta fiabilidad en entornos experimentales complejos. Esta interfaz facilita además la lectura del estado de cada MFC, permitiendo detectar errores como sobrepresiones, fallos de comunicación o flujos fuera de rango. Estas comprobaciones automáticas contribuyen a una operación segura y estable del sistema.

La integración de los MFCs en el sistema automatizado de medida no solo mejora la seguridad y la precisión del experimento, sino que también permite definir, programar y reproducir múltiples escenarios experimentales de forma eficiente y consistente.

Figura 5: Aspecto físico de dos Mass Flow Controller de la marca Bronkhorst.

3.1.3 Cámara sensorizada Linkam THMS600

La **Linkam THMS600** [18] **(Figura 6)** es un sistema de control térmico de alta precisión diseñado para aplicaciones científicas que necesitan una regulación estable y reproducible de la temperatura. Es muy útil en el estudio de materiales sensibles a variaciones térmicas, donde las propiedades físicas pueden depender significativamente de la temperatura. En nuestro sistema experimental, la THMS600 se encarga de controlar con precisión la temperatura de la cámara de medida, donde se aloja el sensor bajo estudio. Esta capacidad de regulación térmica es esencial para evaluar la respuesta del sensor bajo diferentes condiciones ambientales y garantizar la repetibilidad de las mediciones.

El control de la Linkam se realiza mediante el software oficial del fabricante. Sin embargo, para complementar esta regulación y obtener una monitorización ambiental en tiempo real, se ha desarrollado una solución electrónica específica en el Departamento de Ingeniería Electrónica. Esta solución se compone de una tarjeta secundaria ubicada en el interior de la cámara, que integra un sensor SHT11 de Sensirion© [19], encargado de medir temperatura y humedad relativa. El sensor transmite los valores medidos al microcontrolador PIC18F27Q83 mediante el protocolo digital I²C, y este, a su vez, los envía al exterior utilizando **High-Speed CAN** [20] (Controller Area Network), un protocolo robusto y eficiente ampliamente utilizado en entornos industriales y de automoción.

En el exterior, los datos se reciben en una placa principal basada en el microcontrolador PIC32MX795L, que actúa como nodo master del bus CAN. Esta placa centraliza la adquisición de datos y transmite la información al ordenador mediante conexión USB. De esta forma, los valores ambientales se integran en el sistema de control y adquisición, sincronizándose con otras variables experimentales como el flujo de gases o la excitación eléctrica aplicada a través de la fuente Keithley.

La elección del protocolo CAN permite una transmisión fiable y resistente a interferencias electromagnéticas, además de una estructura escalable y modular que facilita la futura conexión de múltiples cámaras al mismo sistema.

Figura 6: Aspecto físico de la cámara THMS600.

3.1.4 Placa de control

La **placa de control (Figura 7)** actúa como el cerebro del sistema experimental, coordinando las interacciones entre los distintos dispositivos y colaborando en la automatización de las mediciones. Su función principal es gestionar la comunicación entre instrumentos, enviando comandos de control y recibiendo datos en tiempo real, lo que permite una ejecución sincronizada y precisa del experimento.

La placa se comunica con la Linkam THMS600 mediante el protocolo **High-Speed CAN**, una interfaz robusta y de alta velocidad que permite transmitir órdenes térmicas de forma eficiente y controlar con precisión la temperatura aplicada al sensor durante el experimento.

Además, la placa de control se conecta con el PC a través de **COM Serial**, que sirve como canal principal para la recepción de instrucciones desde el software de gestión y para la transmisión de datos adquiridos durante las mediciones. Esta conexión garantiza una comunicación estable y directa con el sistema informático, desde donde se controla todo el experimento.

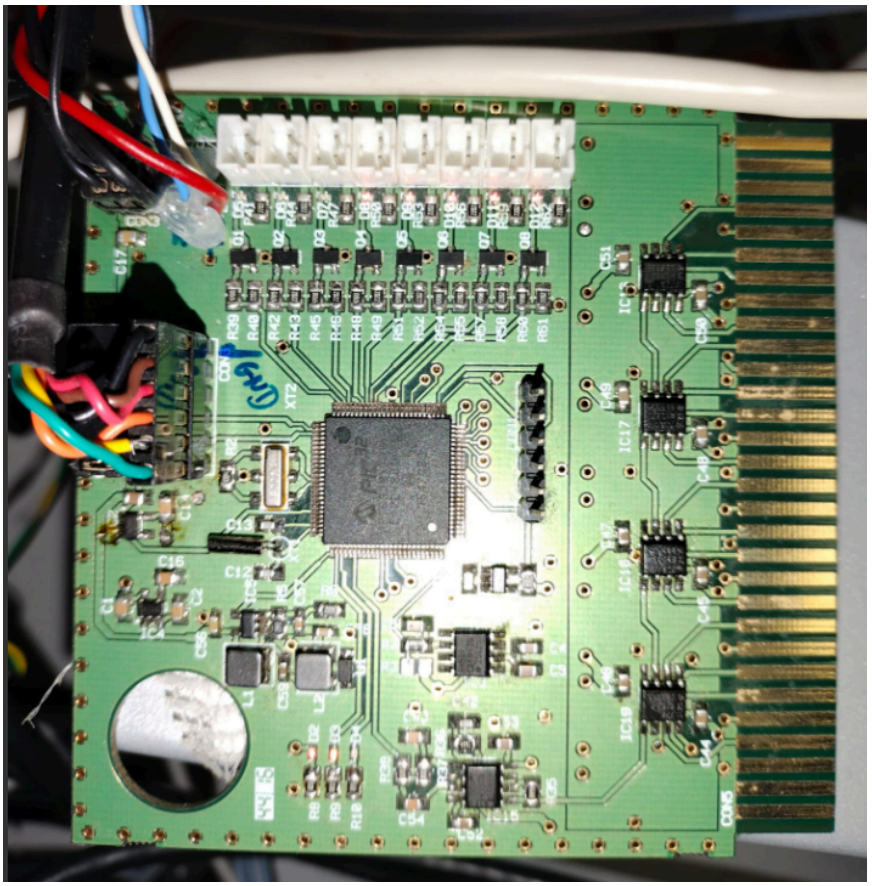

Figura 7: Aspecto físico de la placa de control.

3.2 Protocolos de comunicación

En esta sección se describen los protocolos de comunicación empleados en el sistema experimental. Estos protocolos son fundamentales para garantizar la correcta interconexión de los diversos dispositivos, permitiendo una comunicación eficiente y estable entre ellos. Cada protocolo ha sido seleccionado en función de las necesidades de velocidad, fiabilidad y compatibilidad de los distintos elementos del sistema.

A continuación, analizaremos las características de cada protocolo, su aplicación en el sistema y las ventajas que aportan en términos de control, adquisición de datos y sincronización de las mediciones. Además, se abordarán las conexiones físicas asociadas

a cada protocolo, así como su implementación en el contexto de los instrumentos y dispositivos utilizados. Para facilitar la comprensión global del sistema, se incluye un esquema general **(Figura 8)** que representa gráficamente cómo se integran y comunican los dispositivos mediante estos protocolos.

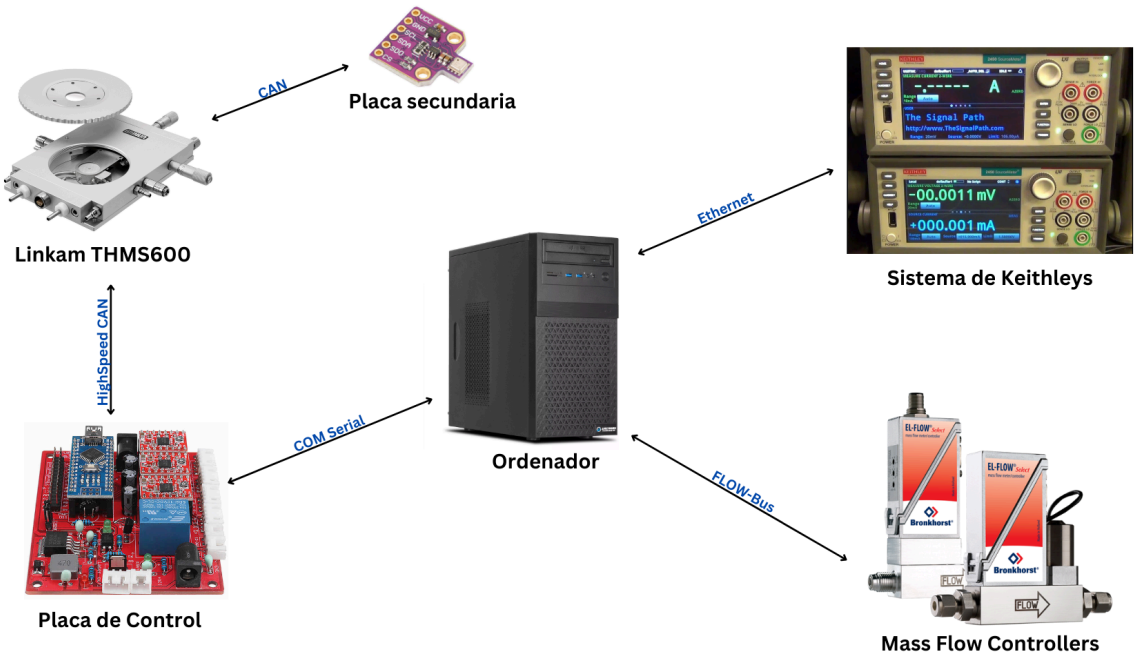

Figura 8: Esquema general del sistema experimental.

3.2.1 Ethernet - TCP/IP con SCPI

El protocolo Ethernet con TCP/IP es ampliamente utilizado en entornos de instrumentación debido a su capacidad para establecer conexiones de red fiables y de alta velocidad. En nuestro sistema experimental, se emplea para la comunicación entre el PC y las unidades Keithley 2450 SourceMeter. Estas unidades utilizan el estándar SCPI (Standard Commands for Programmable Instruments) para recibir comandos y enviar datos, lo que permite un control preciso y programable de los parámetros eléctricos durante los experimentos.

Algunas de las ventajas que este protocolo nos ofrece son:

- **Alta velocidad de transmisión:** El uso de Ethernet permite una transferencia rápida de datos, esencial para aplicaciones que requieren una adquisición de datos en tiempo real.

- **Fiabilidad y estabilidad:** TCP/IP proporciona una comunicación robusta y estable, minimizando la pérdida de datos y garantizando la integridad de la información transmitida.

- **Flexibilidad y escalabilidad:** La infraestructura del protocolo facilita la integración de múltiples dispositivos y la expansión del sistema sin necesidad de cambios significativos en el hardware existente.

- **Compatibilidad con estándares abiertos:** El uso de SCPI permite una programación estandarizada y facilita la interoperabilidad entre diferentes dispositivos y plataformas de software.

La conexión entre el PC y las unidades Keithley se realiza mediante cables Ethernet estándar, utilizando el protocolo TCP/IP para la transmisión de datos. El software de control envía comandos SCPI a las unidades para configurar parámetros como la tensión o corriente aplicada, y recibe las mediciones correspondientes en tiempo real.

3.2.2 FLOW-Bus

FLOW-BUS es un protocolo de comunicación desarrollado por Bronkhorst, basado en la tecnología RS485, diseñado específicamente para la interconexión de dispositivos digitales como los MFCS. En nuestro sistema, se utiliza para la comunicación entre el PC y los MFCs de Bronkhorst, permitiendo un control preciso y coordinado del flujo de gases durante los experimentos.

Algunas de las ventajas que este protocolo nos ofrece son:

- **Comunicación digital robusta:** Este protocolo ofrece una transmisión de datos fiable y resistente a interferencias, imprescindible para mantener la precisión en el control de flujos de gases.

- **Topología en red:** Permite la conexión de múltiples dispositivos en una misma red, facilitando la sincronización y el control centralizado de varios MFCs.

- **Integración con software de control:** Es compatible con entornos de programación como Python, lo que facilita su integración en sistemas automatizados.

El PC se comunica con los MFCs a través de un convertidor USB a RS485, utilizando el protocolo FLOW-BUS para enviar comandos de control y recibir datos de flujo. Esto permite ajustar dinámicamente las mezclas de gases y monitorizar en tiempo real las condiciones del experimento.

3.2.3 COM - Serial

La **comunicación serie** es una técnica ampliamente utilizada para la transmisión de datos entre dispositivos electrónicos. En nuestro sistema experimental, se emplea para enlazar el PC con la placa de control, permitiendo el envío de comandos y la recepción de datos relacionados con el estado del sistema y las mediciones adquiridas.

Esta comunicación es **asíncrona**, basada en el protocolo **RS232** [21], aunque encapsulada en la capa física USB para facilitar su integración con equipos modernos. Para ello, se utiliza un adaptador USB a UART de tipo TTL, concretamente el modelo **FTDI TTL-232R-5V** [22], el cual actúa como puente entre el puerto USB del ordenador y la UART del microcontrolador. Este cable incorpora un chip FTDI que proporciona una conversión entre ambos estándares y facilita la detección automática del dispositivo en la mayoría de sistemas operativos. El conector permite una conexión directa y segura con la placa controladora.

Algunas de las ventajas que este protocolo nos ofrece son:

- **Simplicidad y bajo costo:** La implementación de la comunicación serie sencilla y económica, requiriendo un mínimo de hardware adicional.

- **Fiabilidad en la transmisión de datos:** Ofrece una comunicación estable y con bajo riesgo de interferencias, adecuada para la transmisión de datos a velocidades moderadas.

- **Amplia compatibilidad:** Es compatible con una gran variedad de dispositivos y sistemas operativos, facilitando su integración en diferentes entornos.

El PC se conecta a la placa de control mediante un cable serial, utilizando este canal para enviar instrucciones de control y recibir información sobre el estado de los dispositivos conectados. Esta comunicación es esencial para la coordinación de las operaciones del sistema experimental.

3.2.4 CAN Bus - HighSpeed CAN

El Controller Area Network es un protocolo de comunicación robusto que permite la comunicación entre múltiples dispositivos sin la necesidad de una computadora central. En nuestro sistema experimental, se utiliza **High Speed CAN** para la comunicación eficiente y sincronizada entre sensores industriales de presión y temperatura conectados al sistema a través de un bus común.

Algunas de las ventajas que este protocolo nos ofrece son:

- **Alta fiabilidad y tolerancia a fallos:** Este protocolo detecta automáticamente errores de transmisión, aisla nodos defectuosos y garantiza la integridad de los datos gracias a sus mecanismos internos de verificación como CRC y ACK.

- **Velocidad y eficiencia:** High Speed CAN permite la transmisión rápida de datos con baja latencia, adecuada para sistemas en los que es fundamental la sincronización precisa de señales de sensores.

- **Topología simple y escalabilidad:** Permite conectar múltiples nodos (hasta 112 dispositivos) con solo dos cables, lo que reduce el cableado y facilita la expansión del sistema.

- **Priorización de mensajes:** El protocolo implementa un sistema de prioridad basado en el identificador del mensaje, lo que garantiza que los datos críticos se transmitan con prioridad.

Los sensores de presión y temperatura industriales están conectados al **bus CAN** mediante conectores estándar M12. Los datos recopilados se transmiten en tiempo real al PC a través de un **convertidor USB-CAN**, que traduce los paquetes CAN a una interfaz compatible con el software de adquisición. Esta arquitectura permite capturar mediciones de múltiples sensores de forma simultánea, garantizando su sincronización temporal.

Capítulo 4: Ingeniería del detalle

4.1 Introducción

El desarrollo del software compone una parte esencial del proyecto, ya que permite la gestión integral de la estación experimental: desde el control de los distintos dispositivos físicos hasta la adquisición, tratamiento y visualización de los datos generados. A lo largo de este capítulo se explica con detalle la arquitectura del sistema software implementado, justificando las decisiones de diseño adoptadas y analizando los distintos bloques funcionales que lo componen.

Con el objetivo de garantizar la modularidad, la escalabilidad y la facilidad de mantenimiento del código, se ha optado por una estructura basada en el patrón de diseño **Modelo-Vista-Controlador** (MVC) [23]. Este enfoque favorece una clara separación entre la lógica de negocio, la interfaz gráfica y la gestión de eventos, permitiendo una evolución ordenada del proyecto y una integración eficiente de nuevos módulos.

En las siguientes secciones se describe la organización general del código, incluyendo un diagrama de clases que ilustra la distribución de responsabilidades entre los distintos componentes. Además, se detallarán los módulos que conforman cada una de las capas del patrón MVC, explicando sus funciones específicas y su interacción general. Finalmente, se detallarán aspectos técnicos relevantes como la gestión de la concurrencia mediante **threading**, los mecanismos de comunicación con los dispositivos, y las estrategias adoptadas para asegurar la robustez del sistema.

4.2 Arquitectura del software

4.2.1 Elección del patrón arquitectónico

Con el objetivo de garantizar una estructura de software robusta y escalable, se ha adoptado por utilizar el patrón arquitectónico Modelo-Vista-Controlador (MVC) como base del desarrollo. Este patrón permite una separación clara de responsabilidades dentro de una aplicación, facilitando su comprensión, desarrollo paralelo y futura extensión.

El patrón MVC se compone de tres componentes fundamentales:

- **Modelo:** Este componente se encarga de encapsular la lógica del negocio, los datos el sistema y las operaciones que los manipulan.
- **Vista:** Este componente se encarga de representar gráficamente los datos obtenidos de las medidas, además de gestionar el comportamiento de la interfaz visual.
- **Controlador:** Este componente actúa como intermediario entre el modelo y la vista, y se encarga de gestionar las interacciones del usuario y de actualizar los componentes según sean necesarios.

Este enfoque es especialmente beneficioso en entornos experimentales complejos como el desarrollado en este proyecto, donde coexisten múltiples dispositivos, protocolos y operaciones asincrónicas. Mediante MVC, se minimiza el **acoplamiento** entre componentes y se mejora la **modularidad**, lo que permite modificar o sustituir partes del sistema sin afectar al conjunto [24].

Durante el desarrollo del sistema, se identificó la necesidad de desacoplar aún más la lógica asociada a la gestión de hardware. Por este motivo, se decidió extender el patrón MVC con una capa adicional denominada **P** (Packets). Esta capa encapsula de forma individual la interacción con cada uno de los dispositivos físicos o módulos funcionales del sistema.

Cada paquete actúa como una abstracción de un subsistema, y se encarga de gestionar su conexión, configuración y operación.

El propósito fundamental de esta capa es desacoplar completamente la lógica específica del hardware del resto del sistema, estableciendo una interfaz clara y modular entre los dispositivos físicos y las capas superiores del software. Esta separación proporciona múltiples ventajas clave:

- **Reutilización del código**: Cada *packet* se diseña como un módulo independiente y autocontenible, lo que permite su uso en otros proyectos sin necesidad de conocer o adaptar la lógica de control del sistema actual.

- **Facilidad de pruebas**: Al aislar la lógica de comunicación con los dispositivos, cada módulo puede ser testeado de forma individual, impulsando una arquitectura robusta y orientada a pruebas (test-driven).

- **Escalabilidad y mantenimiento simplificados**: En caso de que sea necesario cambiar el modelo de un instrumento (por ejemplo, un nuevo modelo de Keithley) o incorporar un nuevo tipo de controlador de flujo (MFC), bastará con modificar o añadir un nuevo *packet*, sin alterar el resto de la estructura del software.

- **Extensibilidad del sistema**: Esta arquitectura facilita la integración de nuevas funcionalidades o el soporte de dispositivos adicionales sin afectar al núcleo de la aplicación, preservando su estabilidad y favoreciendo el crecimiento modular del sistema.

En esta estructura extendida, el Modelo se encarga exclusivamente de la lógica de operación y los algoritmos genéricos (por ejemplo, cómo interpretar los datos de medida), mientras que la capa P (Packets) se responsabiliza del acceso al hardware físico y sus particularidades.

Este enfoque sigue los principios del Diseño Orientado a Objetos, tales como **encapsulamiento** y **responsabilidad única**, alineándose también con buenas prácticas de desarrollo como el principio **Open/Closed** del paradigma SOLID [25]. El resultado es una arquitectura flexible, comprensible y preparada para modificaciones futuras.

La estructura final puede visualizarse como una evolución del patrón MVC clásico, con la incorporación explícita de una capa de comunicación y control de hardware (P), dando lugar a una arquitectura **MVC-P (Figura 9)** .

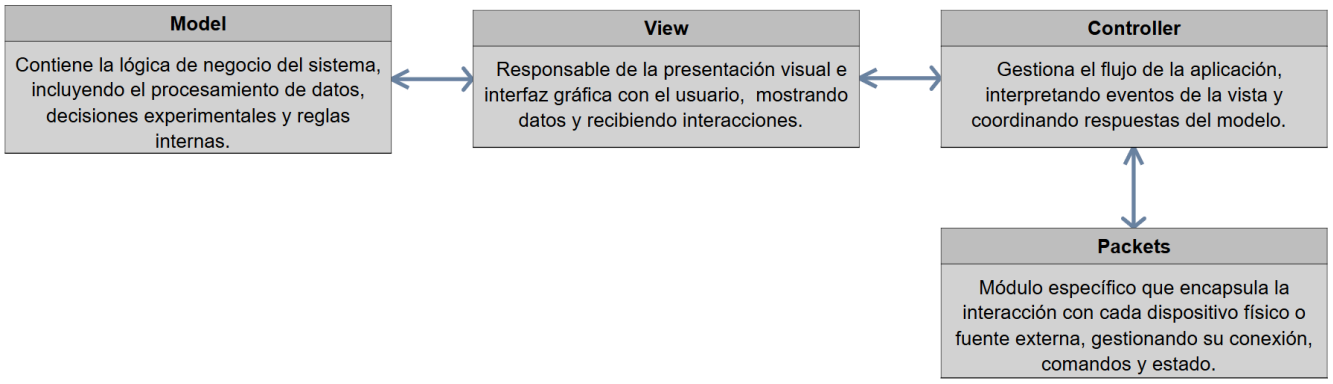

Figura 9: Representación esquemática de la arquitectura MVC-P.

La decisión de estructurar el software según este patrón se tomó desde las primeras fases de desarrollo, tras evaluar la diversidad de dispositivos y protocolos implicados, así como los requerimientos de claridad, mantenibilidad y robustez que se esperaban de la aplicación final.

4.2.2 Arquitectura interna del software

La estructura interna del código ha sido diseñada para reflejar la arquitectura lógica descrita anteriormente, facilitando la comprensión de sus componentes y su posible ampliación. Esta sección tiene como objetivo ofrecer una visión clara del diseño modular implementado, así como ilustrar las interacciones entre las distintas capas del sistema.

Para ello, se ha incluido un **diagrama de clases general** que representa las entidades principales y sus relaciones, y se proporciona una descripción detallada de cada uno de los módulos arquitectónicos: Modelo, Vista, Controlador y Packets. A través de este análisis se pretende mostrar cómo se implementan los principios de diseño en el código fuente y cómo se ha logrado una separación eficaz de responsabilidades.

4.2.2.1 Diagrama de clases general

A continuación, se presenta el diagrama de clases del sistema desarrollado **(Figura 10)**. Este diagrama proporciona una visión general estructural de alto nivel, mostrando cómo se organizan e interrelacionan los distintos módulos del proyecto. Este diagrama no detalla los atributos ni métodos de las clases, sino que ilustra las principales relaciones entre módulos, la organización jerárquica del software y el grado de modularidad logrado en su implementación. Sirve como punto de partida para comprender la arquitectura del sistema antes de analizar con mayor detalle cada uno de sus componentes.

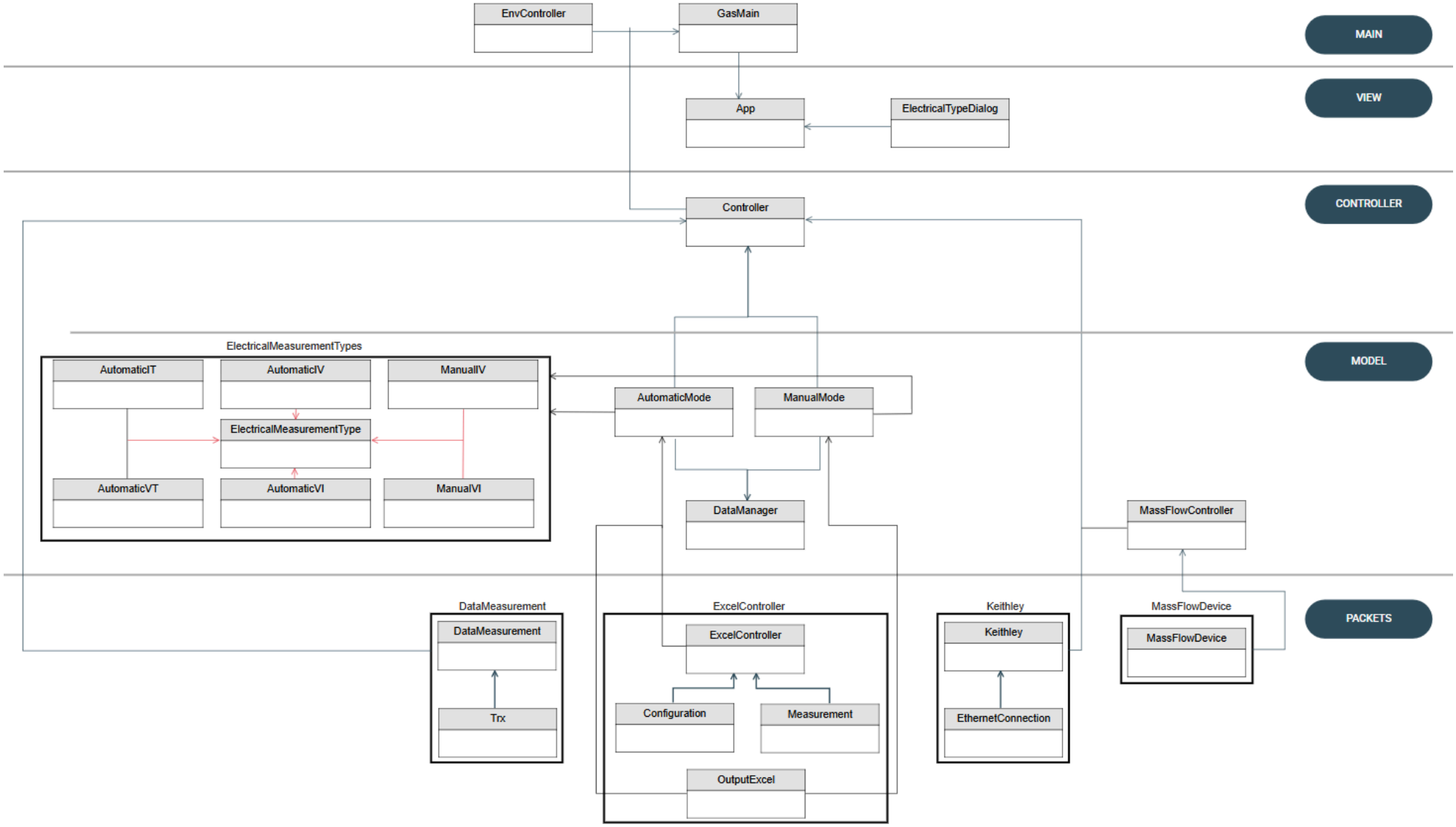
EnvController
GasMain
MAIN
App
ElectricalTypeDialog
VIEW
Controller
CONTROLLER
ElectricalMeasurementTypes
AutomaticIT
AutomaticIV
ManualIV
ElectricalMeasurementType
AutomaticVT
AutomaticVI
ManualVI
AutomaticMode
ManualMode
DataManager
MODEL
MassFlowController
DataMeasurement
DataMeasurement
Trx
ExcelController
ExcelController
Configuration
Measurement
OutputExcel
Keithley
Keithley
EthernetConnection
MassFlowDevice
MassFlowDevice
PACKETS

Figura 10: Diagrama general de clases.

El sistema se encuentra organizado en cinco bloques principales:

- **Main:** Contiene las clases encargadas del arranque del sistema.
- **View:** Agrupa los elementos relacionados con la interfaz gráfica.
- **Controller:** Centraliza la lógica de control del flujo del programa.
- **Model:** Implementa la lógica de negocio y el procesamiento de datos.
- **Packets:** Engloba los módulos que interactúan con dispositivos y recursos externos.

Los recuadros de bordes más gruesos, como los de *DataMeasurement, ExcelController, Keithley, MassFlowDevice* y *ElectricalMeasurementTypes,* representan **carpetas** dentro del proyecto, cada una conteniendo un conjunto de clases que están relacionadas.

Las flechas de color rojo indican relaciones de herencia. Estas se utilizan exclusivamente en el módulo *ElectricalMeasurementTypes*, donde las clases *ManualIV, ManualVI, AutomaticIV, AutomaticVI, AutomaticIT* y *AutomaticVT,* heredan de la clase base *ElectricalMeasurementrType*, que actúa como clase abstracta.

Las flechas negras representan dependencias y relaciones de uso mediante importación. La dirección de la flecha indica que la clase destino, donde apunta la flecha, importa y utiliza a la clase origen. Esta notación permite visualizar las dependencias entre los distintos módulos del sistema.

El diseño del sistema busca reflejar una **separación de responsabilidades** a través del uso del patrón MVC-P, y demuestra un esfuerzo por alcanzar una arquitectura modular y escalable. A continuación, se presentan con mayor detalle los diagramas de clases correspondientes a cada uno de los módulos mencionados, mostrando, en la medida de lo posible, las clases específicas y sus métodos.

4.2.2.2 Diagrama de clases del módulo Main

El módulo Main es responsable de la inicialización del entorno de ejecución del sistema y de asegurar que todas las dependencias necesarias estén instaladas y listas para usar **(Figura 11)**.

Este módulo encapsula toda la lógica necesaria para preparar el entorno en el que se ejecutará la aplicación, lo cual resulta fundamental para garantizar reproducibilidad y facilidad de instalación del sistema.

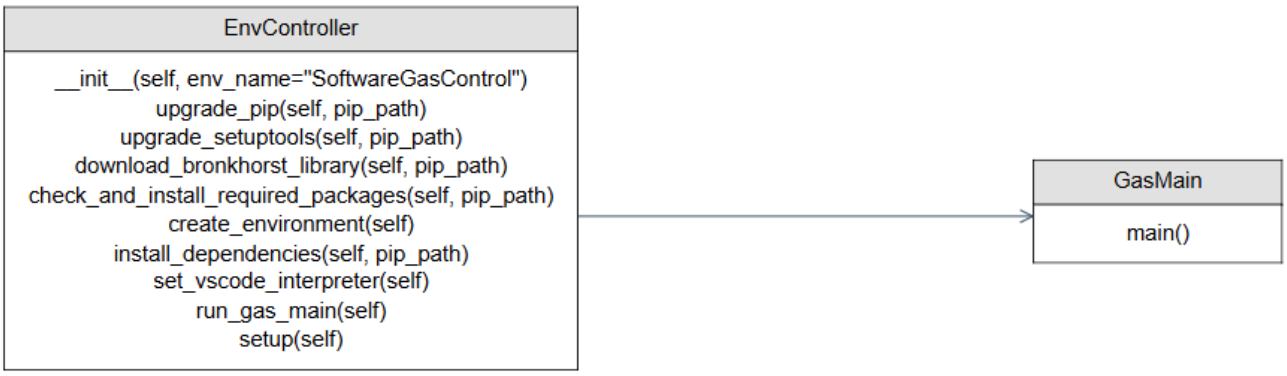

Figura 11: Diagrama de clases del módulo Model.

A continuación, explicaremos brevemente qué hace cada clase, para tener una visión de la funcionalidad y de su importancia.

- **GasMain:** Se encarga de preparar el entorno de ejecución y lanzar la aplicación principal. Su función principal es iniciar el controlador y la vista, asegurando un cierre ordenado de los dispositivos y recursos al finalizar.
- **EnvController:** Gestiona la creación y configuración automática del entorno virtual necesario para la aplicación, asegurando la instalación y actualización de todas las dependencias requeridas. También se encarga de preparar el entorno para el desarrollo y ejecutar la aplicación principal de forma controlada.

4.2.2.3 Diagrama de clases del módulo View

El módulo View representa la capa de interfaz gráfica del sistema, permitiendo la interacción directa del usuario con las funcionalidades experimentales **(Figura 12)**.

Este módulo está compuesto por clases que gestionan la visualización de datos, configuración de mediciones y selección de parámetros mediante elementos gráficos.

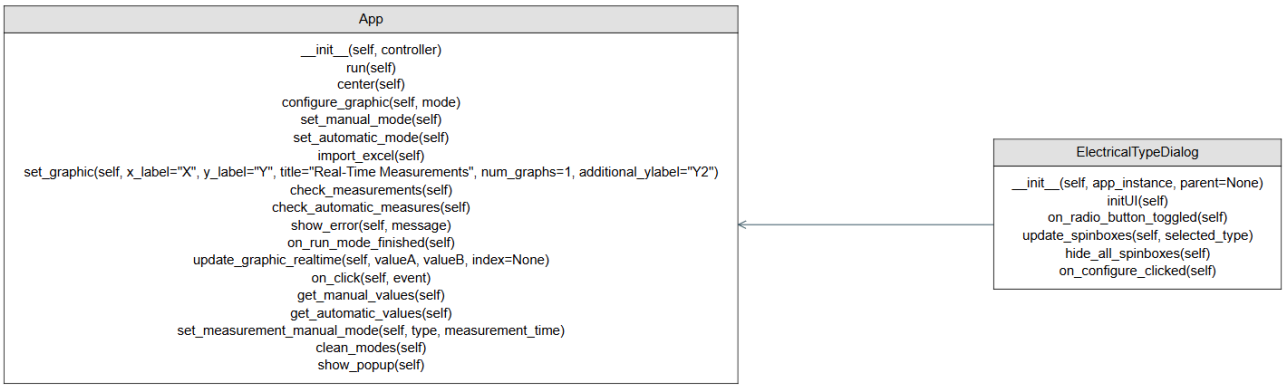

Figura 12: Diagrama de clases del módulo View.

A continuación, explicaremos brevemente qué hace cada clase, para tener una visión de la funcionalidad y de su importancia.

- **App:** Es el núcleo de la aplicación que inicia la interfaz gráfica y gestiona la interacción del usuario con el sistema. Coordina la comunicación entre la GUI y la lógica de control, facilitando la configuración, ejecución y visualización de las mediciones.
- **ElectricalTypeDialog:** Ofrece una ventana dinámica para que el usuario configure el modo de ejecución deseado, adaptando los campos de entrada según el tipo de medición seleccionado.

4.2.2.4 Diagrama de clases del módulo Controller

El módulo Controller actúa como capa de coordinación del sistema, encargada de llevar a cabo la ejecución de los experimentos mediante la configuración y gestión de los distintos modos de operación, flujos, equipos de medición y parámetros **(Figura 14)**.

Este módulo ofrece una interfaz para iniciar, configurar y ejecutar mediciones tanto manuales como automáticas, centralizando el control a través de su clase principal, Controller. Su diseño encapsula la lógica de control de alto nivel, desacoplando las decisiones operativas de los detalles específicos del hardware.

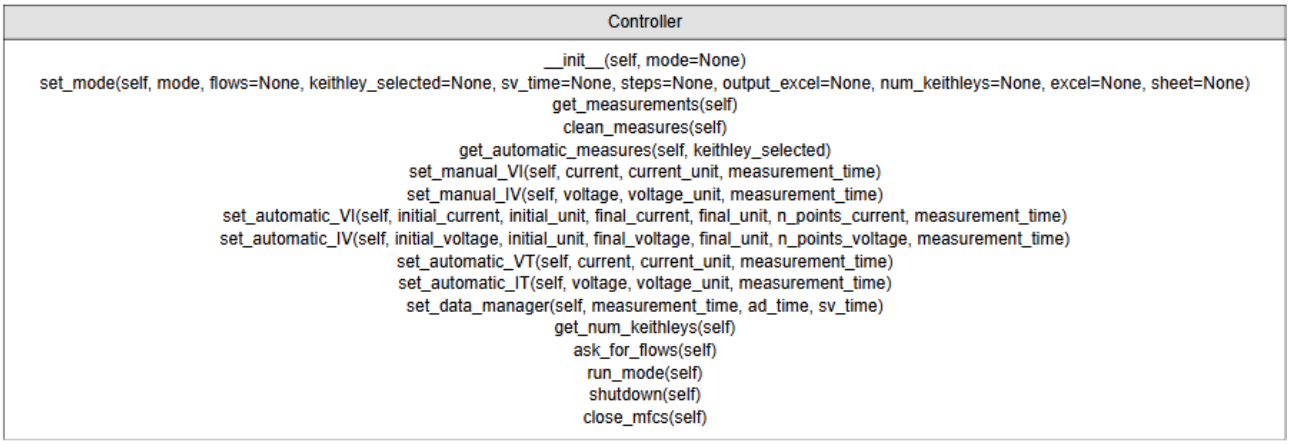

Figura 13: Diagrama de clases del módulo Controller.

A continuación, explicaremos brevemente qué hace cada clase, para tener una visión de la funcionalidad y de su importancia.

- **Controller:** Gestor central que controla y coordina la operación de diferentes modos de medición, gestionando dispositivos de medición y control de flujo. Permite configurar, ejecutar y obtener datos de las mediciones, además de gestionar la comunicación con los instrumentos conectados.

4.2.2.5 Diagrama de clases del módulo Model

El módulo Model es uno de los pilares fundamentales del sistema, ya que agrupa las clases responsables de definir y coordinar la lógica de mediciones eléctricas y el control de flujo en experimentos automatizados o manuales **(Figura 14)**.

Este módulo constituye el núcleo lógico de las operaciones experimentales, definiendo la forma en que se ejecutan las mediciones y coordinando la interacción entre los distintos componentes del sistema. Su diseño permite organizar y ejecutar mediciones de manera estructurada en los distintos modos.

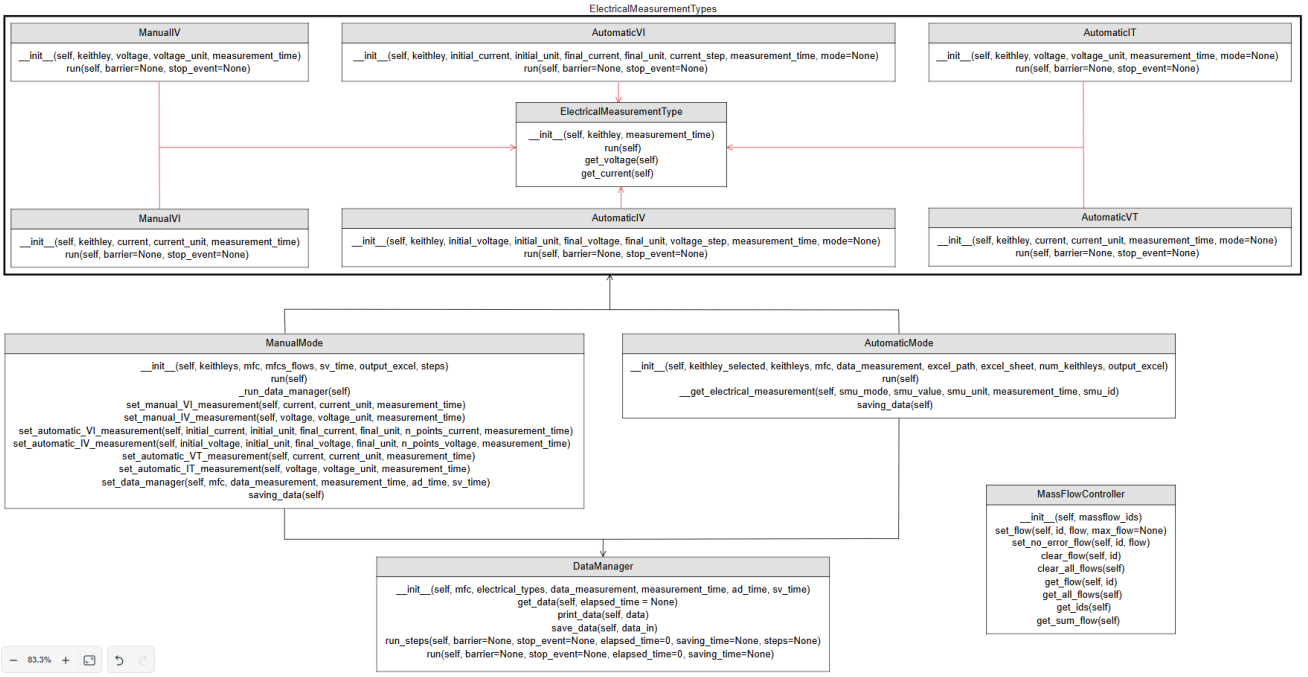

Figura 4: Diagrama de clases del módulo Model.

A continuación, explicaremos brevemente qué hace cada clase, para tener una visión de la funcionalidad y de su importancia.

- **ElectricalMeasurementType:** Es una clase base que define la estructura común para realizar mediciones eléctricas con un equipo Keithley. Su función principal es gestionar los valores y unidades de voltaje y corriente, sirviendo como plantilla para distintos tipos específicos de mediciones eléctricas. Los modos de medición que heredan de esta clase son:
    - ***ManualIV:*** En este modo se aplica un voltaje constante y se registra la corriente resultante en ese instante. A diferencia de los modos automáticos, simplemente se aplica el voltaje indicado y se guarda la lectura correspondiente. Es útil cuando se quiere hacer una medición puntual y rápida.

- ***ManualVI:*** En este modo se aplica una intensidad constante y se registra el voltaje resultante en ese instante. A diferencia de los modos automáticos, simplemente se aplica la intensidad indicada y se guarda la lectura correspondiente. Es útil cuando se quiere hacer una medición puntual y rápida.

- ***AutomaticIV:*** En este modo se realiza un barrido de voltaje entre dos valores (inicial y final) y mide la corriente en cada paso. Se generan varios puntos de voltaje, se aplican uno a uno al equipo y se registra la corriente que circula en cada caso. Es útil para obtener la curva I-V de un componente.

- ***AutomaticVI:*** En este modo se realiza un barrido de intensidad entre dos valores (inicial y final) y mide el voltaje en cada paso. Se generan varios puntos de intensidad, se aplican uno a uno al equipo y se registra el voltaje que circula en cada caso. Es útil para obtener la curva V-I de un componente.

- ***AutomaticIT:*** En este modo se aplica un voltaje fijo durante un tiempo determinado, mientras se mide la corriente en función del tiempo. Durante toda la duración configurada, va registrando la corriente solo si detecta cambios significativos respecto a la medición anterior. Esto permite ahorrar datos y centrarse donde realmente hay variaciones relevantes en la corriente.

- ***AutomaticVT:*** En este modo se aplica una intensidad fija durante un tiempo determinado, mientras se mide el voltaje en función del tiempo. Durante toda la duración configurada, va registrando el voltaje solo si detecta cambios significativos respecto a la medición anterior. Esto permite ahorrar datos y centrarse donde realmente hay variaciones relevantes en el voltaje.

- **ManualMode:** Gestiona la ejecución de un modo de operación manual para un sistema de medición experimental. Se encarga de configurar y sincronizar la operación de diferentes instrumentos, coordinando la adquisición de datos eléctricos y ambientales de forma paralela mediante hilos.

- **AutomaticMode:** Gestiona la ejecución automática de mediciones, coordinando varios instrumentos. Lee una configuración de medición desde un archivo Excel, y ejecuta las mediciones eléctricas.

- **DataManager:** Se encarga de coordinar la recopilación y almacenamiento de datos provenientes de distintos dispositivos de medición. Su función principal es centralizar la adquisición de datos en tiempo real, organizar estos datos para un fácil acceso y guardarlos durante procesos de medición que pueden ser continuos o por etapas, facilitando así el monitoreo.

- **MassFlowController:** Permite inicializar varios controladores de flujo mediante sus identificadores, establecer y obtener el flujo individual de cada uno, limpiar flujos específicos o todos a la vez, y consultar el listado de IDs conectados.

4.2.2.6 Diagrama de clases del módulo Packets

El módulo **Packets** es uno de los pilares fundamentales del sistema, ya que agrupa las clases responsables de la interacción con hardware y dispositivos externos, así como sensores, controladores de flujo, equipos de medición **(Figura 15).**

Este módulo facilita la abstracción y encapsulación de la comunicación a bajo nivel, permitiendo que otras capas del sistema interactúen con los dispositivos de forma uniforme y desacoplada. Su diseño se caracteriza por una organización modular y una clara separación de responsabilidades entre los diferentes dispositivos gestionados.

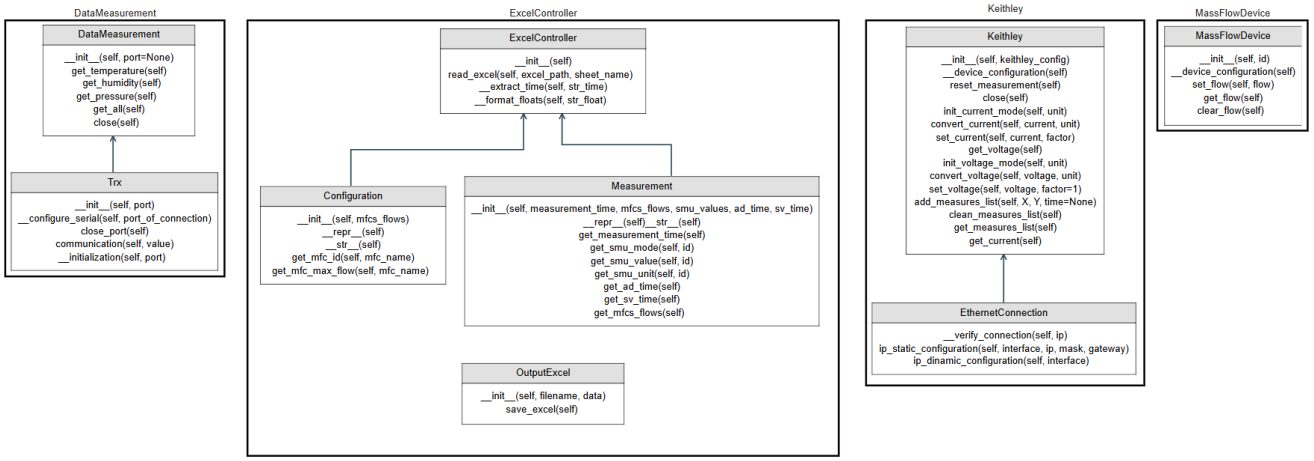

Figura 15: Diagrama de clases del módulo Packets.

A continuación, explicaremos brevemente qué hace cada clase, para tener una visión de la funcionalidad y de su importancia.

- **DataMeasurement:** Se encarga de obtener los valores de temperatura, humedad y presión desde un dispositivo físico a través de comunicación serie. También gestiona la conexión con el puerto y permite simular los datos en ausencia de hardware.

- **Trx:** Gestiona la comunicación serie con los dispositivos físicos, encargándose de abrir, cerrar y enviar comandos al puerto especificado, asegurando una transmisión y recepción fiables de datos.

- **ExcelController:** Procesa archivos Excel que contienen configuraciones y medidas experimentales, extrayendo y estructurando la información en objetos Configuration y Measurement para su posterior uso en el sistema.

- **Configuration:** Almacena y gestiona los parámetros de configuración de los controladores de flujo, incluyendo su identificación, flujo máximo y concentración de gas, facilitando el acceso a estos datos para el sistema.

- **Measurement:** Representa una medición concreta, almacenando el tiempo de la medida, los flujos de los controladores de flujo, los valores y modos de los SMU, así como tiempos asociados de adquisición y estabilización.
- **OutputExcel:** Se encarga de recibir un conjunto de datos, convertirlo en un DataFrame de pandas y preparar el archivo Excel para guardarlo.
- **Keithley:** Controla y gestiona el instrumento Keithley a través de conexión Ethernet, permitiendo configurar modos de fuente y medición de corriente y voltaje, además de almacenar y manejar las medidas realizadas durante los experimentos.
- **EthernetConnection:** Se encarga de configurar la conexión Ethernet del instrumento Keithley, permitiendo establecer direcciones IP estáticas o dinámicas y verificando la conectividad mediante comandos del sistema operativo.
- **MassFlowDevice:** Se encarga de configurar la conexión con un controlador de flujo, establecer y leer el flujo de gas, y asegurar la consistencia de los valores mediante reintentos y validaciones.

4.3 Gestión de hilos y concurrencia

Es fundamental gestionar correctamente la concurrencia en aplicaciones que interactúan con varios dispositivos físicos y, al mismo tiempo, deben mantener una interfaz gráfica ágil. Esto permite que el sistema ejecute varias tareas simultáneamente, aprovechando mejor los recursos del procesador. Así se garantiza una respuesta rápida a los distintos eventos del sistema, evitando bloqueos o retrasos que afectarían a la experiencia de uso.

En nuestro sistema experimental, la concurrencia es especialmente relevante por la necesidad de mantener una adquisición continua de datos, compuesta por señales eléctricas, temperatura, humedad o flujos de gases, sin comprometer la interfaz de usuario. Esto se logra mediante la implementación de **hilos de ejecución (threads)** que operan en paralelo a la interfaz gráfica y a otros procesos del sistema [26].

Para evitar problemas como las condiciones de carrera, accesos simultáneos a recursos compartidos o inconsistencias en los datos, el sistema implementa distintos **mecanismos de sincronización** que permiten coordinar de forma segura la ejecución de múltiples hilos. Estos mecanismos actúan como herramientas de control que aseguran que las operaciones sensibles se ejecuten de manera ordenada y sin interferencias, preservando la integridad del sistema y los datos.

Entre los mecanismos utilizados en nuestro software experimental destacan los siguientes:

- **Barriers**: son puntos de sincronización en los que un conjunto de hilos debe detener su ejecución hasta que todos los hilos implicados hayan alcanzado ese mismo punto. Este recurso resulta útil cuando es necesario garantizar que varios procesos paralelos han finalizado una fase determinada antes de continuar a la siguiente. En nuestro sistema, por ejemplo, pueden emplearse para coordinar la adquisición de datos simultánea de varios instrumentos antes de proceder al análisis conjunto.

- **Events**: permiten establecer una comunicación flexible entre hilos mediante señales que activan o suspenden la ejecución de determinadas tareas. Un hilo puede, por ejemplo, esperar a que se dispare un evento para iniciar un proceso crítico, como el envío de un nuevo conjunto de parámetros al instrumento. Esta técnica es especialmente útil en contextos donde se requiere sincronización dinámica y reactiva entre distintas partes del sistema.

La gestión robusta de estos elementos no solo mejora la fiabilidad del sistema, sino que también aporta escalabilidad, al permitir fácilmente la integración de nuevos dispositivos o modos de operación. A continuación, se describen con detalle los dos modos principales de funcionamiento del sistema: **modo manual** y **modo automático**, explicando cómo se organiza la concurrencia en cada caso.

4.3.1 Modo Manual

En el modo manual, el sistema ejecuta dos tareas concurrentes esenciales para garantizar un control completo y en tiempo real del experimento: por un lado, la medición eléctrica definida por el usuario, y por otro, la recopilación continua de datos ambientales y de flujo, que incluye variables como la temperatura y humedad de la cámara, así como el flujo proporcionado por los controladores de flujo **(Figura 16).**

Para asegurar que ambas tareas se desarrollen en paralelo sin interferir con la fluidez de la interfaz gráfica ni bloquear la ejecución general del programa, se implementan dos hilos independientes, los cuales se coordinan mediante una barrera de sincronización. Este mecanismo garantiza que ambas tareas se inicien de manera controlada una vez establecidas las condiciones iniciales del sistema, como la estabilización del flujo de gases.

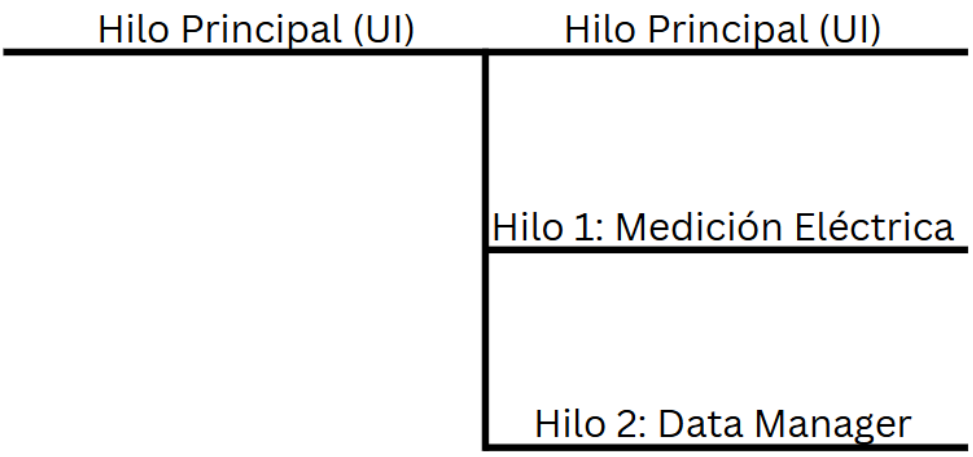

Figura 16: Diagrama de los hilos activos en el modo manual.

Una vez configurado el flujo en los MFCs, se inicializan los dos hilos principales del modo manual:

- El hilo de **medición eléctrica**, se encarga de ejecutar la medida específica seleccionada por el usuario. Este hilo se comunica directamente con el instrumento Keithley 2450, configurando parámetros como la tensión o corriente y capturando las mediciones correspondientes. Su ejecución está optimizada para operar de manera continua y precisa, proporcionando al usuario lecturas eléctricas en tiempo real sin interrupciones.

- El hilo de **gestión de datos**, se encarga de recopilar periódicamente las condiciones experimentales de entorno. Esto incluye lecturas de temperatura y humedad dentro de la cámara Linkam, así como los valores de flujo configurados en los MFCs Bronkhorst. Toda la información recolectada es almacenada en estructuras de datos ordenadas, lo que facilita su posterior análisis, exportación y visualización en la interfaz.

Ambos hilos se sincronizan mediante una barrera de dos hilos, que garantiza que cada iteración de los hilos se realice de manera coordinada. De este modo, por ejemplo, los datos ambientales y los valores eléctricos quedan alineados temporalmente, evitando desajustes que comprometan la calidad de los resultados.Además, se utiliza un evento de parada que permite detener ambos hilos de manera controlada, ya sea al completar las medidas o si se solicita una interrupción externa.

Para implementar la ejecución concurrente en el modo manual, se utiliza la biblioteca estándar **threading** de Python, que permite crear y gestionar hilos de manera eficiente. En este fragmento de código se observa cómo se establece una barrera de sincronización para coordinar el inicio simultáneo de los dos hilos principales, y cómo se crean, inician y esperan ambos hilos de forma controlada **(Figura 17).**

```
# Crear una barrera para sincronizar los dos hilos (medición y data manager)
self.barrier = Barrier(2)

# Asegurar que la señal de detención esté limpia
self.stop_event.clear()

# Crear los hilos para medición eléctrica y data manager
measurement_thread = Thread(target=self.electrical_measurement.run,
                            args=(self.barrier, self.stop_event))
data_manager_thread = Thread(target=self._run_data_manager)

# Iniciar ambos hilos
data_manager_thread.start()
measurement_thread.start()

# Esperar a que ambos hilos terminen
measurement_thread.join()
data_manager_thread.join()
```

Figura 17: Fragmento de código que muestra la creación y sincronización de los hilos principales.

El primer hilo ejecuta la rutina de medición eléctrica, pasando como argumentos la barrera y un evento de parada que permite detener la ejecución de forma segura si es necesario. El segundo hilo se encarga de gestionar la adquisición continua de datos mediante el método **_run_data_manager()**, que a su vez realiza llamadas periódicas al método **get_data()**. Este método recopila datos como temperatura, humedad, flujo de gases y mediciones eléctricas de las unidades Keithley, construyendo una estructura completa y sincronizada para su posterior análisis o almacenamiento.

Gracias a este enfoque, el sistema puede realizar múltiples tareas críticas de forma paralela sin bloquear la interfaz gráfica ni comprometer la precisión o integridad de los datos.

4.3.2 Modo Automático

En el modo automático, el sistema extiende la funcionalidad del modo manual para permitir la ejecución secuencial y programada de múltiples puntos de medida. Este modo ha sido diseñado para ofrecer una mayor flexibilidad y eficiencia en los experimentos, ya que permite definir series de medidas automáticas y, además, soporta tanto la conexión de un único instrumento Keithley 2450 como de dos dispositivos funcionando en paralelo.

Dependiendo del número de instrumentos conectados, el sistema ajusta automáticamente el número de hilos y la barrera de sincronización necesarios para garantizar una ejecución ordenada y simultánea de todas las tareas críticas. Esta arquitectura permite mantener la coherencia temporal entre las medidas eléctricas y las condiciones ambientales del entorno de la muestra **(Figura 18).**

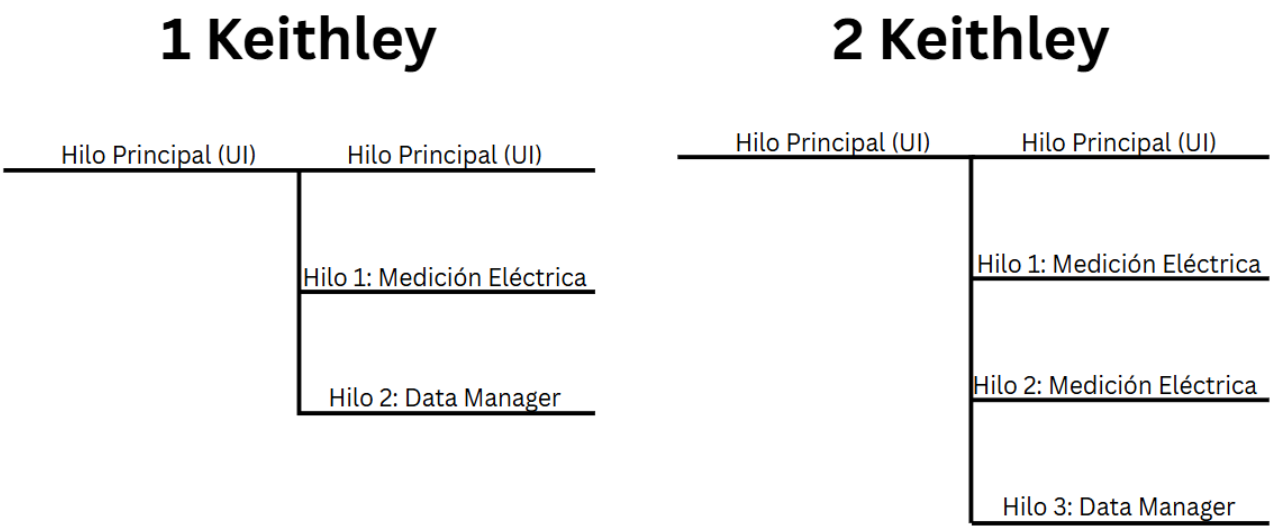

Figura 18: Diagrama de los hilos activos en el modo automático.

Cuando se detecta un único instrumento Keithley, el comportamiento del sistema es idéntico al del modo manual: se crean dos hilos, uno para la medición eléctrica y otro para la gestión de datos, sincronizados mediante una barrera de dos hilos. En este caso, se ejecuta el mismo flujo descrito anteriormente para garantizar una adquisición en tiempo real de los parámetros experimentales.

Una vez configurado el flujo en los MFCs, se inicializan tres hilos principales en el modo automático con dos Keithleys conectados:

- Dos hilos de **medición eléctrica**, cada uno asociado a un instrumento Keithley 2450 distinto. Ambos hilos operan de forma independiente pero sincronizada, ejecutando las medidas eléctricas definidas por el usuario en paralelo. Cada hilo configura los parámetros correspondientes (como tensión o corriente) y recoge las lecturas de forma continua y precisa, maximizando la eficiencia del proceso experimental.

- El hilo de **gestión de datos**, al igual que en el modo manual, se encarga de recopilar de forma periódica las condiciones experimentales del entorno. Incluye la lectura de temperatura y humedad dentro de la cámara, así como el control de flujo de los MFCs. Toda esta información se almacena junto con las mediciones eléctricas de ambos Keithleys en una estructura de datos coherente y sincronizada.

En el modo automático, cuando se utilizan dos instrumentos Keithley 2450, el sistema lanza tres hilos principales que se sincronizan mediante una barrera de tres hilos. Esta barrera asegura que cada iteración de las tareas se ejecute de forma coordinada, permitiendo que las dos mediciones eléctricas y la adquisición de datos ambientales queden perfectamente alineadas en el tiempo. Esto es crucial para mantener la consistencia de los datos recogidos y evitar desajustes entre las señales eléctricas y las condiciones experimentales.

Además, se emplea un evento de parada que permite detener todos los hilos de forma segura y controlada, ya sea al finalizar la secuencia de mediciones o ante una interrupción manual o por error.

Para la ejecución concurrente, se sigue utilizando la biblioteca estándar **threading** de Python, que facilita la gestión y sincronización de los hilos. En el siguiente fragmento de código se muestra cómo se define la barrera, se crean los tres hilos principales y cómo estos se inician y sincronizan de forma estructurada **(Figura 19)**.

```
# Crear una barrera para sincronizar los dos hilos (medición y data manager)
self.barrier = Barrier(3)

electrical_measurement_1 = self.__get_electrical_measurement(smu_1_mode, smu_1_value,
                                                             smu_1_unit, measurement_time, 0)
electrical_measurement_2 = self.__get_electrical_measurement(smu_2_mode, smu_2_value,
                                                             smu_2_unit, measurement_time, 1)

# Creación de hilos para medir
self.thread_1 = Thread(target=electrical_measurement_1.run, args=(self.barrier, self.stop_event))
self.thread_2 = Thread(target=electrical_measurement_2.run, args=(self.barrier, self.stop_event))
self.thread_1.start()
self.thread_2.start()

# Creación del DataManager
data_manager = DataManager(self.mfc, [electrical_measurement_1, electrical_measurement_2],
                           self.data_measurement, measurement_time, ad_time, sv_time)
if self.start_time is None:
    self.start_time = time.time()
elapsed_time = time.time() - self.start_time

try:
    data = data_manager.run2(self.barrier, elapsed_time=elapsed_time, saving_time=sv_time)
    self.data.append(data)  # Guardar datos en cada iteración

    measurement_data.append(data)  # Acumulando datos de esta medición

except Exception as e:
    print(f"{colored('[ERROR]', 'red')} Error in measurement {i+1} point: {e}")
    # Guardar los datos acumulados hasta el fallo
    self.data.append(measurement_data)  # Guardar lo acumulado hasta el fallo

finally:
    self.thread_1.join()
    self.thread_2.join()
```

Figura 19: Fragmento de código que muestra la creación y sincronización de los hilos principales.

El primer y segundo hilo ejecutan las rutinas de medición eléctrica para cada Keithley, recibiendo como argumentos tanto la barrera de sincronización como el evento de parada. Cada hilo configura su respectivo instrumento y recoge las medidas correspondientes de forma continua y precisa.

El tercer hilo es el encargado de la gestión de datos experimentales. A través del método **run2()**, se realiza una adquisición periódica de las condiciones de temperatura, humedad y flujo, así como de las mediciones eléctricas acumuladas en los dos Keithleys. Estos datos se organizan en una estructura temporalmente coherente y se almacenan para su posterior análisis.

Gracias a este enfoque multihilo, el sistema puede gestionar simultáneamente dos mediciones eléctricas independientes y una adquisición ambiental, todo ello sin bloquear la interfaz gráfica ni comprometer la calidad ni la sincronización de los datos experimentales.

4.4 Flujo principal de ejecución

El flujo de ejecución principal del software ha sido diseñado para ofrecer una experiencia de usuario intuitiva y flexible, facilitando el control y configuración del experimento a través de dos modos de operación claramente diferenciados: manual y automático.

En primer lugar, el proceso se inicia con la presentación de una interfaz inicial que permite al usuario seleccionar el modo de operación más adecuado según las características y objetivos del experimento. Acto seguido, el sistema activa el flujo de ejecución correspondiente al modo seleccionado, gestionando de manera eficiente la adquisición de datos y la comunicación con los dispositivos físicos involucrados.

Este enfoque modular no solo facilita la adaptabilidad del software a diferentes necesidades experimentales, sino que también garantiza la integridad y coherencia de los datos adquiridos en todo momento. En las siguientes secciones se detalla el funcionamiento de la interfaz inicial, así como las particularidades y procesos internos específicos de cada modo de operación.

4.4.1 Vista general

Al iniciar el programa, el usuario observa una interfaz gráfica que representa el punto de entrada principal al sistema. Esta vista inicial **(Figura 20)**, recoge las configuraciones básicas necesarias para arrancar cualquiera de los dos modos de funcionamiento disponibles: modo manual y modo automático.

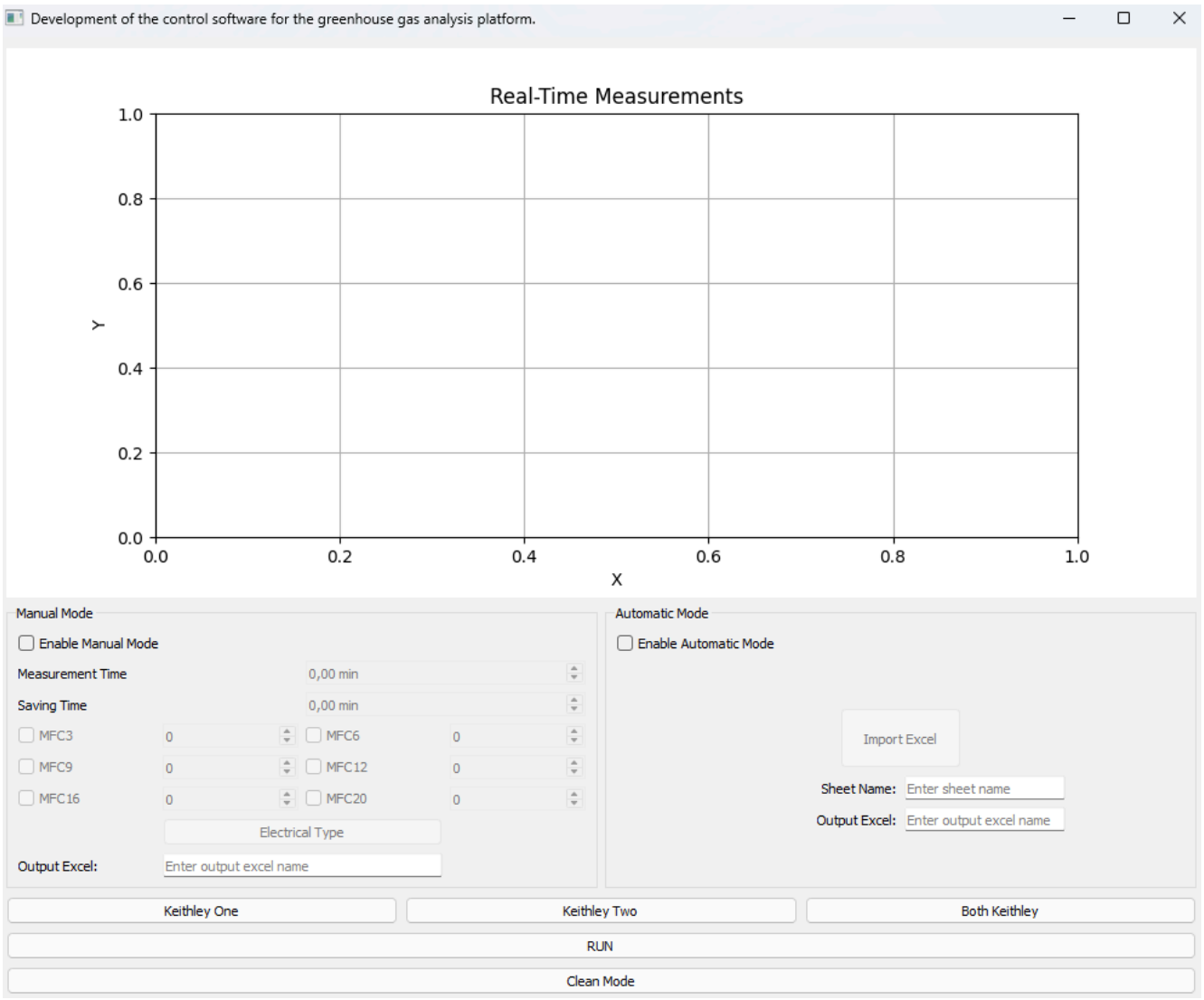

Figura 20: Imagen de la primera vista de la interfaz gráfica.

En la parte superior de la ventana se encuentra una gráfica que se utilizará para visualizar en tiempo real las mediciones obtenidas durante la ejecución. Debajo de esta, la interfaz se divide en dos secciones claramente diferenciadas:

- **Modo manual:** En esta sección, el usuario puede habilitar el modo marcando la casilla correspondiente y debe completar parámetros clave como el tiempo de medición, los tiempos de guardado para cada controlador de flujo, en caso de querer aplicar, y el tipo de medida eléctrica, cuya selección le llevará a otra ventana para introducir estos datos. Además, debe introducir el nombre del archivo de salida donde se guardarán los resultados, en caso contrario, se guardará como *MeasureFechaActual*, por defecto.

- **Modo automático:** En esta sección, el usuario puede activar el modo automático, importar un archivo Excel con la configuración experimental, el cual tendrá definida todas las mediciones que se llevarán a cabo, y deberá introducir el nombre de la hoja del Excel que se desea usar, y definir el nombre del archivo de salida para los datos obtenidos, en caso contrario, las medidas se guardarán como *AutomaticMeasureFecha*, y la hoja de excel será *data.*

Una vez completados los campos requeridos del modo seleccionado, el usuario debe indicar cuántos dispositivos **Keithley** desea utilizar para la adquisición eléctrica (uno, dos, o ambos), seleccionando uno de los botones inferiores: *Keithley One*, *Keithley Two* o *Both Keithley*. Si todos los datos están correctamente introducidos, el proceso se iniciará pulsando el botón **RUN MODE**.

En caso de que el usuario desee reiniciar el proceso o haya cometido un error durante la configuración, puede utilizar el botón **Clean Mode**, que restablece todos los campos a su estado inicial y permite comenzar desde cero.

Este punto común de interacción representa el núcleo del flujo de ejecución, donde se define la lógica inicial que determina el desarrollo posterior del sistema. En los siguientes apartados se detallarán las particularidades de cada modo y los procesos que se desencadenan tras la activación del botón RUN.

4.4.2 Ajustes de parámetros en el modo manual

Tal como se ha mencionado anteriormente, el sistema puede operar en dos modos distintos. En el **modo manual**, el usuario tiene control total sobre los parámetros experimentales, los cuales deben ser definidos a través de la interfaz gráfica antes de iniciar la ejecución. A continuación, se describen los principales parámetros configurables:

- **Measurement Time:** Define la duración total de la medida. Este valor determina el tiempo durante el cual se llevará a cabo la adquisición de datos eléctricos y ambientales.

- **Saving Time:** Establece el intervalo temporal con el que el sistema almacenará los datos recopilados en un diccionario estructurado. Es imprescindible que este valor sea inferior al tiempo de medida para evitar conflictos y asegurar una frecuencia adecuada de guardado.

- **MFCX:** Disponemos de un botón independiente para cada controlador de flujo, permitiendo al usuario seleccionar cuáles desea activar durante la medida. Cada MFC tiene un flujo máximo predefinido, y el sistema impide introducir valores superiores a este límite.

  Es obligatorio seleccionar al menos un MFC para cada experimento, aunque el flujo configurado para dicho dispositivo puede ser igual a cero.

- **ElectricalType:** Al seleccionar esta opción, se abre una ventana adicional que permite al usuario configurar con precisión el tipo de medida eléctrica que desea realizar **(Figura 21)**.

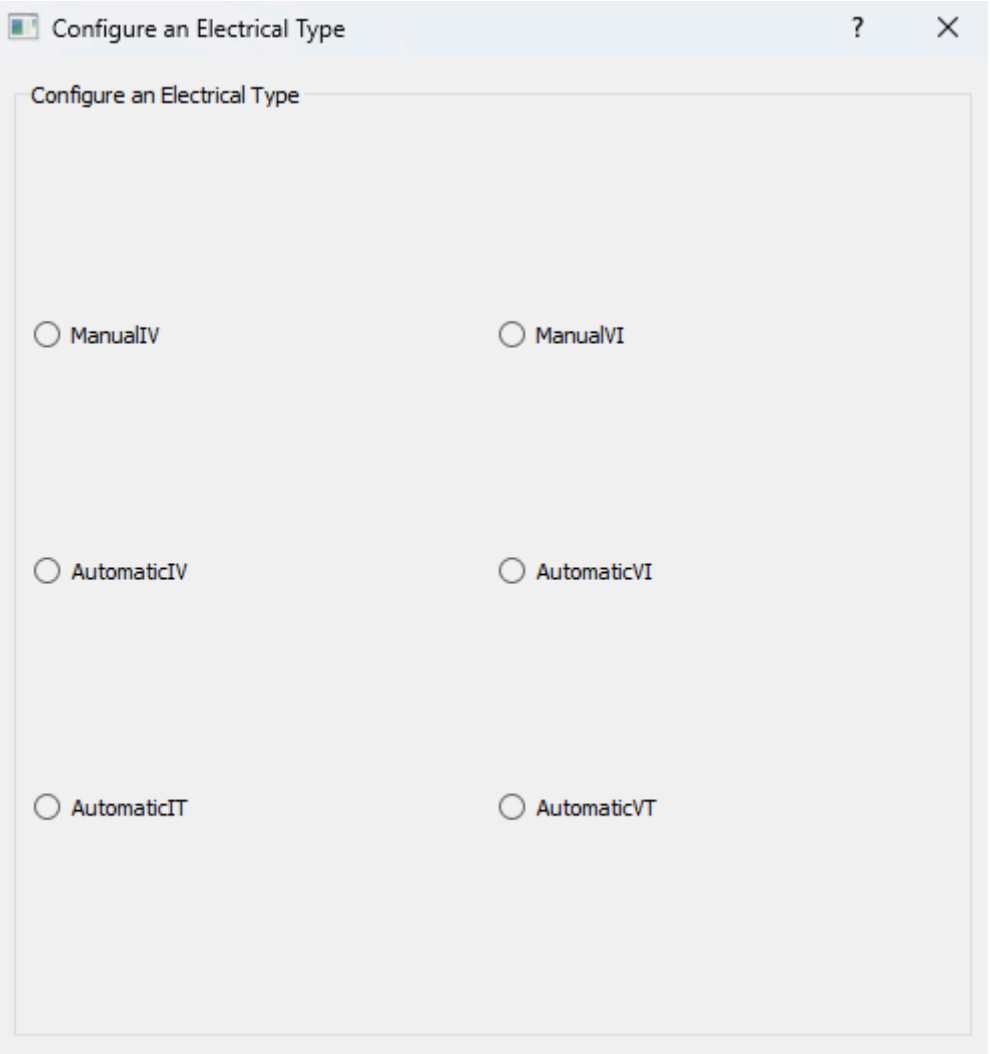

Figura 21: Ventana de configuración de modo de medición.

En esta ventana se presentan distintos modos de medida, descritos previamente en la sección de diagramas de clase.

La interfaz se actualiza dinámicamente según el modo elegido, cargando únicamente los parámetros necesarios para cada uno y minimizando así posibles errores de configuración.

- **OutputExcel:** Este campo permite al usuario asignar un nombre personalizado al archivo de salida que almacenará los datos experimentales. Si no se introduce ningún nombre, el sistema asignará uno por defecto y lo almacenará en la carpeta predeterminada del proyecto */data*.

4.4.3 Ajustes de parámetros en el modo automático

A primera vista, el **modo automático** puede parecer más sencillo de configurar que el modo manual. Sin embargo, esto no es del todo cierto, ya que toda la configuración necesaria debe definirse previamente mediante una estructura específica dentro de un archivo Excel denominado *llibre_Input_Output_Estacio_gasos*, el cual se almacena por defecto en la carpeta /*data*.

Antes de explicar cómo debe estructurarse correctamente este archivo Excel, se describen los principales parámetros configurables a través de la interfaz gráfica:

- **Import Excel:** Este parámetro permite al usuario seleccionar el archivo Excel que desea importar al sistema. Generalmente, estos archivos se encuentran en la carpeta /*data*, donde se almacenan tanto configuraciones como resultados de las mediciones anteriores. Es fundamental que el archivo importado siga la estructura esperada por el sistema, de otras forma, no se ejecutará la medición y saltará un error..
- **Sheet Name:** Permite especificar qué hoja del archivo Excel se debe analizar. Esta funcionalidad ofrece flexibilidad al usuario, ya que permite la existencia de múltiples hojas dentro de un mismo documento, separando configuraciones y datos experimentales por sesiones. De esta manera, se reduce la complejidad de navegación y organización de los datos.
- **Output Excel:** Este campo permite al usuario asignar un nombre personalizado al archivo de salida que almacenará los datos experimentales. Si no se introduce ningún nombre, el sistema asignará uno por defecto y lo almacenará en la carpeta predeterminada del proyecto */data*.

Una vez definidos estos parámetros en la interfaz, el sistema procederá a interpretar la hoja de cálculo seleccionada, extraerá los valores de configuración y ejecutará las rutinas correspondientes de forma autónoma.

En la siguiente sección se detalla cómo debe estructurarse el archivo Excel para asegurar una correcta ejecución del modo automático y evitar errores de lectura o interpretación.

4.4.3.1 Estructura del Excel

El archivo Excel que utiliza el programa como fuente de configuración en el modo automático debe seguir una estructura bien definida, tal como se muestra en la **Figura 22**. Esta estructura es fundamental para que el sistema pueda interpretar correctamente los parámetros necesarios para la ejecución del experimento.

													SMU1		SMU2	
step	tipus	ad_time	sv_time	temps	mfc3	mfc6	mfc9	mfc12	mfc16	mfc20	mfc24	flux total	v(V)	i(A)	v(V)	i(A)
1	CONFIG			no aplica	200	200	200	200	200	20	200	200		X	X	
		no aplica	no aplica		1000000	3000	50	60	1000000	3000	50					
					SA	CO2	H2S	PEPE	SA	CO2	H2S					
	MESURA	15s	10s	10s	100		100			20		20		10 mA	1-10/3 V	
	MESURA	15s	10s	10s						0		0		10 uA	5 V	
	MESURA	15s	10s	10s						10		10		10 mA	5 V	
	MESURA	15s	10s	10s						20		20		1 mA	2 mV	
	MESURA	15s	10s	30s						35		35		10 uA	1 uV	
	MESURA	15s	10s	30s						30		30		5 mA	5 V	
	MESURA	15s	10s	30s						20		20		10 nA	1-10/3 V	
	MESURA	15s	10s	30s								0		10 nA	5 mV	
	FINAL															

Figura 22: Estructura del Excel.

A continuación, se describen los distintos parámetros contenidos en el Excel, explicando su función dentro del sistema y los requisitos que debe cumplir su formato:

- **TIPUS**: Define el tipo de cada línea del archivo. Puede tomar tres valores:
    - **CONFIG**: Líneas de configuración general del experimento.
    - **MESURA**: Línea que define una medición concreta.
    - **FINAL**: Marca el final de las medidas definidas en el archivo.
- **AD_TIME**: Establece la frecuencia de adquisición de datos. Este campo ha sido eliminado del software, ya que el sistema ahora adquiere los datos en cuánto están disponibles, optimizando así el rendimiento.
- **SV_TIME**: Especifica el intervalo de tiempo con el que los datos se almacenan en una estructura interna, que posteriormente servirá para crear un archivo Excel de salida.
- **TEMPS**: Define el tiempo total de duración de la medición. Es un parámetro obligatorio en las líneas de tipo MESURA.
- **MFCX**: Representa los valores de flujo asignados a cada controlador. En las líneas CONFIG se indican los valores máximos permitidos para cada dispositivo.
- **FLUX TOTAL**: Establece el flujo máximo total que el sistema puede manejar. Si la suma de los flujos definidos en una línea de tipo MESURA supera este valor, la medición es cancelada automáticamente y el programa continúa con la siguiente configuración.
- **SMU1 v / SMU1 i / SMU2 v / SMU2 i**: Estos campos definen los valores de voltaje (v) e intensidad (i) que se aplicarán a los instrumentos Keithley 1 y 2. Su interpretación depende del modo de medida configurado.

4.4.3.2 Formato de los valores

Para que el sistema funcione correctamente en modo automático, es fundamental que los valores introducidos en el archivo Excel sigan una estructura coherente y precisa. Cada tipo de parámetro requiere un formato específico, y es importante respetarlo para evitar errores de interpretación que podrían comprometer la ejecución de las mediciones o procesos definidos.

En el caso de los parámetros de tiempo, como *TEMPS* o *SV_TIME*, estos deben escribirse siguiendo un orden fijo de unidades: semanas, días, horas, minutos y segundos. El formato establecido es: xw xd xh xm xs, donde cada componente indica una unidad temporal concreta (w = semanas, d = días, h = horas, m = minutos, s = segundos). Por ejemplo, un valor como 1d 30m representa una duración de un día y treinta minutos. Es importante mantener los espacios entre las unidades, ya que de otra forma provocaría errores al interpretar el valor.

En cuanto al voltaje, este debe especificarse siempre acompañado de su unidad, separada del número mediante un espacio. Las unidades admitidas son voltios (V), milivoltios (mV) y microvoltios (uV). Algunos ejemplos válidos serían 5 V o 200 mV.

De forma parecida, los valores de corriente deben introducirse con su unidad correspondiente y un espacio en medio. Las unidades aceptadas en este caso son amperios (A), miliamperios (mA) y microamperios (uA). Por ejemplo, tanto 100 uA como 1 mA serían expresiones válidas. Al igual que con el voltaje, un formato erróneo puede modificar completamente la configuración esperada.

En los modos de medición *AutomaticIV* o *AutomaticVI*, el formato que debe utilizarse es: *ValorInicial - ValorFinal / Saltos Unidad*. Este patrón indica desde qué valor se inicia, hasta cuál se llega, cuántos pasos hay entre medias y en qué unidad se expresa el parámetro. Por ejemplo, la expresión *0 - 5 / 6 V* describe una medida desde 0 hasta 5 voltios dividido en 6 incrementos. Este tipo de definición permite al sistema generar automáticamente los valores intermedios necesarios.

4.5 Gestión de errores

La robustez del sistema es esencial para garantizar que las mediciones se ejecuten correctamente y se minimicen las interrupciones inesperadas. Por ello, se han implementado diversas estrategias de gestión de errores, tanto a nivel de interfaz de usuario como durante la ejecución de los procesos experimentales.

4.5.1 Validación de configuración previa a la ejecución

Antes de iniciar cualquier medición, el sistema valida que todos los parámetros requeridos estén correctamente definidos. En caso de que el usuario omita algún parámetro necesario (como no seleccionar un dispositivo Keithley, introducir mal un tiempo, no seleccionar ningún MFC, etc.), se muestra un mensaje de error mediante una ventana emergente (pop-up), informando al usuario sobre el problema detectado.

Estas ventanas emergentes son mensajes temporales que explican de forma clara el error concreto, permitiendo al usuario corregirlo antes de intentar ejecutar el experimento nuevamente. Por ejemplo, si el usuario pulsa el botón *RUN MODE* sin haber configurado antes ningún modo de medición, el sistema muestra un mensaje indicando que debe configurarse al menos un modo para continuar **(Figura 23)**.

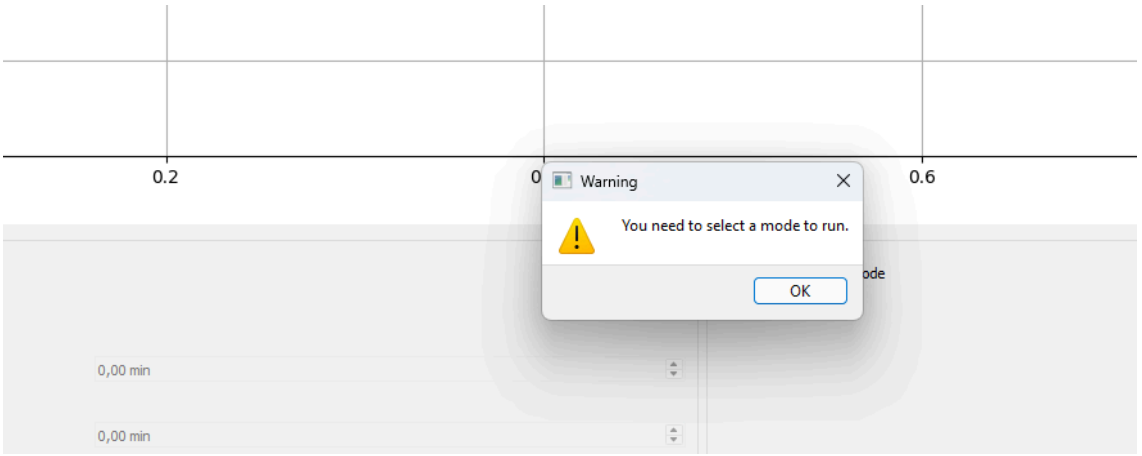

Figura 23: Error al no configurar ningún modo de medición.

Esta validación permite detectar de antemano la mayoría de errores típicos que podrían comprometer el flujo del experimento, evitando así tener que ejecutar el programa repetidamente para identificar y corregir dichos errores.

4.5.2 Gestión de errores en ejecución automática

Durante la ejecución de mediciones en modo automático, se han contemplado situaciones en las que el sistema pueda fallar debido a errores internos, errores en la lectura del Excel o fallos de conexión con los dispositivos.

En estos casos, se ha diseñado una gestión de errores que detecta automáticamente interrupciones inesperadas y:

- Detiene el proceso de medida de forma segura.
- Muestra al usuario un mensaje informativo sobre la causa (si puede determinarse).
- Guarda automáticamente en un archivo Excel los datos recogidos hasta el momento del fallo, asegurando que la información obtenida no se pierde y puede ser aprovechada para análisis posteriores.

En concreto, el sistema genera un archivo Excel que contiene todas las mediciones válidas realizadas hasta el momento de la interrupción. Este archivo se guarda automáticamente en la carpeta */data* con un nombre que incluye la fecha y la hora del fallo, lo que facilita su identificación.

Esta funcionalidad permite preservar los resultados parciales y evita repetir experimentos desde cero, lo que ahorra tiempo, reduce el desgaste de los dispositivos y mejora la eficiencia del proceso experimental.

Capítulo 5: Resultados

En esta sección se presenta un ejemplo práctico que demuestra el funcionamiento del sistema desarrollado, con el objetivo de mostrar el flujo completo del proceso: desde la configuración inicial de los dispositivos hasta la adquisición de datos y su representación gráfica en tiempo real. Para ello, hemos realizado una medición manual configurada mediante el modo *AutomaticIV*, el cual permite obtener la curva característica corriente-voltaje (I-V) del dispositivo bajo prueba.

La **Figura 24** muestra la configuración experimental utilizada. Hemos establecido un tiempo de medida de 0,50 minutos (30 segundos) en la que el MFC20 es el único que tendrá flujo, programado para suministrar un flujo constante de 20 mln/min. A nivel eléctrico, hemos aplicado una rampa de tensión que va desde 0 mV hasta 10 mV (0.001 V), dividida en 20 pasos equidistantes.

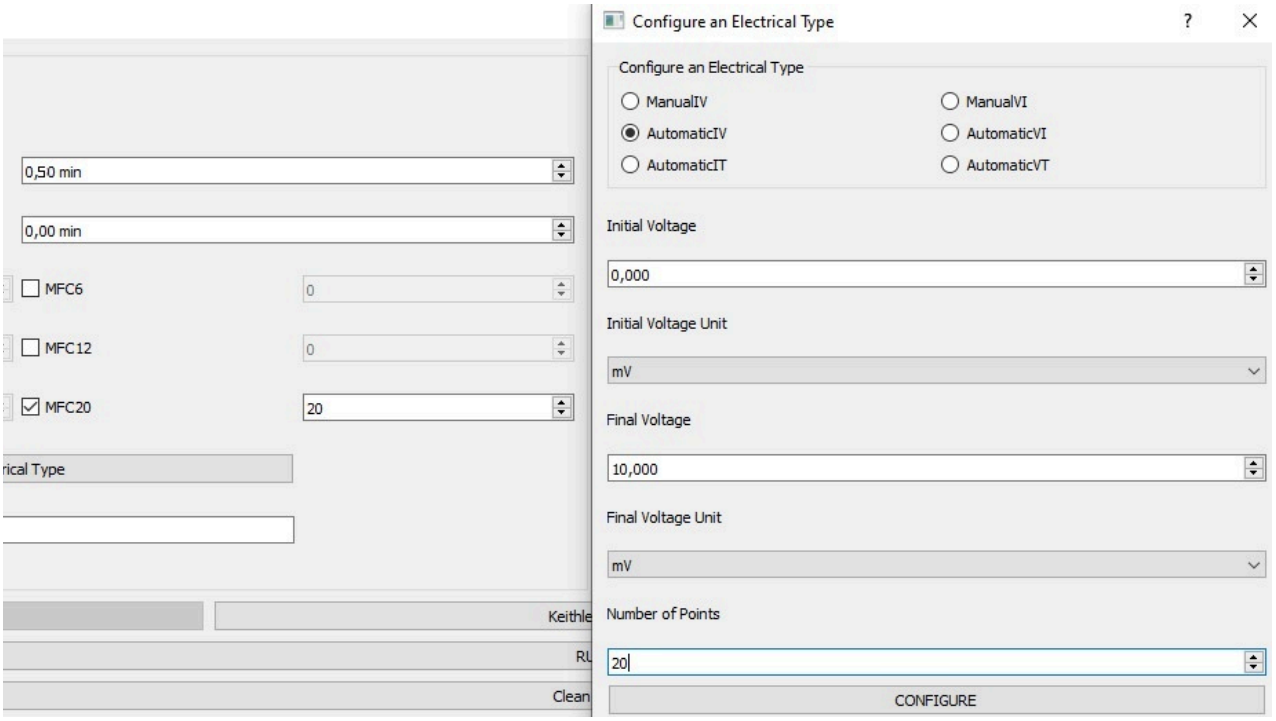

Figura 24: Configuración de la medida.

En la **Figura 25** se muestra la representación gráfica de los resultados obtenidos. La curva I-V refleja la evolución progresiva de la corriente a medida que aumenta el voltaje, demostrando así la respuesta eléctrica del componente bajo ensayo.

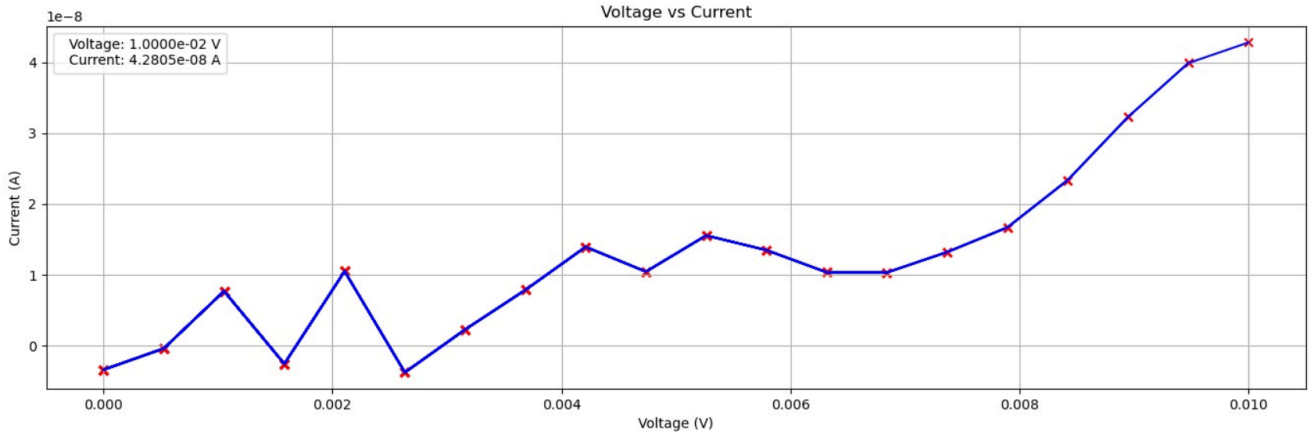

Figura 25: Representación gráfica de la curva I-V.

Por último, la **Figura 26** muestra el archivo Excel generado tras la ejecución del experimento. En este se almacenan de forma estructurada las variables adquiridas durante el proceso. Las lecturas de corriente se sitúan en el orden de los nanoamperios (10^{-9} A). Cabe destacar que el resto de Mass Flows permanecieron desactivados, como podemos observar en las columnas correspondientes, ya que sus valores son nulos.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	MFC3	MFC6	MFC9	MFC12	MFC16	MFC20	Temperature	Humidity	Time	Voltage_1	Voltage_Unit	Current_1	Current_Unit
2	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	20.031761169433594 mln/min	25	28	0	0.00052631578	V	-3.937067e-10	A
3	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.93636703491211 mln/min	25	27	1,5	0.00157894736	V	-2.571578e-09	A
4	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.918920516967773 mln/min	25	27	3	0.00263157894	V	-3.778814e-09	A
5	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.927831649780273 mln/min	25	26	4,5	0.00315789473	V	2.278807e-09	A
6	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.93827247619629 mln/min	25	26	6	0.00368421052	V	7.865243e-09	A
7	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.938934326171875 mln/min	25	26	7,5	0.00473684210	V	1.044819e-08	A
8	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.95102310180664 mln/min	25	26	9	0.00578947368	V	1.348261e-08	A
9	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.95159339904785 mln/min	25	28	10,5	0.00684210526	V	1.03333e-08	A
10	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.948593139648438 mln/min	25	28	12	0.00789473684	V	1.670187e-08	A
11	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.960338592529297 mln/min	25	30	13,5	0.00894736842	V	3.22855e-08	A
12	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.966127395629883 mln/min	25	31	15	0.01	V	4.280463e-08	A
13	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.964956283569336 mln/min	25	32	16,5	0.01	V	4.280463e-08	A
14	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.965511322021484 mln/min	25	33	18	0.01	V	4.280463e-08	A
15	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.975290298461914 mln/min	25	34	19,5	0.01	V	4.280463e-08	A
16	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.975950241088867 mln/min	25	34	21	0.01	V	4.280463e-08	A
17	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.97885513305664 mln/min	25	35	22,5	0.01	V	4.280463e-08	A
18	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.97336196899414 mln/min	25	35	24	0.01	V	4.280463e-08	A
19	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.97397804260254 mln/min	25	36	25,5	0.01	V	4.280463e-08	A
20	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.975799560546875 mln/min	25	37	27	0.01	V	4.280463e-08	A
21	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.974985122680664 mln/min	25	37	28,5	0.01	V	4.280463e-08	A
22	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	0.0 mln/min	19.974260330200195 mln/min	25	38	30	0.01	V	4.280463e-08	A

Figura 26: Excel generado con los datos adquiridos.

En esta segunda medición, se presenta un ejemplo del funcionamiento del sistema en un escenario de medida automática. El objetivo es demostrar la flexibilidad del sistema para adaptarse a diferentes configuraciones experimentales y necesidades. En este caso, se ha realizado una medida que hemos configurado mediante un fichero de configuración, controlando de forma precisa los flujos y parámetros eléctricos durante el experimento.

La **Figura 27** muestra el archivo de configuración utilizado para esta medida. En este se establece una secuencia de medidas, en las que se mantiene un flujo total constante de 200 mln/min.

												SMU1	
tipus	ad_time	sv_time	temps	mfc3	mfc6	mfc9	mfc12	mfc16	mfc20	mfc24	flux total	v(V)	i(A)
CONFIG			no aplica	200	200	200	200	200	20	200	200		X
	no aplica	no aplica		1000000	3000	50	60	1000000	3000	50			
				SA	SA	H2S	PEPE	SA	H2O VAPOR	H2S			
MESURA	15s	5s	20m	0	200	0			0		200		100 nA
MESURA	15s	5s	20m		140				60		200		100 nA
MESURA	15s	5s	20m		200				0		200		100 nA
MESURA	15s	5s	20m		120				80		200		100 nA
MESURA	15s	5s	20m		200				0		200		100 nA
FINAL													

Figura 27: Configuración de la medida utilizando Excel.

En la **Figura 28** se muestra la representación gráfica de los resultados obtenidos en tiempo real. La curva refleja claramente la evolución de la corriente medida en función del tiempo. Se observan distintos niveles escalonados de corriente que corresponden a los cambios de flujo realizados en cada etapa de la medida. La respuesta del dispositivo muestra cómo la variación en la composición del gas afecta su comportamiento eléctrico. Este tipo de análisis es fundamental para evaluar la sensibilidad y la selectividad del sensor.

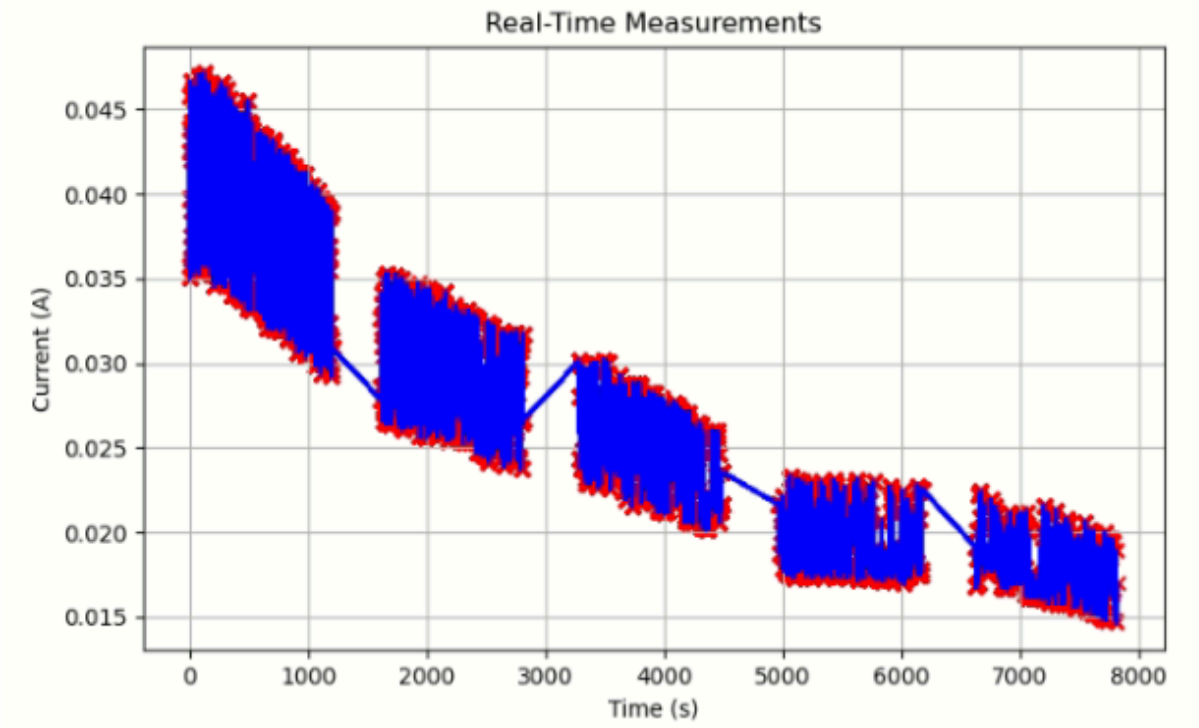

Figura 28: Representación gráfica de la medida.

Finalmente, se genera un archivo Excel que almacena de manera estructurada todos los datos adquiridos durante la prueba. Este archivo resulta muy útil para posteriores análisis y comparación de resultados.

Capítulo 6: Costes

En este proyecto, el coste principal ha sido el tiempo dedicado para llevar a cabo todas las tareas necesarias, desde la planificación inicial hasta la ejecución final. El trabajo ha requerido un esfuerzo constante para investigar, desarrollar y testear las soluciones, lo cual ha implicado muchas horas de dedicación personal.

Además del tiempo invertido, se han generado algunos gastos mínimos relacionados con la movilidad, principalmente el coste de la gasolina utilizada para desplazamientos necesarios durante el desarrollo del proyecto. Estos desplazamientos han sido puntuales y no representan un gasto significativo.

En resumen, el coste del proyecto se ha centrado fundamentalmente en el compromiso temporal y la dedicación personal.

Capítulo 7: Conclusiones

El desarrollo de este proyecto ha sido una oportunidad de gran aprendizaje. tanto a nivel técnico como personal. A lo largo del proceso, se ha implementado un sistema integral para la adquisición, gestión y análisis de datos eléctricos y de gases, integrando múltiples tecnologías, dispositivos físicos y protocolos de comunicación.

Uno de los aspectos más importantes ha sido la oportunidad de trabajar con una variedad de dispositivos y protocolos industriales. Esta integración me ha permitido adquirir experiencia práctica en la comunicación hardware-software y en el control de dispositivos físicos en un entorno experimental real.

Durante el desarrollo del software se ha prestado especial atención a la detección y prevención de errores en la transmisión de datos, un elemento crítico en sistemas donde la fiabilidad de la información es esencial. En este sentido, se han aplicado técnicas de validación, gestión de excepciones y verificación de integridad, con la finalidad de mejorar la robustez y estabilidad del sistema desarrollado.

Desde el punto de vista del diseño del software, la integración del patrón arquitectónico Modelo-Vista-Controlador (MVC) ha resultado beneficiosa. Este enfoque ha facilitado una clara separación de responsabilidades entre los distintos componentes del sistema, mejorando la organización del código y favoreciendo su escalabilidad y reutilización en futuros proyectos.

El uso de herramientas de depuración y pruebas ha sido clave para detectar y solucionar errores durante el desarrollo, asegurando el correcto funcionamiento del sistema final. Todo este proceso ha servido para reforzar la importancia de aplicar buenas prácticas a lo largo del ciclo de vida del software, como documentar bien el código, automatizar pruebas y diseñar pensando en futuras ampliaciones.

A nivel formativo, el proyecto ha supuesto un reto multidisciplinar que ha requerido la aplicación conjunta de conocimientos en programación, electrónica, automatización, análisis de datos y diseño de interfaces gráficas. Gracias a esta integración, he podido reforzar competencias clave en el ámbito de la ingeniería aplicada a la instrumentación y al análisis experimental.

En resumen, este Trabajo de Final de Grado se trata de un proyecto de alto valor práctico y formativo, que puede ampliarse fácilmente según nuevas necesidades. El sistema desarrollado ofrece una base sólida sobre la que construir futuras funcionalidades, integrar nuevos dispositivos o abordar proyectos más complejos en el ámbito de la monitorización, el control y el análisis de datos en entornos industriales o de investigación.

Capítulo 8: Bibliografía

1. *ESP32 Wi-Fi & Bluetooth SOC | Espressif Systems*. (s. f.). https://www.espressif.com/en/products/socs/esp32
2. *MQTT - The Standard for IoT Messaging*. (s. f.). https://mqtt.org/
3. *PIC32MX MIPS32®-Based M4K® Microcontrollers (MCUs)*. (s. f.). Microchip Technology. https://www.microchip.com/en-us/products/microcontrollers-and-microprocessors/32-bit-mcus/pic32-32-bit-mcus/pic32mx
4. Reynolds, M. (s. f.). *PIC18F Microcontroller Family Summary - Developer Help*. https://developerhelp.microchip.com/xwiki/bin/view/products/mcu-mpu/8bit-pic/pic18/
5. *MPLAB® X IDE*. (s. f.-b). Microchip Technology. https://www.microchip.com/en-us/tools-resources/develop/mplab-x-ide
6. R. Hernández-Gómez, D. Tuma, J.J. Segovia, & C.R. Chamorro. (2016). Experimental determination of (P, P, T) data for binary mixtures of methane and helium. *The Journal Of Chemical Thermodynamics*. https://www.sciencedirect.com/science/article/abs/pii/S0021961415004516
7. *U.S. Environmental Protection Agency | US EPA*. (2025, 15 mayo). US EPA. https://www.epa.gov/
8. Vaughan, A. (2025, 8 abril). Gas boilers now biggest source of air pollution in central London. *The Times*. https://www.thetimes.com/uk/environment/article/gas-boilers-air-pollution-central-london-c0n7bqrbz?region=global
9. *Sistemas de pruebas y medidas, una división de Emerson*. (2022, 2 septiembre). NI. https://www.ni.com/es.html
10. *Product*. (s. f.). https://www.ni.com/es-es/shop/product/labview.html?srsltid=AfmBOop47-5oCerykY4_O1BvoD4DzcL1YfOzFkum-RVpatUmTjRqZ7FI
11. Colaboradores de Wikipedia. (2025, 13 enero). *I2C*. Wikipedia, la Enciclopedia Libre. https://es.wikipedia.org/wiki/I%C2%B2C
12. *2450 SMU Instrument Datasheet*. (s. f.). Tektronix. https://www.tek.com/en/datasheet/smu-2400-graphical-sourcemeter/model-2450-touchscreen-source-measure-unit-smu-instrument
13. Colaboradores de Wikipedia. (2025b, mayo 14). *Ethernet*. Wikipedia, la Enciclopedia Libre. https://es.wikipedia.org/wiki/Ethernet

14. Rohde & Schwarz. (s. f.). *2. Introducir comandos SCPI*. https://www.rohde-schwarz.com/es/driver-pages/control-remoto/2-remote-programming-environments_231250.html

15. Bronkhorst USA LLC. (s. f.). *Our mass flow controllers*. Bronkhorst. https://www.bronkhorst.com/products/mass-flow-controller/

16. Bronkhorst USA LLC. (s. f.-a). *Bronkhorst, the low-flow specialists*. Bronkhorst. https://www.bronkhorst.com/

17. *Using FLOW-BUS with FlowSuite*. (2023, 17 octubre). Process Solutions Corp. https://psctexas.com/using-flowbus-with-flowsuite-fieldbus-communication-protocol/#:~:text=FLOW%2DBUS%20is%20an%20ethernet,FlowSuite%2C%20LabVIEW%2C%20Python

18. *Linkam THMS600 - Temperature Control Stage for Microscopy and Spectroscopy — Linkam Scientific*. (s. f.). Linkam Scientific. https://www.linkam.co.uk/thms600

19. Ag, S. (s. f.). *Sensirion - Smart Sensor Solutions*. sensirion.com. https://sensirion.com/products/catalog/SHT11

20. Colaboradores de Wikipedia. (2024, 13 septiembre). *Bus CAN*. Wikipedia, la Enciclopedia Libre. https://es.wikipedia.org/wiki/Bus_CAN

21. Electronic Team. (2024, 23 diciembre). *Última revisión protocolo RS232*. Serial Port Monitor. https://www.serial-port-monitor.org/es/articles/serial-communication/rs232-interface/

22. FTDI TTL-232R-5V. (s. f.). Mouser Electronics. https://www.mouser.es/ProductDetail/FTDI/TTL-232R-5V

23. *MVC - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN*. (2023, 13 noviembre). MDN Web Docs. https://developer.mozilla.org/es/docs/Glossary/MVC

24. Hernandez, R. D. (2021, 19 abril). *The Model View Controller Pattern – MVC Architecture and Frameworks Explained*. freeCodeCamp.org. https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/

25. Oloruntoba, S. (2024, 23 abril). *SOLID: The First 5 Principles of Object Oriented Design*. DigitalOcean. https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design

26. *Ejecución concurrente*. (s. f.). Python Documentation. https://docs.python.org/es/3.13/library/concurrency.html