



universidad  
de león



# Escuela de Ingenierías Industrial, Informática y Aeroespacial

## MÁSTER EN INGENIERÍA INDUSTRIAL

Trabajo de Fin de Máster

Sistema de adquisición de datos en redes submetering  
con comunicaciones Modbus/TCP

Data acquisition system in submetering networks with  
Modbus/TCP communications

Autor: Guzmán González Mateos  
Tutor: Antonio Morán Álvarez  
Cotutor: Miguel Ángel Prada Medrano

(Febrero, 2022)

**UNIVERSIDAD DE LEÓN**  
**Escuela de Ingenierías Industrial, Informática y**  
**Aeroespacial**

**MÁSTER EN INGENIERÍA INDUSTRIAL**  
**Trabajo de Fin de Máster**

**ALUMNO:** Guzmán González Mateos

**TUTORES:** Antonio Morán Álvarez, Miguel Ángel Prada Medrano

**TÍTULO:** Sistema de adquisición de datos en redes submetering con comunicaciones Modbus/TCP

**TITLE:** Data acquisition system in submetering networks with Modbus/TCP communications

**CONVOCATORIA:** Febrero, 2022

**RESUMEN:**

El sector industrial representa entre el 20% y el 40% del consumo energético de la sociedad, y se prevé un aumento de dicho consumo de cara al futuro. Este hecho supone la necesidad de implementar técnicas de control de consumo energético efectivas, las cuales dependen de la disponibilidad de datos masivos y detallados. En la actualidad existen diversas herramientas enfocadas en la gestión energética, pero que, sin embargo, presentan una serie de problemas que comprometen el dinamismo de las mismas, convirtiéndolas en herramientas ciertamente rígidas e ineficaces al tratarse de gestionar sistemas complejos, debido a la variedad de dispositivos existentes, con sus propias exigencias en lo que se refiere a generación de datos y monitorización.

En el presente proyecto se desarrollará una herramienta de adquisición de datos multipropósito, basada en el conocido lenguaje de programación Python, orientada tanto a la gestión energética como a la automatización de procesos, que pretende satisfacer así las necesidades de sistemas complejos de gestión. También se implementará una aplicación web desarrollada mediante el conocido *framework* Django, para simplificar la configuración y parametrización de la misma, ocultando la complejidad técnica asociada a la programación que la implementa. Los datos adquiridos serán volcados a una base de datos de Microsoft SQL Server.

Por último, se analiza el comportamiento de la herramienta mediante la realización de una serie de experimentos para evaluar las ventajas de la misma frente a las herramientas comerciales existentes.

**ABSTRACT:**

The industrial sector represents between 20% and 40% of society's energy consumption, and this consumption is expected to increase in the future. This fact implies the need to implement effective energy consumption control techniques that depend on the availability of massive and detailed data. Currently there are several tools focused on energy management, but nevertheless, they present a number of problems that compromise their dynamism, making them certainly rigid and ineffective tools when it comes to managing complex systems, due to the variety of existing devices, with their own requirements in terms of data generation and monitoring.

This project will develop a multipurpose data acquisition tool, based on the well-known Python programming language, oriented both to energy management and process automation, which aims to meet the needs of complex management systems. A web application developed using the well-known Django framework will also be

implemented, in order to simplify its configuration and parameterization, hiding the technical complexity associated with the programming that implements it. The acquired data will be stored into a Microsoft SQL Server database. Finally, the behavior of the tool is analyzed by performing a series of experiments to evaluate its advantages over existing commercial tools.

**Palabras clave:** *Submetering*, Python, MODBUS/TCP, Adquisición de datos, Base de datos

**Firma del alumno:**

**VºBº Tutor/es:**

# INDICE

<b>1</b>	<b>INTRODUCCIÓN .....</b>	<b>9</b>
<b>2</b>	<b>ESTADO DEL ARTE .....</b>	<b>11</b>
2.1	SISTEMA ELÉCTRICO TRADICIONAL .....	11
2.2	SMART GRIDS .....	13
2.3	REDES DE SUBMETERING.....	15
2.4	BREVE REPASO SOBRE HERRAMIENTAS COMERCIALES DE GESTIÓN ENERGÉTICA .....	16
2.5	EL PROTOCOLO DE COMUNICACIÓN MODBUS.....	19
2.5.1	<i>Introducción al protocolo.....</i>	<i>19</i>
2.5.2	<i>Modelo de datos MODBUS.....</i>	<i>20</i>
2.5.3	<i>Direccionamiento de los datos MODBUS.....</i>	<i>21</i>
2.5.4	<i>Principales funciones predefinidas .....</i>	<i>21</i>
2.5.5	<i>Endianness.....</i>	<i>22</i>
2.6	CONCEPTOS INFORMÁTICOS FUNDAMENTALES EN LA COMPRESIÓN DE LA IMPLEMENTACIÓN .....	23
2.7	EL LENGUAJE DE PROGRAMACIÓN PYTHON.....	25
2.7.1	<i>Principales frameworks para el desarrollo web.....</i>	<i>26</i>
2.7.2	<i>Librerías destacadas que se han utilizado .....</i>	<i>27</i>
<b>3</b>	<b>TRABAJO DESARROLLADO .....</b>	<b>28</b>
3.1	ESPECIFICACIONES TÉCNICAS Y DISEÑO.....	28
3.2	PLANIFICACIÓN DEL TRABAJO DESARROLLADO .....	30
3.3	DESCRIPCIÓN DE LA ARQUITECTURA CONCEPTUAL E IMPLEMENTACIÓN.....	31
3.3.1	<i>Sistema multihilo de adquisición de datos.....</i>	<i>32</i>
3.3.1.1	<i>Comunicación con los equipos .....</i>	<i>32</i>
3.3.1.2	<i>Volcado de datos a la base de datos .....</i>	<i>36</i>
3.3.1.3	<i>Adquisición de datos de forma determinista .....</i>	<i>37</i>
3.3.2	<i>Aplicación web.....</i>	<i>38</i>
3.3.2.1	<i>Funcionamiento del Backend .....</i>	<i>38</i>
3.3.2.2	<i>Descripción del Frontend .....</i>	<i>39</i>
3.3.3	<i>Interfaces de comunicación desarrolladas .....</i>	<i>47</i>
<b>4</b>	<b>EXPERIMENTOS Y RESULTADOS.....</b>	<b>50</b>
4.1	PRESENTACIÓN DEL SISTEMA FÍSICO .....	50
4.2	EXPERIMENTO 1: CAPTURA MASIVA DE DATOS A ELEVADA FRECUENCIA.....	55
4.2.1	<i>Representación gráfica de los datos del PLC .....</i>	<i>55</i>
4.2.2	<i>Representación gráfica de los datos del PM8000.....</i>	<i>56</i>
4.2.3	<i>Representación gráfica de los datos de la PowerTag de cabecera.....</i>	<i>57</i>
4.2.4	<i>Representación gráfica de los datos del PM5000.....</i>	<i>58</i>
4.2.5	<i>Representación gráfica de los datos de las PowerTags del motor, entrada230V y alimentación de 24Vcc.....</i>	<i>60</i>
4.3	EXPERIMENTO 2: COMPARATIVA ENTRE EL COMPORTAMIENTO DE LA HERRAMIENTA Y EL COMPORTAMIENTO DEL SOFTWARE POWER MONITORING EXPERT .....	63
4.3.1	<i>Representación gráfica de los datos del PLC .....</i>	<i>63</i>
4.3.2	<i>Representación gráfica de los datos del PM5000.....</i>	<i>64</i>
4.3.3	<i>Representación gráfica de los datos de las PowerTags del motor y de la entrada de 230V .....</i>	<i>65</i>
4.4	DISCUSIÓN DE RESULTADOS.....	67
<b>5</b>	<b>CONCLUSIONES.....</b>	<b>68</b>
<b>6</b>	<b>LÍNEAS FUTURAS.....</b>	<b>69</b>

<b>7</b>	<b>BIBLIOGRAFÍA.....</b>	<b>70</b>
----------	--------------------------	-----------

## ***INDICE DE TABLAS***

Tabla 2.1 Prefijos asignados a los rangos de datos MODBUS [20].....	21
---	----

## INDICE DE FIGURAS

Ilustración 2.1 Sistema eléctrico tradicional[11].....	11
Ilustración 2.2 Concepto de Smart Grid [11].....	13
Ilustración 2.3 Diferencia entre submetering y bulk metering [27].....	15
Ilustración 2.4 Trama de datos típica del protocolo MODBUS, junto con la trama específica de MODBUS TCP/IP [18].....	19
Ilustración 2.5 Solicitud y respuesta ante una solicitud exitosa y ante error [18].....	20
Ilustración 2.7 Esquema de una ejecución multihilo en Python [33].....	26
Ilustración 3.1 Cronograma del trabajo desarrollado .....	30
Ilustración 3.2 Arquitectura conceptual de la herramienta.....	31
Ilustración 3.3 Diagrama de clases de los módulos desarrollados.....	35
Ilustración 3.4 Respuesta de Django ante una solicitud [21].....	38
Ilustración 3.5 Estructura básica de la web desarrollada.....	40
Ilustración 3.6 Página principal de la aplicación web.....	41
Ilustración 3.7 Página básica de un equipo .....	42
Ilustración 3.8 Página de configuración del equipo .....	43
Ilustración 3.9 Página estándar de parametrización de los registros .....	44
Ilustración 3.10 Página de registros correspondiente a los PLCs.....	44
Ilustración 3.11 Estructura particular para dar soporte a pasarelas y Power Tags .....	45
Ilustración 3.12 Página específica de configuración de las pasarelas.....	46
Ilustración 3.13 Página de adicción y acceso a las Power Tags.....	46
Ilustración 3.14 Página de configuración de las Power Tags y parametrización de registros .....	47
Ilustración 3.15 Funcionamiento de las interfaces de comunicación entre recursos.....	49
Ilustración 4.1 Maqueta de 4 variables empleada en la experimentación[42] .....	50
Ilustración 4.2 Válvula de 3 vías existente en la maqueta de 4 variables .....	51
Ilustración 4.3 Electroválvula entre tanques de proceso.....	51
Ilustración 4.4 Bomba de recirculación.....	52
Ilustración 4.5 Esquema eléctrico completo .....	53
Ilustración 4.6 Esquema eléctrico simplificado.....	54
Ilustración 4.7 Datos recabados del PLC .....	55
Ilustración 4.8 Datos recabados del PM8000.....	56
Ilustración 4.9 Datos recabados de la PowerTag de cabecera.....	57
Ilustración 4.10 Datos recabados del PM5000 .....	58
Ilustración 4.11 Comparativa entre la lectura del neutro del PM8000 y PM5000 .....	59
Ilustración 4.12 Datos recabados de las PowerTags motor, entrada230V y alimentacion 24Vcc... 61	
Ilustración 4.13 Gráficos filtrados de las PowerTags en cuestión.....	62
Ilustración 4.14 Datos recabados del PLC .....	63
Ilustración 4.15 Datos recabados del PM5000 .....	64
Ilustración 4.16 Datos recabados de las PowerTags del motor y de la entrada de 230V .....	65
Ilustración 4.17 Datos filtrados de la PowerTag de entrada 230V.....	66

## ***GLOSARIO DE ABREVIATURAS Y TÉRMINOS***

**TCP** - Protocolo de Control de Transmisión  
**PLC** - Controlador Lógico Programable  
**ADU** – Unidad de Datos de Aplicación  
**PDU** - Unidad de Datos del Protocolo  
**HMI** - Interfaz Hombre Máquina  
**MVT** – Modelo Vista Plantilla  
**PME** – Power Monitoring Expert  
**SCADA** – Supervisory Control and Data Adquisition  
**GIL** – Global Interpreter Locker  
**API** – Interfaz de Programación de Aplicaciones  
**REST** – Transferencia de Estado Representacional





# 1 Introducción

Actualmente, el sector industrial representa entre el 20% y el 40% del consumo energético de la sociedad [1], lo cual, añadido al actual problema derivado del encarecimiento de los combustibles fósiles, la lucha contra el cambio climático mediante acciones como la imposición de tasas sobre las emisiones de carbono, y la perspectiva de un futuro cada vez más dependiente de la energía eléctrica (entre otros motivos, como consecuencia de la previsible expansión de los vehículos eléctricos [2]), supone la popularización de la práctica del *Submetering*, con el objetivo de lograr una mayor trazabilidad de un hipotético problema, que permita identificar consumos locales anómalos [3], [4], aplicando las técnicas y acciones de control necesarias. Dichas acciones dependen de la disponibilidad de datos masivos y detallados, accesibles en tiempo real y mediante históricos, con el objetivo de facilitar la toma de decisiones mediante su interpretación.

En la actualidad existen diversas herramientas enfocadas en la gestión energética [5]–[8], pero que, sin embargo, presentan una serie de problemas, debido a: su bajo dinamismo a la hora de soportar tecnologías multifabricante; soportar tecnologías multipropósito (por ejemplo, orientadas a la gestión energética y a la automatización de procesos industriales); hacer un uso racional y optimizado de los recursos disponibles para el almacenamiento y manipulación de datos; y, por último y como principal problema identificado que motiva la realización del presente proyecto, las herramientas comerciales enfocadas en la gestión energética se muestran ineficaces a la hora de abordar actividades con tiempos de respuesta muy ajustados, debido a la rigidez que presentan a la hora de realizar capturas de datos cíclicas en pequeños intervalos de tiempo.

Por todo lo anteriormente descrito, en el presente proyecto se desarrollará una herramienta de adquisición de datos multipropósito, orientada tanto a la gestión energética como a la automatización de procesos, cuyo principal objetivo será ofrecer una parametrización del intervalo de captura de datos de cada equipo particular, cubriendo las numerosas y variadas necesidades temporales existentes dentro de un sistema complejo. De acuerdo con la norma UNE-EN ISO 50001:20218 [9], dicha herramienta estará dotada de una aplicación web para permitir su operación por parte del usuario, de forma simple e intuitiva, abstrayéndolo de la complejidad técnica de las herramientas y tecnologías utilizadas para dar soporte a la misma, y facilitando la operación de la herramienta.

Por último, se llevarán a cabo una serie de experimentos para probar el comportamiento de la herramienta, haciendo uso de una de las maquetas de 4 variables de las que dispone el Grupo de Investigación SUPPRESS, las cuales se tratan de células de ensayo de estrategias de control dotadas de instrumentación industrial inteligente para medida y control de caudal, nivel, presión y temperatura.



Se concluye el presente documento con una serie de conclusiones extraídas de los mencionados experimentos, así como una serie de indicaciones para el desarrollo futuro y crecimiento de la herramienta en cuestión.



## 2 Estado del arte

### 2.1 Sistema eléctrico tradicional

En la actualidad, buena parte del sistema eléctrico se estructura en torno a una topología tradicional y bien definida, donde existen una serie de entidades claramente diferenciadas y encargadas de las diferentes actividades que se suceden dentro del sector eléctrico, como son [10]:

- Generación
- Transporte
- Distribución
- Comercialización
- Consumo

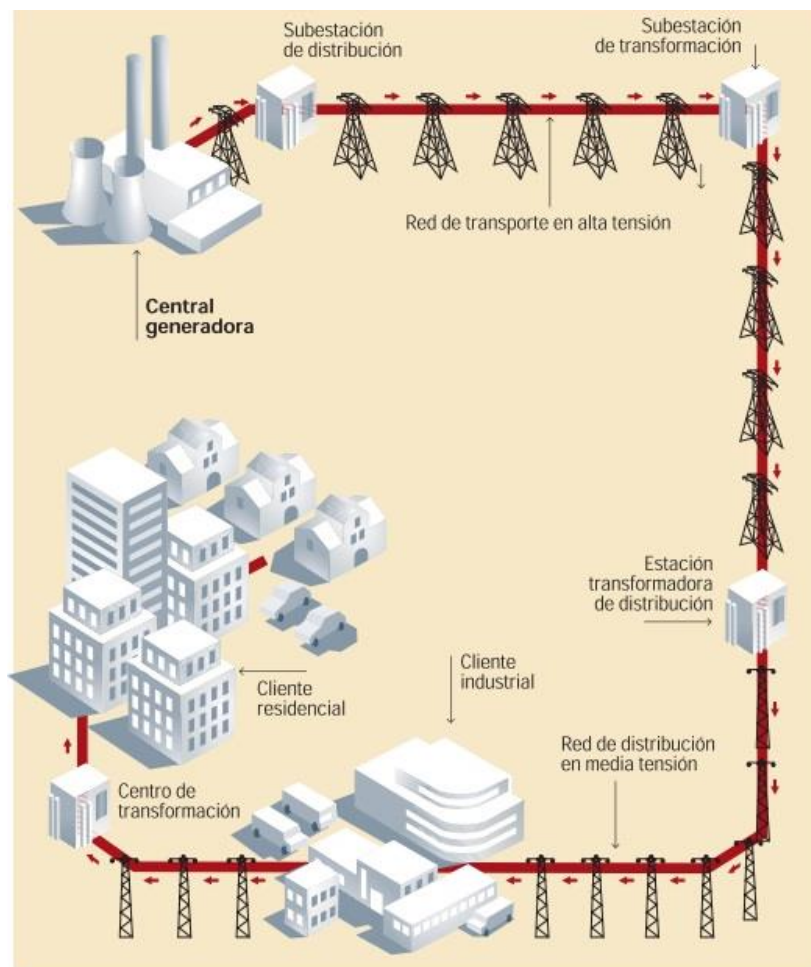


Ilustración 2.1 Sistema eléctrico tradicional[11]



Este sistema tradicional tiene sentido en un panorama donde se dan una serie de circunstancias muy marcadas:

- La variación de la demanda energética es fácilmente predecible, dado su carácter cíclico anual, pudiendo establecerse una previsión de la demanda suficientemente fiable en base a históricos y condiciones climatológicas actuales.
- La superposición entre actividades del sector eléctrico es mínima, lo que permite identificar claramente los nodos extremos del sistema (generadores y consumidores), simplificándose las estrategias de transporte y distribución.
- Los principales consumidores del sistema eléctrico siguen unos patrones de demanda fácilmente identificables, dado que se tratan de industrias con hábitos de consumo claramente marcados.
- La generación centralizada tiene por objetivo facilitar la explotación de fuentes de energía primarias, lo que condiciona fuertemente la estrategia de transporte a seguir.

No obstante, en la actualidad, dichas circunstancias comienzan a dejar de representar la realidad del sector:

- La variación de la demanda resulta cada vez menos predecible, debido, por un lado, a cambios climatológicos que dificultan la previsión de la influencia de las estacionalidades en la demanda, mientras que la mayor preocupación se centra en la entrada del vehículo eléctrico en el mercado, el cual se prevé que ocasione fuertes desviaciones en la demanda entre horas punta y horas valle, las cuales el sistema eléctrico actual no es capaz de asimilar de manera efectiva, y además, dada la naturaleza móvil del vehículo eléctrico, así como la dependencia de la sociedad actual por el transporte privado, el sistema eléctrico actual se encuentra en serios problemas para garantizar el suministro.
- La popularización de la cogeneración lleva al surgimiento de figuras que oscilan entre un rol de generador y de consumidor. Esto sumado al abaratamiento de las tecnologías de generación eléctrica a pequeña escala (también llamada microgeneración), dan lugar al concepto de generación distribuida, lo cual supone un reto para la actividad de distribución.
- La cogeneración implantada en grandes industrias u otras entidades que demanden una gran cantidad de energía térmica, supone la variación de la demanda de dichas entidades respecto a su comportamiento tradicional, reduciendo su papel modulador del consumo eléctrico y suponiendo un nuevo reto para las actividades de distribución y transporte.

Actualmente, las redes convencionales poseen sistemas de comunicaciones unidireccionales entre los consumidores y los centros de control, gracias a la incorporación de medidores digitales, lo cual permite la gestión y monitorización de datos masivos que facilitan la toma de decisiones. No obstante, con esto no basta para afrontar los retos anteriormente mencionados, sino que se hace necesaria una reconcepción de la propia red eléctrica [2], [11].



## 2.2 Smart Grids

Según Red Eléctrica de España, una red inteligente (o *Smart Grid*) es “aquella que puede integrar de forma eficiente el comportamiento y las acciones de todos los usuarios conectados a ella, de tal forma que se asegure un sistema energético sostenible y eficiente, con bajas pérdidas y altos niveles de calidad y seguridad de suministro” [12].

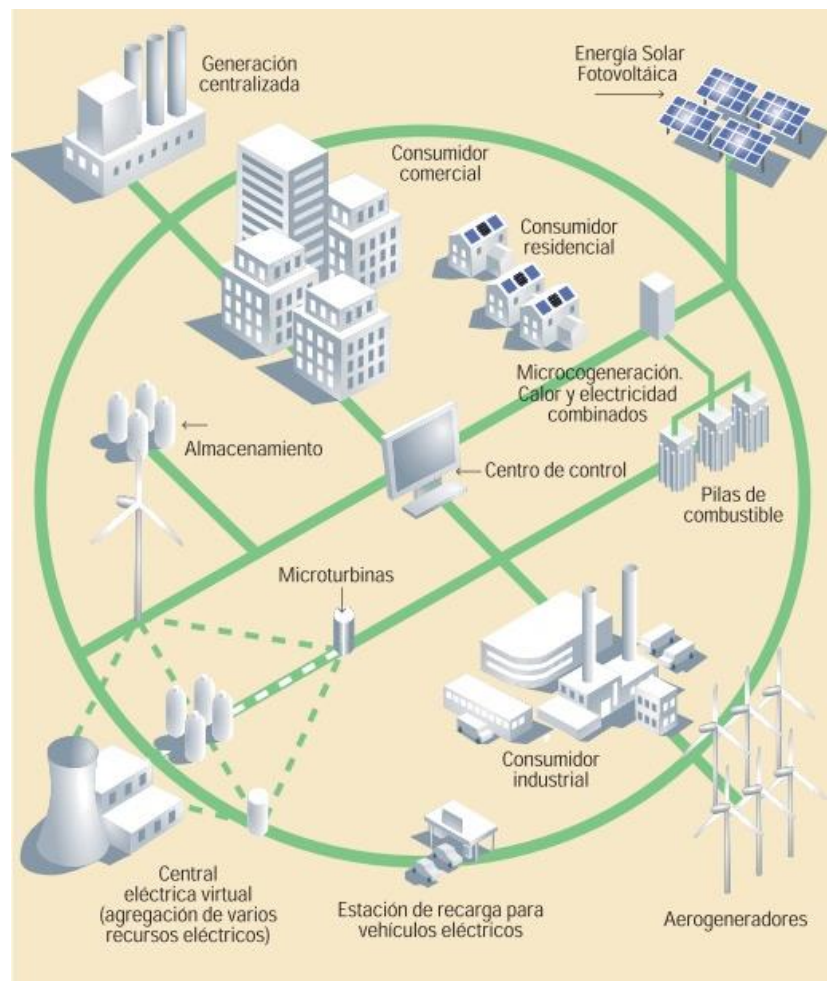


Ilustración 2.2 Concepto de Smart Grid [11]

La monitorización en tiempo real de la que goza el sistema eléctrico actual permite garantizar la calidad y seguridad del suministro mediante la identificación temprana y el mantenimiento predictivo de fallas, no obstante, este grado de automatización no permite el nivel de integración de los usuarios de la red que se necesita de cara al futuro inmediato. La integración mencionada está muy relacionada con la comunicación bidireccional entre generadores y consumidores, independientemente de su tamaño, lo que da lugar a un mayor grado de control y automatización de la propia red. La existencia de generación distribuida implica la necesidad de controlar flujos de potencia en tiempo real, con el objetivo de hacer un uso eficiente de los componentes de la red [11], [13].

En general, las principales características que ha de reunir una red inteligente serían:



- Autonomía: las redes inteligentes funcionan como nodos que, en un momento dado, pueden operar de forma aislada al resto del sistema, por lo tanto, contemplan entidades que realizan las actividades de generación, consumo, y distribución, necesarias en un sistema eléctrico mínimo. Se ha de simplificar el acceso al sistema eléctrico tanto como consumidor y como generador, eliminando trabas técnicas para garantizar la viabilidad económica.
- Autodiagnóstico: el diagnóstico y la identificación de fallas dejan de considerarse actividades centralizadas y pasan a tener un carácter distribuido, siendo una red inteligente capaz de monitorizar su propia actividad.
- Transparencia con el consumidor: la información generada en la red está a disposición del consumidor, quien debe ser capaz de modificar sus hábitos de consumo. El objetivo principal es lograr una estabilización de la demanda mediante beneficios al consumidor que se acoge a ciertos hábitos de consumo beneficiosos para el correcto funcionamiento de la red inteligente, huyendo de crestas y valles en la demanda, que fuerzan la necesidad de sobredimensiones ineficientes.
- Ciberseguridad: Al dotar de conectividad a los equipos de la red, surgen problemas de vulnerabilidad, que en una infraestructura crítica como es la red eléctrica tiene aún mayor trascendencia. Se debe dotar a la infraestructura de los niveles de seguridad necesarios para evitar accesos no autorizados y ataques de otro tipo, como puede ser la alteración de la información.
- Calidad del suministro: Dada la dependencia de la sociedad por la energía eléctrica, así como la existencia de cada vez más dispositivos electrónicos vulnerables a perturbaciones de red, las redes inteligentes deben garantizar una calidad de suministro referida tanto a la estabilidad y continuidad del mismo como a la eliminación de dichas perturbaciones de la red, mediante dispositivos de electrónica de potencia con este propósito.

Teniendo en cuenta las características mencionadas en las redes inteligentes, se puede identificar una fuerte dependencia de las mismas por una infraestructura organizada en torno a la generación, almacenamiento y procesamiento de datos masivos y detallados [4].



## 2.3 Redes de submetering

El *submetering* hace referencia a la medición parcial de unas determinadas magnitudes de interés. En el presente documento, cuando se habla de *submetering* se hace referencia implícita al *submetering* energético, el cual se enfoca en la medición de magnitudes eléctricas [14].

La razón principal de la implementación de redes de *submetering* radica en la necesidad de obtener información detallada en relación con los consumos energéticos, con el objetivo de identificar patrones de consumo, elementos susceptibles de introducir perturbaciones y comprometer la calidad del suministro, identificar dispositivos cuyo consumo represente un porcentaje importante respecto del consumo general de una determinada área o edificio, y, en definitiva, disponer de información en tiempo real, detallada y desglosada de los diferentes consumos que se están sucediendo en una determinada instalación [1], [14], [15].

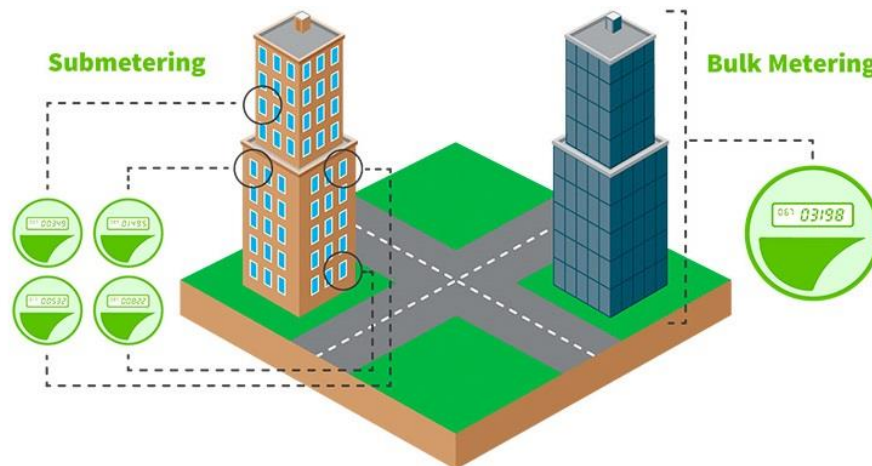


Ilustración 2.3 Diferencia entre submetering y bulk metering [27]

Tradicionalmente se ha realizado la medición únicamente del consumo energético general de una determinada entidad (comúnmente conocido como *Bulk metering*, en la literatura anglosajona). Este tipo de mediciones enmascaran problemas de consumo locales, no permitiendo la aplicación de estrategias de control de consumo.



## 2.4 Breve repaso sobre herramientas comerciales de gestión energética

Debido a la trayectoria de Schneider Electric en torno a la industria eléctrica, así como su enfoque actual en la transformación digital orientada a la gestión energética, la automatización de edificios, industrias e infraestructuras, la mayoría de herramientas de este tipo son suministradas por esta compañía. En la actualidad, existen una serie de paquetes y soluciones software enfocadas en la adquisición y procesamiento de datos masivos, destacando principalmente:

- **Power SCADA Operation:** se trata de una solución de supervisión y control en tiempo real, enfocado en la supervisión de grandes instalaciones e infraestructuras críticas. Las principales funcionalidades de esta herramienta son [7], [16], [17]:
  - 1) La supervisión de sistemas complejos, donde se integra gestión energética con supervisión y control industrial.
  - 2) El registro de eventos, tendencias y valores históricos.
  - 3) La configuración de una serie de alarmas típicas en herramientas SCADA.
- **Power Monitoring Expert:** a diferencia de Power SCADA Operation, se trata de un paquete completo de software orientado a la gestión energética para entornos industriales, comerciales, de grandes instalaciones, etc. Consiste en una aplicación de software cliente-servidor, que recopila datos de monitorización de energía a través de una red de dispositivos conectados. Los datos de monitorización de la energía se procesan y almacenan utilizando Microsoft SQL Server y los usuarios pueden acceder a ellos en una variedad de formatos a través de diferentes interfaces de usuario, utilizándose comúnmente interfaces web intuitivas para facilitar la presentación de la información recabada y procesada [8], [16], [17].

Las principales funcionalidades de esta herramienta son:

  - 1) Supervisión en tiempo real a través de interfaces web multiusuario.
  - 2) Posibilidad de representación gráfica de tendencias.
  - 3) Análisis de la calidad del suministro eléctrico.
  - 4) Configuración de notificaciones asociadas a alarmas del usuario.
- **SIMATIC Energy Manager Pro:** se trata de una herramienta del fabricante SIEMENS, orientada a la monitorización de consumos energéticos industriales, con el objetivo de establecer una relación entre los equipos industriales utilizados y los consumos producidos, identificando a los mayores consumidores. Contempla especialmente aspectos de ciberseguridad asociados con la protección de los datos generados, así como controles de accesos a los equipos industriales que monitoriza [5], [16].
- **DESIGO:** como el anterior, es una herramienta del fabricante SIEMENS, pero en este caso orientada a la automatización y control de edificios, por lo que su utilización se encuentra centrada en la domótica, soportando tecnologías de comunicación propias de este campo como es BACnet, aunque también soporta





protocolos como MODBUS, MODBUS/TCP y OPC para la integración de equipos de diferentes fabricantes [6], [16].

De las anteriores, la herramienta más completa es el Power Monitoring Expert, pero presenta una serie de problemas asociados:

- Si bien no se encuentra limitada la utilización exclusiva de tecnología propietaria, la incorporación de equipos desarrollados mediante tecnología de fabricantes terceros es complicada, lo cual es un problema serio teniendo en cuenta el entorno heterogéneo que es común encontrar en grandes industrias, edificios o infraestructuras.
- Las bases de datos que genera se caracterizan por almacenar una gran cantidad de información, en muchos casos, innecesaria. La definición de la información que se desea volcar en la base de datos, diferenciándola de la que no se desea recabar, resulta complicada, lo cual es un problema al trabajar con numerosos equipos volcando más información de la estrictamente deseada, ya que aumentará innecesariamente el tamaño de las bases de datos, haciendo ineficiente su consulta, tratamiento de los datos consultados, almacenamiento...
- Ocasionalmente, podría haber un interés en recabar datos provenientes de fuentes diferentes a los equipos típicos que se encuentran en una red de *submetering*, como por ejemplo datos relacionados con equipos de control (esto es, variables asociadas a una estrategia de control y almacenadas en memoria de un autómatas). Este tipo de equipos no están soportados por el conjunto de herramientas de Power Monitoring Expert. Sin embargo, Power SCADA sí ofrece soporte para los mismos, aunque limitado únicamente a la tecnología propietaria de Schneider Electric.
- El problema más crítico de Power Monitoring Expert está en la imposibilidad para parametrizar el período de captura de datos. Si bien una de sus características es la supervisión en tiempo real, en la práctica dicha supervisión no se produce en tiempo real, puesto que la captura y almacenamiento de nuevos datos por parte del sistema se produce en intervalos de 1 minuto. Esto resulta un impedimento serio si se deseara llevar a cabo acciones como, por ejemplo, hacer uso de los datos recabados para monitorizar el sistema realmente en tiempo real, pues el retardo introducido por la herramienta de adquisición de datos es demasiado elevado para hacer esa consideración. Tampoco sería posible reutilizar estos datos almacenados para otros propósitos exigentes en cuanto a la velocidad de su respuesta se refiere, por el motivo ya mencionado.

Los problemas mencionados son en realidad problemas habituales en el común de las herramientas de este tipo, los cuales pueden no suponer mayor complicación para la mayoría de las aplicaciones, pero en ciertos casos vuelven a estas herramientas poco flexibles, así como desaconsejables candidatas para ser utilizadas en la experimentación dentro de un entorno heterogéneo de dispositivos.



En lo que se refiere a los sistemas de gestión energética, se ha de tener en especial consideración la norma UNE-EN ISO 50001:2018 [9], la cual establece los requisitos de orientación para el uso de los mismos, haciendo una definición de las características que deben guardar las herramientas y elementos utilizados dentro de dichos sistemas de gestión, en función del perfil del personal encargado de la operación de cada una de estas herramientas.



## 2.5 El protocolo de comunicación MODBUS

### 2.5.1 Introducción al protocolo

MODBUS es un protocolo de mensajería, situado en el nivel 7 del modelo OSI (capa de aplicación), que permite la comunicación cliente/servidor entre dispositivos conectados en diferentes tipos de buses o redes. Todo tipo de dispositivos (PLCs, HMIs, paneles de control, variadores de frecuencia, controles de movimiento, dispositivos de E/S, medidores...) pueden utilizar el protocolo MODBUS para iniciar una operación remota. Actualmente se implementa utilizando:

- TCP/IP sobre Ethernet (los equipos utilizan por defecto el puerto 502 como puerto de escucha de las solicitudes y respuestas recibidas).
- Transmisión asíncrona en serie a través de diversos medios (cable: EIA/TIA-232-E, EIA-422, EIA/TIA-485-A; fibra, radio, etc.).
- MODBUS PLUS, una red de alta velocidad.

MODBUS define una unidad de datos de protocolo (PDU) sencilla e independiente de las capas de comunicación que lo sustentan. Las particularidades de la comunicación específica que se utilice para implementar el protocolo pueden introducir algunos campos adicionales en esta unidad de datos, en forma de cabeceras o terminaciones, dando origen a la llamada unidad de datos de aplicación (ADU) [18]. La ADU es construida por el cliente que inicia una transacción MODBUS. En el caso de MODBUS TCP/IP, esta ADU es encapsulada mediante la adición de una cabecera de trama específica [19].

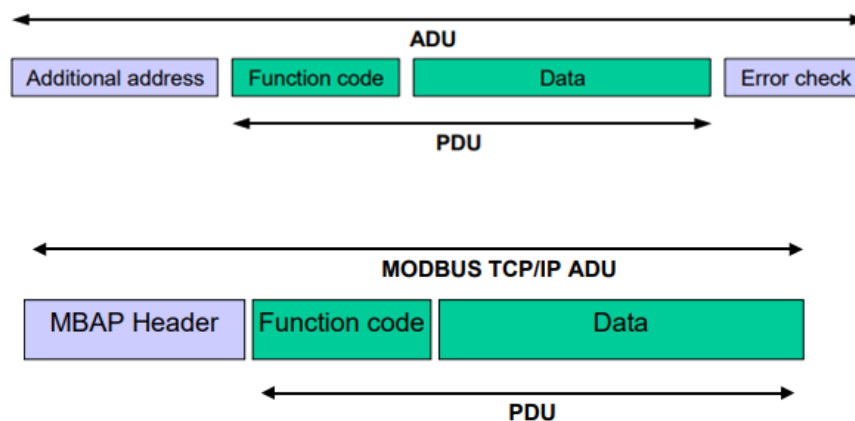


Ilustración 2.4 Trama de datos típica del protocolo MODBUS, junto con la trama específica de MODBUS TCP/IP [18]

Cuando se envía un mensaje desde un cliente a un dispositivo servidor, el campo de código de función indica al servidor qué tipo de acción debe realizar. Este campo es codificado en un byte (1...255, aunque el rango de valores 128...255 se corresponde con valores reservados para responder a excepciones).

El campo de datos de los mensajes enviados desde un cliente a los dispositivos del servidor contiene información adicional que el servidor necesita para llevar a cabo las acciones



especificadas mediante el código de función. En determinadas ocasiones puede no ser necesario especificar ningún dato adicional, porque la función que se desee llamar no los precise. Cuando se produce la respuesta por parte del servidor, el campo de datos contendrá la información solicitada por el cliente, a no ser que la respuesta recibida se deba a algún tipo de error, en cuyo caso dicho campo contendrá el código identificativo de la excepción que ha tenido lugar [18].

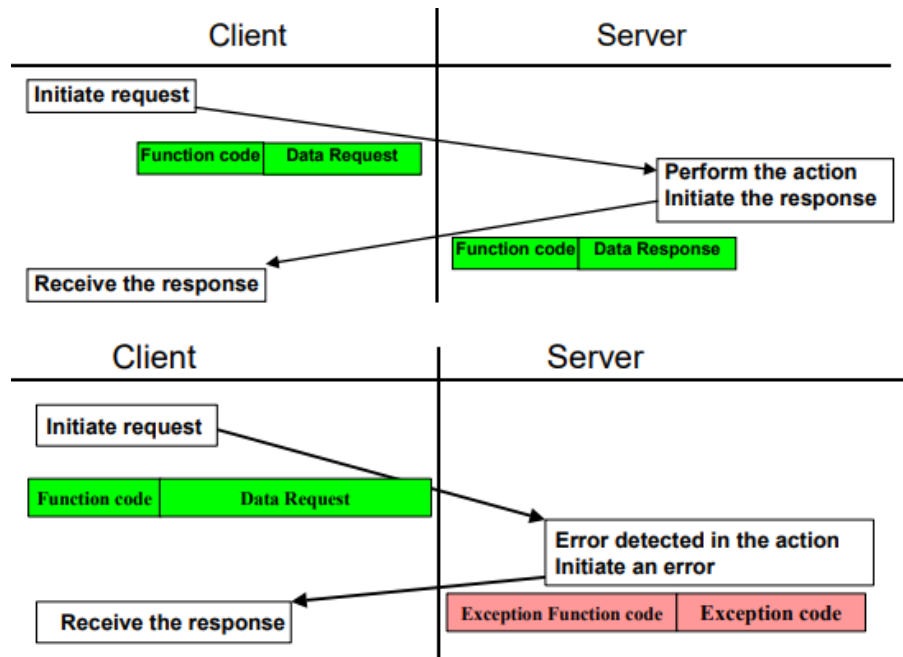


Ilustración 2.5 Solicitud y respuesta ante una solicitud exitosa y ante error [18]

### 2.5.2 Modelo de datos MODBUS

MODBUS organiza los diferentes datos que admite en cuatro bloques principales claramente diferenciados:

- **Discrete Inputs (Entradas digitales)** – Cada dato ocupa 1 bit y son elementos de sólo lectura.
- **Coils (Salidas digitales)**– Cada dato ocupa 1 bit y son elementos que soportan tanto lectura como escritura (pueden ser manipulados por aplicaciones).
- **Input Registers (Entradas analógicas)**– Cada dato tiene una longitud de 16 bits (una palabra) y son elementos de sólo lectura (son una representación de un valor analógico).
- **Holding Registers (Salidas analógicas)**– Cada dato tiene una longitud de 16 bits (una palabra) y son elementos que soportan tanto lectura como escritura (pueden ser manipulados por aplicaciones).

Como se puede observar, la clasificación se realiza atendiendo a la longitud del dato (medida en bits), los permisos que una aplicación tiene sobre el dato (sólo lectura, o lectura



y escritura) y la naturaleza del dato (entrada o salida). Cada tipo de dato dispone de un total de 65.536 elementos con los que se puede interactuar de manera independiente [18].

### 2.5.3 Direccionamiento de los datos MODBUS

En una PDU, cada dato se direcciona de 0 a 65.535, y comúnmente cada elemento de datos está numerado de 1 a N, siendo N menor o igual a 65.536, aunque en algunas aplicaciones el numerado de los elementos coincide con el direccionamiento de los mismos. Por esta libertad, es necesario vincular el modelo de datos MODBUS con la aplicación específica que da soporte a un determinado dispositivo. Para ello, se hace uso de un mapeado entre la aplicación y el modelo de datos MODBUS, el cual es específico del equipo en cuestión y de su fabricante [18]–[20].

Se debe tener en cuenta que la división que realiza el modelo de datos MODBUS en cuatro bloques diferenciados es puramente conceptual, pudiendo intercalarse y superponerse diferentes tipos de datos en la memoria del dispositivo. Con el objetivo de unificar el criterio de direccionamiento de datos, es común añadir un prefijo determinado a las direcciones en cuestión. Los prefijos comúnmente utilizados para los diferentes tipos de datos son los siguientes [20]:

Tipo de dato	Prefijo
Coils	0
Discrete Inputs	1
Input Registers	3
Holding Registers	4

*Tabla 2.1 Prefijos asignados a los rangos de datos MODBUS [20]*

Este prefijo precederá a la dirección correspondiente, de tal modo que el rango de valores 400.001 a 465.536 se identificará inequívocamente con el rango de direcciones correspondiente a los registros de retención.

### 2.5.4 Principales funciones predefinidas

El estándar MODBUS ofrece a sus usuarios la posibilidad de definir sus propias funciones, y de asignarlas a una serie de códigos de función reservados para este propósito, no obstante, es común hacer uso de las funciones predefinidas únicamente, ya que ofrecen una versatilidad suficiente para la mayoría de usuarios, cubriendo la mayor parte de escenarios posibles. Algunas de las principales funciones de las que dispone el estándar son [18]:

- **(0x01) Read Coils:** Función utilizada para leer de 1 a 2000 estados contiguos binarios (denominados bobinas por el estándar, como sinónimo de salidas binarias). Se ha de especificar la dirección inicial y el número de estados que se desean leer. El estado se indica como 1= ON y 0= OFF.

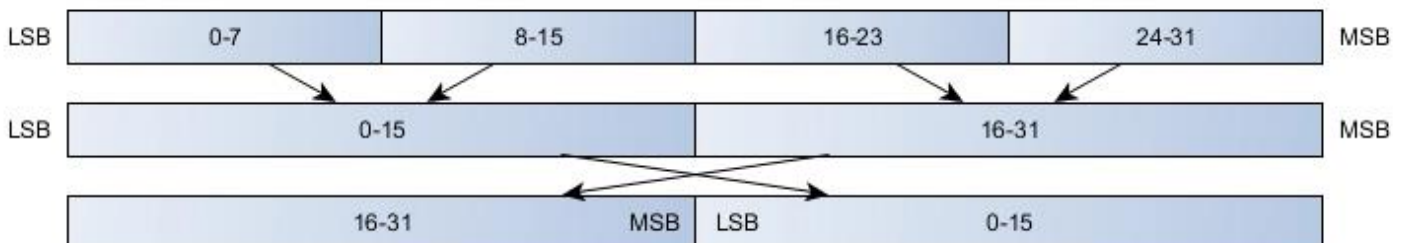


- **(0x02) Read Discrete Inputs:** Función análoga a la anterior, pero utilizada en la lectura de entradas binarias (o discretas, como las identifica el estándar).
- **(0x03) Read Holding Registers:** Función utilizada para leer de 1 a 125 registros contiguos (denominados registros de retención o almacenamiento, como sinónimo de registros de memoria, que hacen las veces de salidas de tamaño palabra). Los datos del registro en el mensaje de respuesta se empaquetan como dos bytes por registro (16 bits, una palabra).
- **(0x04) Read Input Registers:** Función análoga a la anterior, pero utilizada en la lectura de registros de entrada (por ejemplo, los correspondientes a un conversor analógico-digital).
- **(0x05) Write Single Coil:** Similar a *Read Coils*, pero ofrece la posibilidad de escribir en un único estado, dado que este tipo de datos ofrecen permisos tanto de lectura como escritura. Existe una versión de escritura multiestado, la cual corresponde a la función **(0x0F) Write Multiple Coils**.
- **(0x06) Write Single Register:** Similar a *Read Holding Registers*, pero ofrece la posibilidad de escribir en un único registro de memoria, dado que este tipo de datos ofrecen permisos tanto de lectura como escritura. Existe una versión de escritura multirregistro, la cual corresponde a la función **(0x10) Write Multiple Registers**.

### 2.5.5 Endianness

El estándar de Modbus especifica que, al enviar un tipo de datos de tamaño superior a un byte, como es el caso de los registros (16 bits, 2 bytes), se enviará primero el byte más significativo (*MSB*), seguido del byte menos significativo (*LSB*). Esto es lo que comúnmente se denomina una representación *Big Endian* [18], [20].

Sin embargo, los conjuntos de datos de tamaño superior a la palabra (16 bits) no aparecen especificados por el estándar, y es común encontrar equipos que invierten el orden de las palabras enviadas, por ejemplo, en el caso de codificaciones de coma flotante simple (32 bits, 2 palabras), enviando los registros con una codificación de palabra *Big Endian* (para cumplir con el estándar) y una codificación de datos multipalabra *Little Endian* [20].





Es necesario entonces en estos casos que el maestro sea conocedor de la codificación utilizada por los esclavos, con el objetivo de formular correctamente las solicitudes e interpretar correctamente los datos recibidos.

## 2.6 Conceptos informáticos fundamentales en la comprensión de la implementación

Debido a que el desarrollo de la herramienta en cuestión conlleva la programación de diferentes funcionalidades informáticas, en el presente proyecto se manejarán una serie de términos con los que el lector puede o no encontrarse familiarizado, por lo que conviene incluir un apartado específico para tratarlos.

Dentro de los sistemas informáticos multitarea, existen dos unidades básicas que, pese a guardar semejanza, se deben diferenciar claramente [21], [22]:

- **Hilo:** Representa un fragmento de código para ejecutar una actividad u operación de forma simultánea con otros hilos. Un hilo se ejecuta en el contexto de un determinado proceso, compartiendo con el resto de hilos del mismo proceso el espacio de memoria, instancia del intérprete (en el caso de lenguajes de programación interpretados), archivos... lo que agiliza el trabajo con los recursos que se desean manipular, pero da origen a la necesidad de gestionar el acceso a los mismos por parte de los múltiples hilos de ejecución, evitando comportamientos no deseados.
- **Proceso:** Representa un entorno de mayor complejidad al de un hilo, donde aparte de ejecutarse tareas de manera simultánea con otros procesos, se dispondrá de un espacio de memoria independiente para ese proceso, una instancia del intérprete no compartida (en el caso de lenguajes de programación interpretados), albergará en su seno diferentes hilos de ejecución... Como contrapartida, el intercambio de información entre procesos es más delicado que entre hilos.

La característica común de estos elementos se puede definir entonces como la capacidad de ejecutar un fragmento de código, de mayor o menor complejidad. Dicho código puede estar constituido por una serie de sentencias distribuidas de manera descendente, combinadas también con condicionales, bucles y funciones (lo que correspondería a la programación estructurada más tradicional), o se puede componer de una serie de fragmentos de código predefinido y reutilizable, denominados **clases**, las cuales representan una serie de características y funcionalidades básicas que un conjunto de elementos de código van a poseer. Dichos elementos se denominan **objetos**, y se tratan de la entidad básica del paradigma de programación orientada a objetos. Estos objetos constan de una serie de datos almacenados (denominados generalmente atributos) y una serie de tareas (denominadas generalmente métodos) íntimamente relacionadas con el objeto en cuestión [23].

Las necesidades del proyecto demandan la implementación de una interfaz gráfica, con el objetivo de incrementar la usabilidad de la herramienta desarrollada. Debido a que la



interfaz desarrolla se trata de una aplicación web, los siguientes conceptos serán de uso habitual a lo largo del proyecto:

- **Framework:** Constituye un conjunto de librerías, conceptos, arquitecturas, prácticas y métodos concretos para abordar un determinado problema. En el mundo del desarrollo de software, constituye un entorno bien documentado, compuesto por diferentes módulos, herramientas, recursos... que simplifican y sistematizan la manera de abordar y solucionar un problema [24]–[26].
- **ORM:** De sus siglas, mapeo objeto-relacional, es una técnica de programación que permite interactuar con bases de datos relacionales utilizando la potencia de la programación orientada a objetos, tratando las tablas de la base de datos en cuestión como objetos con determinados atributos. La forma de interactuar con la base de datos relacional se asemeja entonces a la forma de interactuar con bases de datos orientadas a objetos [27].
- **Microframework:** Constituye un concepto muy ligado con el anterior, pero particularizado para aplicaciones web. Un *microframework* se enfoca exclusivamente en agilizar el manejo de solicitudes HTTP, a diferencia de un *framework* completo que, además, contemplaría gestión de usuarios, manejo de bases de datos, validación y saneamiento de datos... [25], [26]
- **Frontend y Backend:** Constituyen los elementos principales en los que se puede diferenciar el desarrollo web. El *Frontend* se enfoca en la forma en que se presentan los datos al usuario final, y en la manera de interactuar que tienen el usuario y la interfaz desarrollada, mientras que el *Backend* representa toda la arquitectura interna que da soporte a la web, transparente para el usuario final, y donde se suceden toda una serie de acciones y sentencias lógicas para lograr una actividad concreta, como por ejemplo, el procesamiento y devolución de datos [28].





## 2.7 El lenguaje de programación Python

Python se trata de un lenguaje de programación de código abierto, multiparadigma, multiplataforma, dinámico e interpretado con una sintaxis limpia y cercana al propio inglés [29].

Fue desarrollado por Guido van Rossum en el Centro para las Matemáticas y la Informática (CWI, *Centrum Wiskunde & Informatica*), en los Países Bajos, con los objetivos principales de desarrollar un lenguaje [30]:

- Fácil, intuitivo y tan potente como sus principales competidores
- De código abierto, para favorecer la participación por parte de la comunidad
- Legible
- Apto para perfiles con una base matemática suficiente y sin necesidad de poseer un conocimiento profundo en lenguajes de programación
- Apto para su utilización en el día a día, ofreciendo un rápido prototipado para las soluciones que precisen sus usuarios

Python soporta la integración de módulos desarrollados mediante lenguajes clásicos como C/C++, haciéndolo un lenguaje de programación muy adecuado para perfiles muy variados, dada la combinación de una sintaxis simple y legible, una gran libertad para el usuario a la hora de acogerse a un determinado paradigma de programación, junto con toda la potencia de lenguajes menos elegantes. En la actualidad goza de una gran popularidad por su capacidad para ser utilizado en entornos de análisis de datos, inteligencia artificial y *machine learning* [29], [31].

En la actualidad existen numerosas implementaciones del lenguaje, como son [32]:

- *Cpython*, la cual se trata de la implementación original y una de las más extendidas, escrita en C.
- *Jython*, la cual se trata de una implementación del lenguaje desarrollada en Java.
- *IronPython*, implementada en .NET.

Se debe tener en consideración un problema asociado a la implementación CPython. Pese a tratarse de una de las implementaciones más eficientes debido a la eficiencia del propio lenguaje C, presenta un problema en la actualidad muy discutido, y debido a que Python se trata de un lenguaje interpretado.

Cada proceso hace uso de una instancia del intérprete para ejecutar el código desarrollado, pero los distintos hilos que conviven en un determinado proceso compartirán el intérprete del mismo, al tratarse de un recurso más del mencionado proceso.

La compartición del intérprete de Python por los hilos alojados en un determinado proceso supone la imposibilidad de una implementación real de paralelismo en la ejecución de múltiples tareas en un sistema multihilo, con independencia del hardware del que se disponga. La instancia del intérprete de Python se trata en realidad como un recurso más



al que pretende acceder cada uno de los diferentes hilos en ejecución. La solución utilizada para abordar este problema durante la concepción del lenguaje de programación pasó por la implementación del *Global Interpreter Locker* (GIL), el cual simplemente se trata de una primitiva de sincronización que otorga el control del intérprete a un solo hilo durante un instante de tiempo, cediéndoselo a otro hilo tras un cierto período de tiempo. De esta manera se implementa un sistema multitarea con una falsa apariencia de paralelismo, al intercalarse constantemente ejecuciones de parte del código de cada uno de los hilos, de manera secuencial [33], [34].

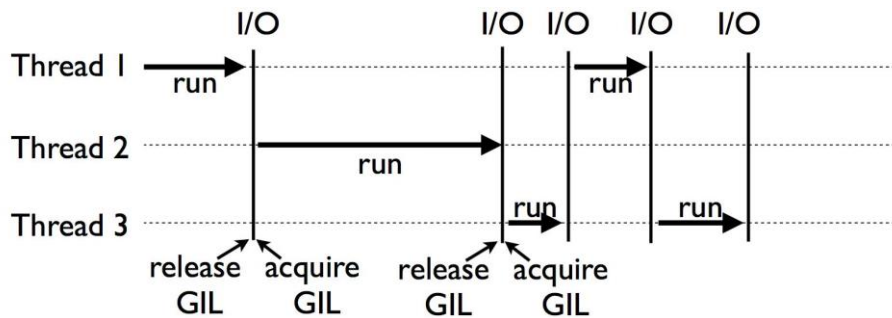


Ilustración 2.6 Esquema de una ejecución multihilo en Python [33]

Los sistemas multiproceso, en cambio, sí que implementan un paralelismo real, puesto que disponen de su propia instancia del intérprete de Python. No obstante, son elementos más complejos computacionalmente hablando, y que necesitan un mayor número de recursos para su despliegue, lo que también se traduce en un incremento de los tiempos de ejecución, que reduce las ventajas del paralelismo.

### 2.7.1 Principales frameworks para el desarrollo web

A modo de breve repaso, los principales *frameworks* de desarrollo web existentes para Python en la actualidad son [26]:

- **Flask:** Constituye un *microframework* para el desarrollo de aplicaciones web con un enfoque minimalista, centrado en el prototipado rápido de pequeñas soluciones web que impliquen una cantidad mínima de líneas de código. Es muy utilizado, por ejemplo, en el desarrollo de APIs REST.
- **Django:** Es un *framework* completo para el desarrollo web, de mucha mayor envergadura que Flask, y que se centra en todo lo contrario, esto es, simplificar el desarrollo de aplicaciones web complejas, ya que implementa funcionalidades muy interesantes, como son: sistema predefinido para el manejo de diferentes roles de usuarios, ORM propio para interactuar con la base de datos de la aplicación web, funciones de validación y saneamiento de los datos en cuestión, motor de plantillas web con una serie de etiquetas y filtros predefinidos para aumentar la reutilización del código...

Aunque Flask ofrece la posibilidad de instalar una serie de paquetes específicos (denominados *Extensiones*) que implementan soluciones muy variadas, aumentando la



funcionalidad del *microframework*, al abordar proyectos de cierta envergadura con el mismo, la potencia de Flask para el desarrollo minimalista desaparece, y las soluciones a implementar, aunque posibles, resultan más complicadas que su equivalente desarrollado con Django. Por otro lado, también cabe destacar que la simplicidad de Flask hace que la curva de aprendizaje del mismo sea muy reducida en comparación de Django, debido a la cantidad limitada de recursos que ofrece por defecto.

## 2.7.2 Librerías destacadas que se han utilizado

Merece la pena detallar algunas de las principales librerías que se han utilizado en el presente proyecto, y que tienen por objetivo ofrecer una solución directa a alguno de los problemas que más adelante se comentarán:

- **Pandas:** Constituye una biblioteca ampliamente conocida en el mundo del análisis de datos, construida sobre la extensión NumPy de Python, y que ofrece una serie de funcionalidades para la manipulación, procesamiento y tratamiento de datos. También ofrece una serie de funcionalidades básicas para implementar el volcado de datos en bases de datos SQL [35].
- **SQLAlchemy:** Comprende una serie de herramientas que ofrecen un ORM para interactuar con bases de datos relacionales como si se tratase de bases de datos orientadas a objetos. Agiliza en gran medida la conexión, consulta y volcado de datos a una base de datos [36].
- **PyModbus:** Se trata de una implementación completa de Modbus, que permite, entre otras funcionalidades, definir un cliente Modbus de manera sencilla, con el cual solicitar datos a los diferentes equipos que trabajarán como servidores, haciendo uso de estos datos directamente en el programa o script que estemos desarrollando [37].
- **APScheduler:** Se trata de un marco de tareas programadas con Python, que proporciona una serie de funcionalidades diferentes de programación de tareas; al mismo tiempo que también proporciona diferentes mecanismos de almacenamiento, los cuales se traducen en el registro del estado y configuración de los hilos programados en la base de datos, de tal modo que no se necesita una redefinición de los mismos ante el reinicio del sistema, ofreciendo cierta persistencia. Ofrece una gran flexibilidad para configurar el lanzamiento y ejecución programado de las diferentes tareas, ya sea atendiendo a intervalos, cronómetros, fechas determinadas... [38]



## 3 Trabajo desarrollado

A continuación, se detallarán las especificaciones técnicas y el diseño conceptual de la herramienta, el trabajo desarrollado en torno a dicho diseño, las diferentes alternativas que se plantearon, la solución que se decidió implementar en cada caso, los problemas encontrados y la resolución de los mismos durante dicha implementación.

### 3.1 Especificaciones técnicas y diseño

Los principales objetivos establecidos como punto de partida del proyecto, así como la traducción de los mismos en características que ha de poseer la herramienta en cuestión, se derivan de los problemas identificados en herramientas comerciales con un propósito similar y ya existentes en el mercado, esto es:

- Desarrollo de un sistema de adquisición de datos cuya frecuencia de adquisición sea parametrizable, para abordar actividades con diferentes exigencias en este aspecto.
- Sistema sencillo de cara a la configuración y filtrado de los datos que se desean capturar, evitando la captura masiva de datos que carecen de interés para el usuario.
- Desarrollo de una herramienta multifabricante y multipropósito, con capacidad de gestionar equipos de diferentes características y enfocados en diferentes actividades.
- Utilización de estándares de comunicaciones de uso extendido en dispositivos multipropósito.
- Desarrollo de una interfaz de usuario que permita la operación con la herramienta de forma transparente a los recursos que dan soporte a la misma, facilitando su operación.
- Marcar como objetivo la escalabilidad del sistema desarrollado, con el fin de prever futuras ampliaciones y cobertura de nuevos equipos, tecnologías...

Condensando los objetivos y especificaciones anteriores, se ha concebido la herramienta a implementar con una estructura centralizada en torno a un elemento que se comunicará con los sistemas físicos. Dicho elemento se encargará principalmente de realizar solicitudes MODBUS/TCP y procesamientos de los datos obtenidos como respuesta, no obstante, contará con dos funcionalidades secundarias: por un lado, establecer comunicación con la base de datos seleccionada para almacenar los datos adquiridos, y donde este elemento central volcará los datos procesados. Por otro lado, el elemento de adquisición y procesamiento de datos establecerá comunicación con una aplicación web, con el objetivo de recibir una serie de órdenes y datos de configuración, de tal modo que la operación del sistema de adquisición de datos se simplificará para el usuario final.



De modo que la arquitectura del sistema girará en torno a un elemento central de adquisición de datos, que se servirá de dos elementos periféricos (aplicación web y base de datos) para operar. El usuario, a su vez, podrá interactuar con dichos elementos periféricos (acceso vía web en el caso de la aplicación web, y utilización de gestores de bases de datos comerciales en el caso de la base de datos), siendo completamente transparente para el mismo todo lo que ocurra aguas abajo.



### 3.2 Planificación del trabajo desarrollado

Con el objetivo de organizar el trabajo a desarrollar, se ha elaborado un cronograma que recoge las tareas principales en las que se divide el presente proyecto, ordenándolas de una forma lógica y eficiente:

Tarea	Descripción	15/10/2021	01/11/2021	15/11/2021	01/12/2021	15/12/2021	01/01/2022	15/01/2022	01/02/2022	15/02/2022	
A	Documentación y estudio	■			■		■				
B	Comunicación con los equipos		■								
C	Desarrollo del sistema multihilo				■						
D	Desarrollo de la aplicación web						■				
E	Desarrollo de las interfaces de comunicación							■	■		
F	Pruebas y resolución de problemas			■		■					
G	Evaluación de resultados								■		
H	Desarrollo del documento		■								

Ilustración 3.1 Cronograma del trabajo desarrollado

Las tareas principales que recoge el cronograma son:

- **Comunicación con los equipos:** Esta tarea se centra en el desarrollo del código necesario para establecer comunicación de forma ordenada y automatizada con los equipos de la red MODBUS/TCP.
- **Desarrollo del sistema multihilo:** Solventados los problemas para establecer comunicación, se procede con el desarrollo del sistema multihilo, con el objetivo de racionalizar y sistematizar la captura de los datos.
- **Desarrollo de la aplicación web:** Una vez se dispone de un sistema de adquisición de datos funcional, se le dotará de la interfaz web para agilizar su utilización.
- **Desarrollo de interfaces de comunicación:** Una vez se dispone del sistema de adquisición de datos y de su interfaz web, se implementa una solución específica para comunicar ambos recursos.

Estas tareas están soportadas por las siguientes:

- **Documentación y estudio:** Antes de abordar la implementación de ninguna de las tareas anteriormente mencionadas, se necesita consultar y analizar información de la materia en cuestión, con el objetivo de esclarecer la solución a implementar.
- **Pruebas y resolución de problemas:** A medida que el sistema va tomando forma y creciendo en complejidad, se hacen necesarias una serie de pruebas y testeos de cada parte de la herramienta, con el objetivo de corroborar el correcto funcionamiento de la misma.

A lo largo de prácticamente la totalidad del proyecto, se ha ido elaborando este documento, a medida que cada tarea o etapa se ha completado.

Finalmente, se han llevado a cabo los experimentos que han servido para evaluar los resultados de la herramienta desarrollada.



### 3.3 Descripción de la arquitectura conceptual e implementación

La arquitectura concebida y anteriormente descrita se muestra a continuación, y sobre la misma nos apoyaremos para comprender en profundidad el funcionamiento de la herramienta desarrollada, comentando las diferentes partes de la misma, las alternativas que en cada caso se plantearon y el porqué de la solución abordada finalmente para resolver los distintos problemas que han ido apareciendo a lo largo del desarrollo del proyecto:

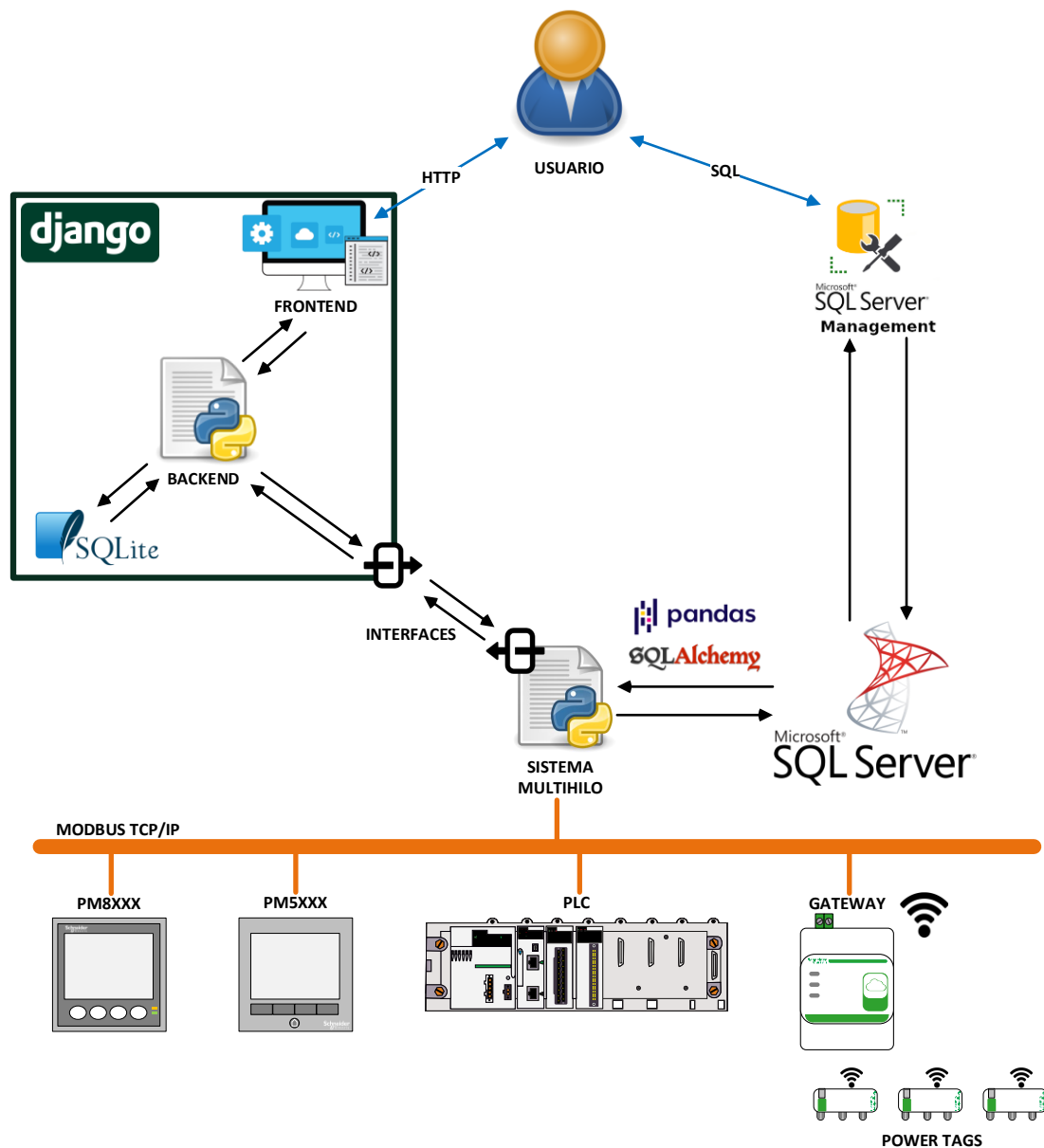


Ilustración 3.2 Arquitectura conceptual de la herramienta

Como se puede observar, la arquitectura del sistema posee una serie de partes claramente diferenciadas:



- Una aplicación web (desarrollada mediante el *framework* Django), así como de la aplicación de escritorio Microsoft SQL Server Management Studio. Estas herramientas servirán como interfaz de usuario para interactuar con el sistema.
- El *backend* de la aplicación web, el cual servirá para procesar los datos introducidos en el *frontend* y prepararlos para servírselos al sistema multihilo.
- Un sistema multihilo como elemento central de la arquitectura, el cual es el encargado de gestionar la captura de datos y el volcado de los mismos en una base de datos de Microsoft SQL Server.
- Por debajo del sistema multihilo se encuentran los diferentes equipos conectados a la red MODBUS TCP/IP, los cuales actuarán como servidores ante las solicitudes realizadas por el sistema multihilo, que funcionará como cliente.

### 3.3.1 Sistema multihilo de adquisición de datos

El sistema de adquisición de datos desarrollado se ha implementado mediante el lenguaje de programación Python, con su implementación estándar CPython, haciendo uso de diferentes librerías disponibles y desarrolladas por la comunidad, así como de librerías propias desarrolladas para satisfacer las necesidades del presente proyecto. A la hora de concebir el sistema de adquisición de datos, se plantean principalmente tres problemas a solventar:

- Encontrar una manera de establecer comunicación con los distintos equipos, de forma replicable, automatizable y lo más estandarizada posible.
- Gestionar el volcado de los datos adquiridos en la base de datos.
- Encontrar una manera de realizar la adquisición de datos de forma determinista, dando lugar a una escala de tiempos bien definida, de modo que cualquier error puntual en la lectura puede corregirse mediante una reconstrucción de la misma a partir de las lecturas realizadas.

A continuación, se detallan las soluciones abordadas para resolver los problemas planteados, así como los principales inconvenientes encontrados al abordar cada solución, la forma en que se han resultado dichos inconvenientes, y las alternativas a la solución implementada, en caso de que se contemplasen.

#### 3.3.1.1 Comunicación con los equipos

Uno de los objetivos principales marcados para el presente proyecto es que la herramienta desarrollada soporte una gran variedad de dispositivos. La razón de comunicarse con los equipos vía MODBUS TCP/IP reside precisamente en que el protocolo MODBUS es ampliamente utilizado tanto para comunicaciones con equipos de redes de *submetering* como para equipos orientados a la automatización de procesos industriales, por lo tanto, incorporar un estándar de comunicaciones de este tipo en el proyecto garantiza el soporte a dispositivos multipropósito y multifabricante.





Para dotar al sistema mutihilo de la capacidad de actuar como cliente MODBUS, se ha utilizado la librería *PyModbus*. No obstante, como ya se mencionó en el apartado de *Background*, a la hora de trabajar con el protocolo MODBUS para comunicarse con un determinado dispositivo, existen una serie de particularidades que se han de resolver y que dependen íntegramente del fabricante del mismo, como es, por ejemplo, el direccionamiento de los datos MODBUS, además del hecho de que el objetivo de soportar equipos de diferente naturaleza complica la estandarización y automatización de la comunicación con los mismos, puesto que se han de resolver las particularidades y peculiaridades de cada tipo de equipo para establecer una comunicación satisfactoria con los mismos.

Los diferentes equipos a los que se pretende dar soporte en el desarrollo del presente proyecto son:

- Equipos enfocados en el *submetering* – Medidores de las familias PM8XXX, PM5XXX y Power Tags.
- Controladores lógicos programables – PLCs M340
- Pasarelas para habilitar comunicaciones vía TCP/IP con Power Tags (las cuales colgarán de las pasarelas, compartiendo IP y diferenciándose mediante un ID único).

Se puede hacer una clasificación y reagrupación de los mismos en función de sus características comunes y sus particularidades:

- Las pasarelas, aparte de establecer la dirección IP asociada a una red de Power Tags, únicamente servirán como repositorio de las propias Power Tags. Las interacciones con las mismas se realizarán apuntando a las pasarelas.
- A excepción de las pasarelas, todos los equipos cuentan con una lista de direcciones (bien un mapa MODBUS en el caso de equipos de *submetering*, o bien una lista de variables en el caso de los PLCs), las cuales se podrán filtrar para seleccionar únicamente aquellas de interés.
- A excepción de las Power Tags, todos los equipos cuentan con una dirección IP, y éstas poseen una dirección ID.
- A excepción de las pasarelas, todos los equipos son susceptibles de generar datos. Estos datos, en función del equipo, tienen ciertas particularidades:
  - 1) La numeración de las direcciones puede coincidir o no con la propia dirección (en algunos equipos la numeración se corresponde con la dirección de un dato más 1, como se comenta en el apartado de *Background*).
  - 2) La forma en la que los bytes de una determinada palabra se ordenan (primero el más significativo o voceversa), así como se ordenan las palabras dentro de un conjunto de datos de tamaño superior a los 16 bits, depende de cada equipo, como se comenta en el apartado de *Background*.



La forma más natural para desarrollar la solución de comunicación con los equipos pasará por el aprovechamiento del enfoque orientado a objetos que ofrece Python, definiendo diferentes tipos de clases que implementen las funcionalidades y características que deben poseer los distintos tipos de equipos, de esta manera también se simplifican previsible actualizaciones futuras para dar soporte a nuevos equipos, garantizando una máxima reusabilidad del código desarrollado.

La implementación de estas clases se ha llevado a cabo mediante el desarrollo de dos módulos:

- **baseClasses.py**: el cual implementa una serie de clases abstractas, que sirven como base para la implementación posterior de clases concretas orientadas a ofrecer soporte a una serie de equipos concretos.
- **Classes.py**: el cual hace uso del módulo anterior para implementar las clases concretas que darán soporte a los equipos, mediante la particularización de los métodos definidos en las clases base.

En estos módulos también se han desarrollado una serie de clases cuya finalidad es implementar el proceso de almacenamiento de datos, conexión y volcado de los mismos en SQL Server, no obstante, este hecho será pasado por alto en este punto, centrándonos únicamente en la comunicación con los equipos.

Como se puede observar en la *Ilustración 3.2 Arquitectura conceptual de la herramienta*, en el módulo *Classes.py* básicamente se han definido 4 clases para comunicarse con los equipos:

- **Submetering**: Clase que da soporte a los medidores PM8XXX y PM5XXX. Las Power Tags también utilizarán las funcionalidades y características de esta clase, aunque con alguna pequeña particularización. Básicamente define una forma particular de realizar el filtrado de las direcciones MODBUS, para seleccionar únicamente las que son de interés de entre las posibles.
- **PowerTagGateway**: Como su nombre indica, implementa funcionalidades y características para representar a las pasarelas de las Power Tags. Básicamente permite interactuar con las Power Tags de manera centralizada.
- **PowerTag**: Clase anidada en la anterior, permite definir nuevas Power Tags. Como ya se mencionó, hereda de la clase *Submetering*, añadiendo a ésta la característica de una dirección ID. Solamente es accesible desde el contexto de las instancias de la clase *PowerTagGateway*.
- **PLC**: Clase que da soporte a los PLCs M340. Al igual que la clase *Submetering*, define su propia forma para filtrar la lista de variables del PLC y seleccionar únicamente las variables de interés.

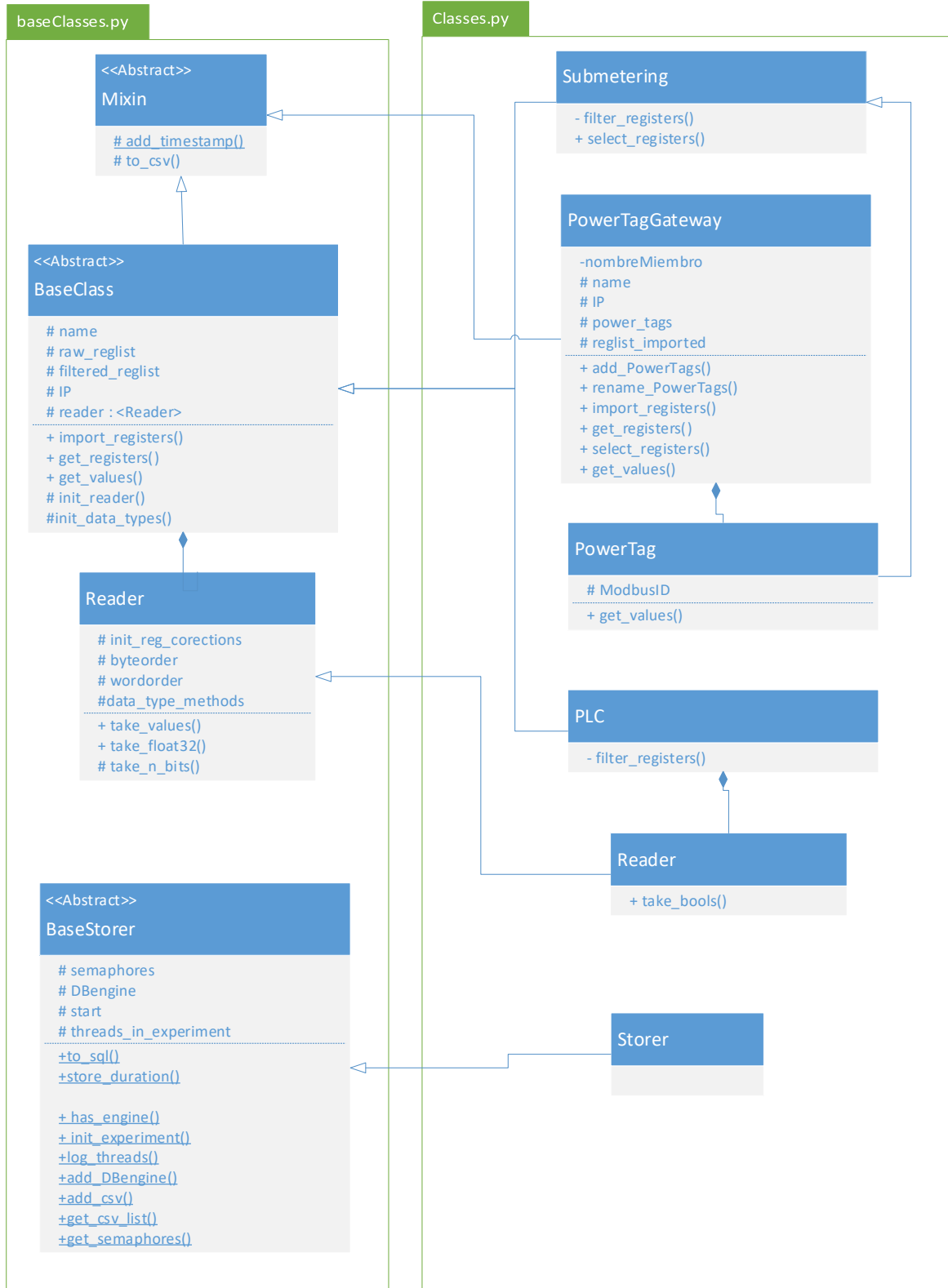


Ilustración 3.3 Diagrama de clases de los módulos desarrollados



A excepción de la clase *PowerTagGateway*, las clases anteriores se basan en la clase *BaseClass* del módulo *baseClasses.py*, la cual reúne las siguientes características:

- Define una serie de características que poseerán los objetos que se van a instanciar, como son: nombre, dirección IP, lista de registros en crudo, lista de registros filtrados y un objeto lector.
- Define una serie de funcionalidades genéricas, cuya implementación definitiva se lleva a cabo en las clases particulares del módulo *Classes.py*, como son: importar la lista de registros completa, obtener la lista de registros filtrada, obtener los valores asociados a dichos registros.
- Dispone de una clase anidada denominada *Reader*, cuya funcionalidad es definir la herramienta de comunicación y lectura de datos MODBUS. Se ha decidido implementar de esta manera la comunicación MODBUS para tener un solo punto centralizado donde definir las cuestiones referentes al posible offset entre numeración y direccionamiento de los datos MODBUS, ordenamiento de los bits dentro de cada byte, y ordenamiento de los bytes dentro de cada palabra. Además, también será en esta clase donde se implemente el soporte para los diferentes tipos de datos que se desean leer, así como el preprocesamiento de los mismos para recibir siempre una respuesta estándar de la solicitud MODBUS realizada.

### 3.3.1.2 Volcado de datos a la base de datos

Una vez resuelto el problema de la comunicación con los equipos para realizar las peticiones MODBUS de los datos de interés, se debe racionalizar el proceso de almacenamiento de los mismos en la base de datos destinada a tal propósito. Las alternativas inicialmente planteadas para llevar a cabo esta labor se pueden resumir en las siguientes:

- **Volcado directo de los datos recibidos a la base de datos, haciendo uso de la librería Pandas para manejarlos:** tiene la ventaja de ser fácilmente implementable, puesto que la información se vuelca directamente a la base de datos sin necesidad de pasos intermedios, no obstante, presenta el gran inconveniente de lanzar sucesivas conexiones contra la base de datos prácticamente en tiempo real, si se piensa en las actividades de captura de datos con intervalos de captura muy reducidos, por lo que el proceso de conexión con la base de datos supondrá un importante condicionante técnico en este tipo de actividades muy exigentes en cuanto a latencia se refiere.
- **Volcado de los datos recibidos a archivos CSV temporales, configurando el volcado posterior de los datos de estos archivos a la base de datos:** resuelve los problemas asociados con la anterior, puesto que al trabajar con archivos temporales estaremos concentrando la información para hacer un único volcado de todo el conjunto de datos cuando se desee, estableciéndose entonces una periodicidad de la actividad del volcado de datos a la base de datos, con



independencia del intervalo con el que se solicitan los mismos a los equipos en cuestión.

La segunda posibilidad ha sido finalmente la que se ha ejecutado en el presente proyecto, desarrollando la actividad del volcado de datos una entidad desarrollada a tal fin, que como se puede observar en la *Ilustración 3.3 Diagrama de clases de los módulos desarrollados*, se implementa mediante la clase *BaseStorer* del módulo *baseClasses.py*, de la cual hereda la clase *Storer* del módulo *Classes.py*. Esta clase, además de realizar el volcado de los datos, se encarga de llevar un registro de los equipos involucrados en cada experimento, de registrar el instante en que se inicia y finaliza un experimento concreto y lleva un registro de los CSVs que se deben manipular para realizar el volcado de los datos.

Cabe destacar que, como se ha optado por esta solución, se hace necesario implementar una funcionalidad para la generación de/volcado de datos a un CSV, así como la incorporación a estos datos de un *timestamp* que permita ordenarlos cronológicamente, utilizando para ello la clase *Mixin* del módulo *baseClasses.py*, cuyo único objetivo es definir estas dos funcionalidades y agilizar su incorporación al resto de clases mediante la herencia múltiple.

A mayores, cabe destacar que la solución adoptada presenta un problema al cual se le debe dar solución, y que se trata de gestionar el acceso a los CSVs, puesto que esporádicamente se dará la situación de que, tanto la actividad de adquisición y volcado de los datos al CSV, como la actividad de lectura de los datos del CSV y volcado de los mismos a la base de datos, intentarán acceder de forma simultánea al CSV en cuestión, dando lugar a un comportamiento indeseado por parte del sistema. Este problema se resuelve mediante la implementación en la clase *BaseStorer* de una serie de funcionalidades para controlar el acceso a los CSVs (denominadas primitivas de sincronización en la literatura existente en relación con la operación de sistemas informáticos multitarea), de modo que, si una actividad de adquisición de datos se encuentra manipulando el CSV en el mismo instante de tiempo en el que se pretende leer y volcar a la base de datos la información del CSV, esta última actividad esperará a la finalización de la primera para proceder.

### 3.3.1.3 Adquisición de datos de forma determinista

Tal y como se ha comentado, se hace necesaria una forma de ejecutar las tareas de adquisición de datos de manera cíclica y con un intervalo de ejecución determinado, con el objetivo de generar una escala de tiempos bien definida. Realmente a este respecto no existe una posibilidad alternativa más allá del desarrollo de un sistema multitarea, desarrollando cada actividad de captura de datos asociada a un equipo en una tarea independiente.

Teniendo esto en cuenta, y de acuerdo con lo mencionado en *El lenguaje de programación Python*, se ha implementado un sistema multihilo y no multiproceso, ya que se ha comprobado que los resultados obtenidos para ambos sistema multitarea son igualmente satisfactorios en el caso que se está tratando, y tanto el coste computacional como la



complejidad a la hora de realizar un intercambio de datos entre tareas se ven ampliamente reducidos.

Finalmente, para racionalizar la ejecución de cada uno de los hilos que implementan las actividades de captura de datos, se ha utilizado *APScheduler*.

### 3.3.2 Aplicación web

Se desea que el sistema multihilo desarrollado sea configurable directamente desde una interfaz web a la que se conecte el usuario, evitando así la necesidad de una comprensión en cierta profundidad tanto del lenguaje de programación Python como de las herramientas particulares (paquetes y módulos) que se han utilizado para el desarrollo de dicho sistema.

El presente proyecto comenzó el desarrollo de su aplicación web mediante Flask, pero este camino fue abandonado en cuanto se comenzó a ganar complejidad en la web a desarrollar, necesaria para dotarla de la flexibilidad que se deseaba alcanzar, por lo que se migró el desarrollo del mismo a Django, como se detalla a continuación.

#### 3.3.2.1 Funcionamiento del Backend

Django emplea una forma particular para desplegar una aplicación web, que es el denominado patrón MTV (modelo, vista, plantilla) [39]:

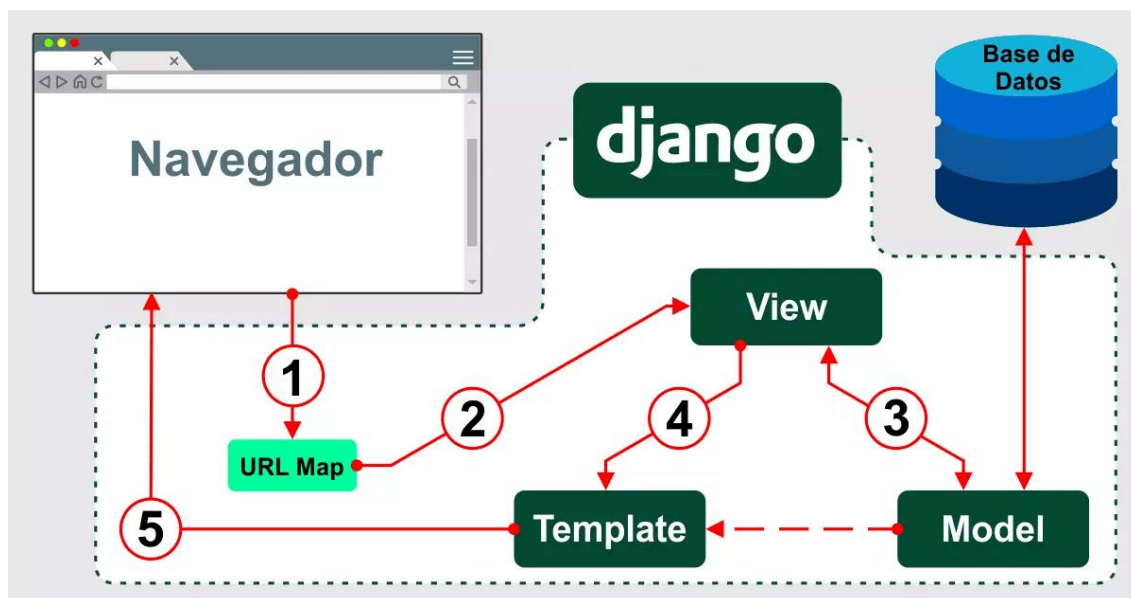


Ilustración 3.4 Respuesta de Django ante una solicitud [21]

- 1) A la hora de acceder a un enlace o dirección mediante el navegador, lo primero que hará la solicitud lanzada a Django será pasar por un documento denominado *urls.py*, el cual mapea cada vista con una URL específica, de tal modo que al acceder al enlace en cuestión Django conocerá qué vista aplicar a la solicitud que se le ha pasado.



- 2) El siguiente paso consiste en servir la solicitud a la vista en cuestión, las cuales se encuentran centralizadas en el documento *views.py*. Dichas vistas simplemente se implementan con una función o clase determinada, desarrollada a tal fin, y cuyo objetivo consiste en procesar la solicitud para preparar la información que se le solicite, y decidir qué plantilla será el encargado de mostrarla. La información de la solicitud que haga referencia al modelo será solicitada por la vista en cuestión.
- 3) El modelo aquí tiene la finalidad de servir de intermediario entre la base de datos, donde se centraliza la información correspondiente a la aplicación web, y la vista, que servirá la información solicitada a la plantilla indicado. Este modelo también se encargará de validar la información que la vista pretenda volcar a la base de datos.
- 4) Una vez la vista disponga de la información que se requiera de la base de datos, aplicará la lógica que se haya programado en la misma para procesarla, pasando a determinar qué plantilla la mostrará, y sirviéndosela a este último para tal fin. La plantilla es capaz de interpretar la información que la vista le pasa mediante una serie de variables de contexto, que incorporará en los lugares indicados del código HTML que se haya escrito en la plantilla en cuestión, haciendo uso de un sistema de etiquetado del cual dispone Django para permitir el incrustado de elementos como bucles, condicionales... intercalados con HTML. También podrán utilizarse estas variables de contexto para otras funcionalidades, como por ejemplo facilitar un *array* a un fragmento de código Javascript.
- 5) Finalmente, la plantilla de la web adecuada para satisfacer las necesidades de la solicitud que se realizó será devuelta al navegador, donde se muestra al usuario.

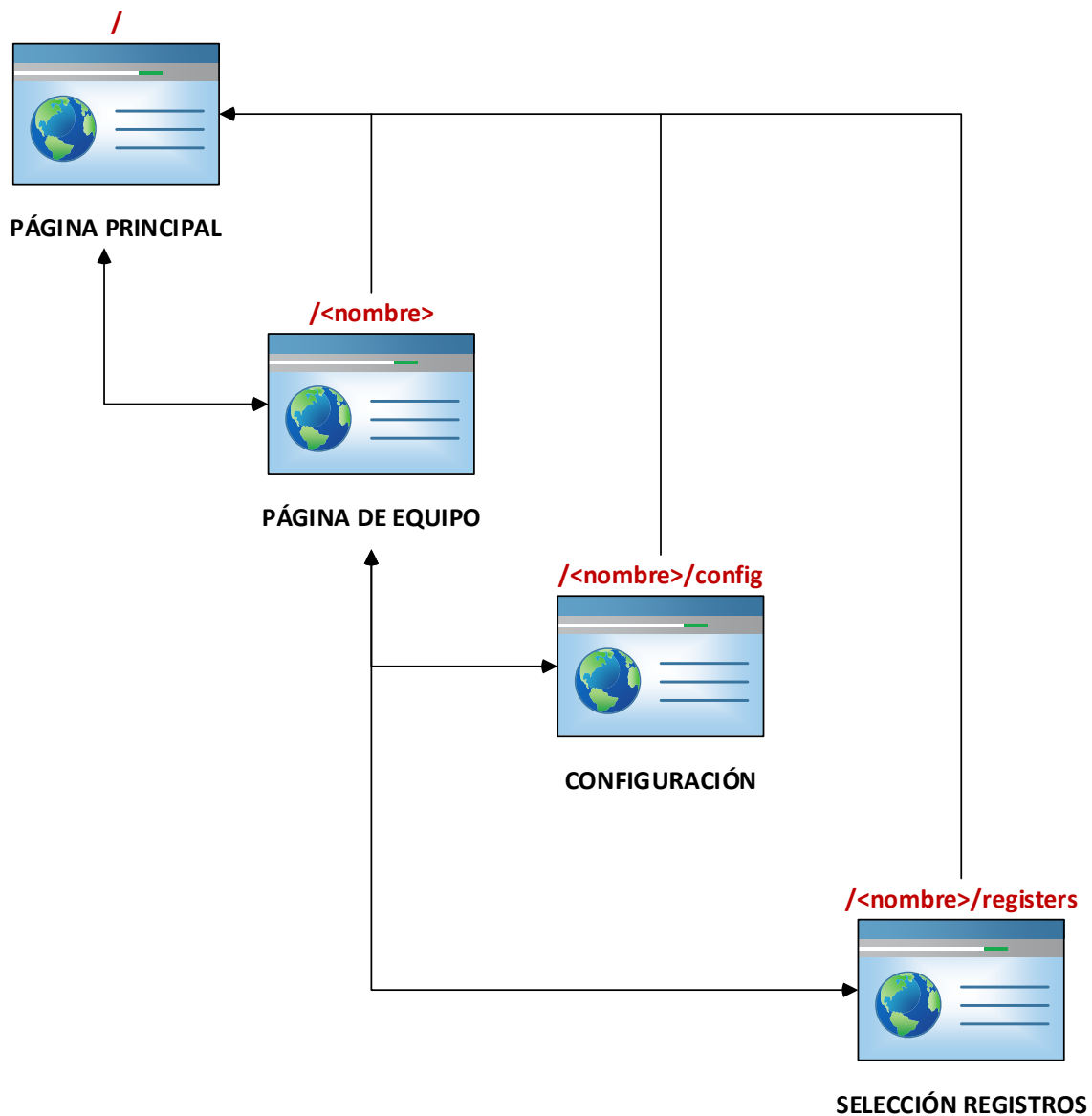
El elemento central del entorno de Django mostrado en la *Ilustración 3.2 Arquitectura conceptual de la herramienta*, e identificado como *Backend*, representa el proceso aquí mencionado, obviando una serie de documentos de los que depende a nivel técnico Django para dar soporte a los principales aquí mencionados, pero que carecen de interés a nivel conceptual para comprender el correcto funcionamiento de la herramienta desarrollada. La base de datos con la que interactuará el modelo de Django será en realidad un fichero SQLite, ya que esta funcionalidad es la proporcionada por defecto por Django, y ofrece un potencial suficiente para abordar la fase de desarrollo del presente proyecto. En cuanto al aspecto del *Frontend*, se han desarrollado una serie de páginas web por las que el usuario navegará, que pasaremos a explicar de manera detallada a continuación.

### 3.3.2.2 Descripción del Frontend

La aplicación web implementada para satisfacer las necesidades del presente proyecto se ha desarrollado con una estructura básica que se mostrará y comentará a continuación (*Ilustración 3.5 Estructura básica de la web desarrollada*), no obstante, y debido a la existencia de las pasarelas como elemento claramente diferenciado de entre los que



componen el proyecto, ha sido necesario concebir una estructura de páginas particular, con el objetivo de abordar la configuración y parametrización tanto de las pasarelas como de las Power Tags que se encontrarán colgando de las mismas, la cual se comentará posteriormente (*Ilustración 3.11 Estructura particular para dar soporte a pasarelas y Power Tags*).



*Ilustración 3.5 Estructura básica de la web desarrollada*

Cada una de las páginas anteriormente mostradas en la ilustración tiene unas funcionalidades muy claras:

- **Página principal:** Ofrece una vista general de la herramienta, con una serie de secciones claramente diferenciadas:
  - 1) Una sección para cargar los mapas MODBUS de los equipos que los necesitarán en su definición (esto es, las familias PM5XXX, PM8XXX y las Power Tags). Se pueden observar una serie de hipervínculos, los cuales





servirán para que el usuario, si lo desea, pueda descargar los documentos cargados en algún momento futuro en el que los precise.

- 2) Una sección para definir nuevos equipos, donde se puede hacer una configuración inicial de los mismos, o únicamente rellenar aquellos campos obligatorios (tipo, nombre e IP) y posponer la configuración.
- 3) Una sección donde se muestran los distintos equipos ya definidos, junto a su dirección IP, nombre, y su estado (representado mediante un LED que puede tomar 4 posibles colores: rojo, para representar un equipo definido pero no configurado; naranja, para representar un equipo correctamente configurado pero no asociado a ningún hilo de ejecución; amarillo, para representar un equipo asociado a un hilo pausado; verde, para representar un equipo asociado a un hilo en ejecución).
- 4) Una sección donde se incorporan una serie de botones para interactuar con el sistema de adquisición de datos (creación y destrucción de hilos, arranque y parada del sistema de adquisición de datos).

Desde esta página se podrá acceder a la página general del equipo que se desee, clicando sobre la imagen del mismo.

The screenshot displays the main interface of the web application, divided into three numbered sections:

- Section 1: Definición de Mapas Modbus** (top left). It contains three rows for defining Modbus maps. Each row has a label (e.g., 'Mapa Modbus (PM8xxx)'), a 'Seleccionar archivo' button, and a status 'Ningún archi... seleccionado'. Below each row is a list of files with their paths (e.g., '/media/modbusmaps/PM8000\_Modbusmap.xlsx'). At the bottom of this section are two buttons: 'Añadir mapas Modbus' and 'Borrar mapas Modbus'.
- Section 2: Definición de equipos** (middle left). It features a form for defining a new device. Fields include 'IP:', 'Nombre:', 'Equipo:' (with a dropdown menu showing 'PM8xxx'), and 'Intervalo de lectura de datos (s):'. An 'Añadir Equipo' button is located at the bottom.
- Section 3: Equipos definidos** (top right). This section displays a grid of six defined devices. Each device is represented by an icon (a monitor or a PLC rack) and a colored LED indicator above it. Below each icon is the device's name and IP address:
  - pm8000 (red LED, IP: 10.30.14.245)
  - pm5000maqueta (yellow LED, IP: 10.30.14.247)
  - pm5000frio (red LED, IP: 10.30.14.249)
  - plc (red LED, IP: 10.30.14.243)
  - pasarela1 (green LED, IP: 10.30.14.248)
  - pasarela2 (green LED, IP: 10.30.14.246)

Ilustración 3.6 Página principal de la aplicación web

- **Página de equipo:** Ofrece una vista general del equipo en cuestión, y también se identifican una serie de secciones diferenciadas:



- 1) Una sección donde se muestra una imagen que representa al equipo en cuestión, junto con su dirección IP, nombre particular, estado, y un menú basado en una serie de botones con los que el usuario podrá interactuar:
  - Se habilita un botón para acceder a la página de configuración del equipo.
  - Se habilita un botón para acceder a la página de selección de los registros deseados.
  - Se habilitan dos botones cuya finalidad es ordenar la creación o destrucción de un hilo asociado al equipo en el sistema de adquisición de datos.
- 2) Una sección en la parte superior derecha que simplemente se corresponde con un menú basado en botones a disposición del usuario, mediante el cual, el mismo puede:
  - Volver directamente a la página principal
  - Volver a esta misma página
  - Volver justo a la página accedida anteriormente a la actual
  - Eliminar el equipo actual y retornar a la página principal

Este menú se arrastrará a lo largo de todas las páginas de la aplicación web.

- 3) Una sección donde simplemente se habilita un texto mediante el cual se informa del propósito de la página en la cual se encuentra el usuario. Esta sección también será arrastrada a lo largo de todas las páginas de la aplicación web, y en aquellas páginas en las que se produzca una parametrización, configuración, cargado de datos... ofrecerá información referente a los mismos para permitir al usuario monitorizar la correcta realización de las distintas actividades que lleve a cabo.

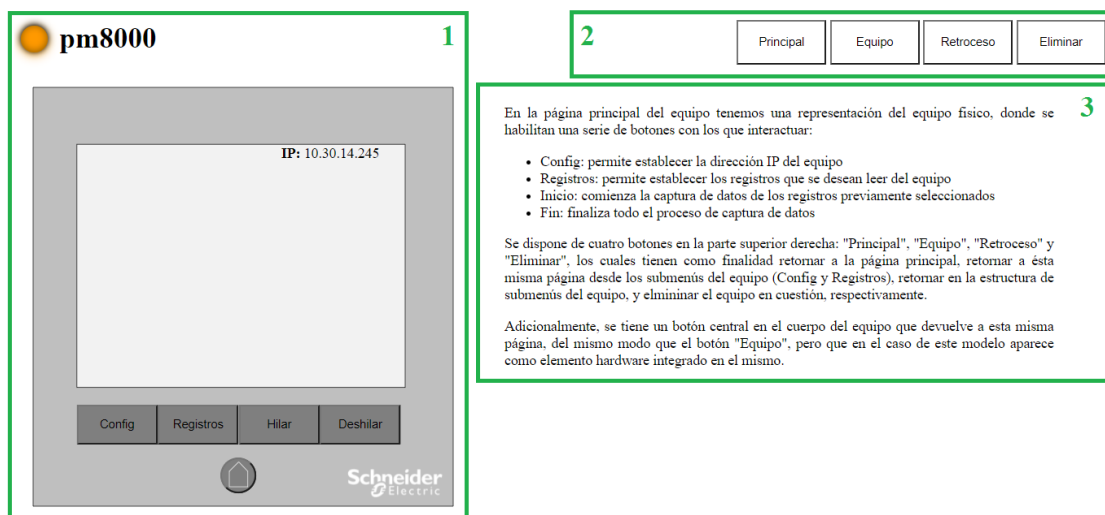


Ilustración 3.7 Página básica de un equipo



- **Página de configuración:** Ofrece la posibilidad de redefinir la dirección IP del equipo (no se aceptan direcciones repetidas), el nombre del equipo (no se aceptan nombres repetidos) y el intervalo que se desea establecer para la adquisición de datos, además de mostrar la sección informativa anteriormente mencionada, donde se ofrece información referente a la página en cuestión.

 pm8000

Principal	Equipo	Retroceso	Eliminar
-----------	--------	-----------	----------

En este submenú se permite hacer una redefinición de:

- Nombre del equipo: Debe ser único
- Dirección IP: Debe ser única
- Intervalo de captura de datos

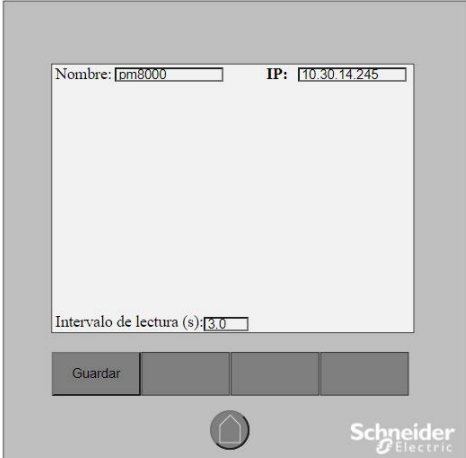


Ilustración 3.8 Página de configuración del equipo

- **Página de selección de registros:** Ofrece la posibilidad de seleccionar los registros que se desean leer, mediante una interfaz para que el usuario interactúe con ella:
  - 1) Se permite hacer la definición de un rango de registros (acotado mediante la definición del extremo superior e inferior del propio rango), o por el contrario seleccionar un único registro.
  - 2) Se pone a disposición del usuario un menú basado en botones para seleccionar una acción a realizar con el rango y/o registro definidos:
    - Añadir: Para cargar el rango y/o registros seleccionados y volver a la misma página para hacer una nueva definición.
    - Eliminar: Para borrar el rango y/o registros seleccionados y volver a la misma página para hacer una nueva definición.
    - Borrar todos: Para eliminar la totalidad de registros previamente definidos.
    - Guardar: Para cargar el rango y/o registros seleccionados y volver a la página básica del equipo.



- 3) Se dispone también de la sección informativa anteriormente mencionada, donde se mostrarán los registros únicos y rangos de registros que ya se han seleccionado con anterioridad para que sean leídos por el sistema de adquisición de datos.

**pm8000**

Principal Equipo Retroceso Eliminar

1

Rango:

Registro:

2

Añadir   Eliminar   Borrar todos   Guardar

3

En este submenú se realiza la selección de los registros ModBus que se desean leer, del siguiente modo:

- Rango: permite seleccionar un rango de registros de la forma [Menor; Mayor]
- Registro: permite seleccionar un único registro individual

La interfaz del equipo tiene la siguiente función:

- Añadir: almacena el nuevo rango y/o registro individual parametrizado, retornando a este mismo directorio para parametrizar nuevamente otro conjunto
- Eliminar: elimina el rango y/o registro individual parametrizado, si previamente fue cargado en memoria
- Borrar todos: elimina todos los registros en memoria
- Guardar: almacena el nuevo rango y/o registro individual parametrizado, retornando al menú principal del equipo

**Registros en memoria:**

- Rangos: [3000, 3004]
- Registros: 3006 , 3008

Ilustración 3.9 Página estándar de parametrización de los registros

Cabe destacar que ésta es la descripción de la página estándar, la cual se implementa para los equipos de las familias PM5XXX, PM8XXX y en las Power Tags. En el caso de los PLCs, debido a que la selección de los registros que se desean leer se realiza de manera automática por identificación de las marcas de memoria, esta página únicamente ofrecerá :

- 1) Un pequeño menú para realizar el cargado de una nueva lista de variables o la eliminación de la ya existente.
- 2) La sección informativa ya mencionada, donde se muestra una representación de la actual lista de variables cargada.

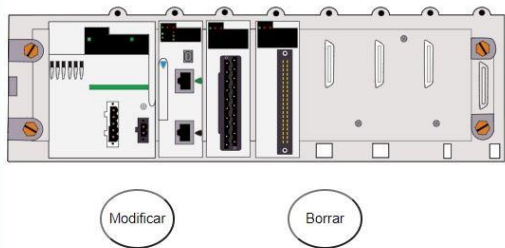
**plc**

Principal Equipo Retroceso Eliminar

1

Lista actual: [/media/variablelists/PLCVariableList\\_2uKHka.xlsx](#)

Nueva lista de variables: Seleccionar archivo Ningún archi... seleccionado



2

En este submenú se carga la lista de variables del PLC (equivalente a seleccionar registros en el resto de equipos).

**Lista actual:**

	Description	Register	Data Type
0	M_DO2_Electrovalvula_FY22	%MW32.0	BOOL
1	DI3_AltoNivelD04_LSH22	%I0.1.2	EBOOL
2	DI4_BajoNivelD04_LSL22	%I0.1.3	EBOOL
3	M_AI6_PresionProceso_PT21	%MW20	REAL
4	M_AO4_ValvulaCaudalProceso_FV21	%MW46	REAL
5	M_AI5_CaudalProceso_FT21	%MW18	REAL
6	M_AI2_TemperaturaProceso_TT22	%MW12	REAL
7	Man_Auto_PID	%MW102	REAL
8	DO2_Electrovalvula_FY22	%Q0.1.17	EBOOL

Ilustración 3.10 Página de registros correspondiente a los PLCs



La existencia de las pasarelas y Power Tags requiere abordar una solución particular para las mismas:

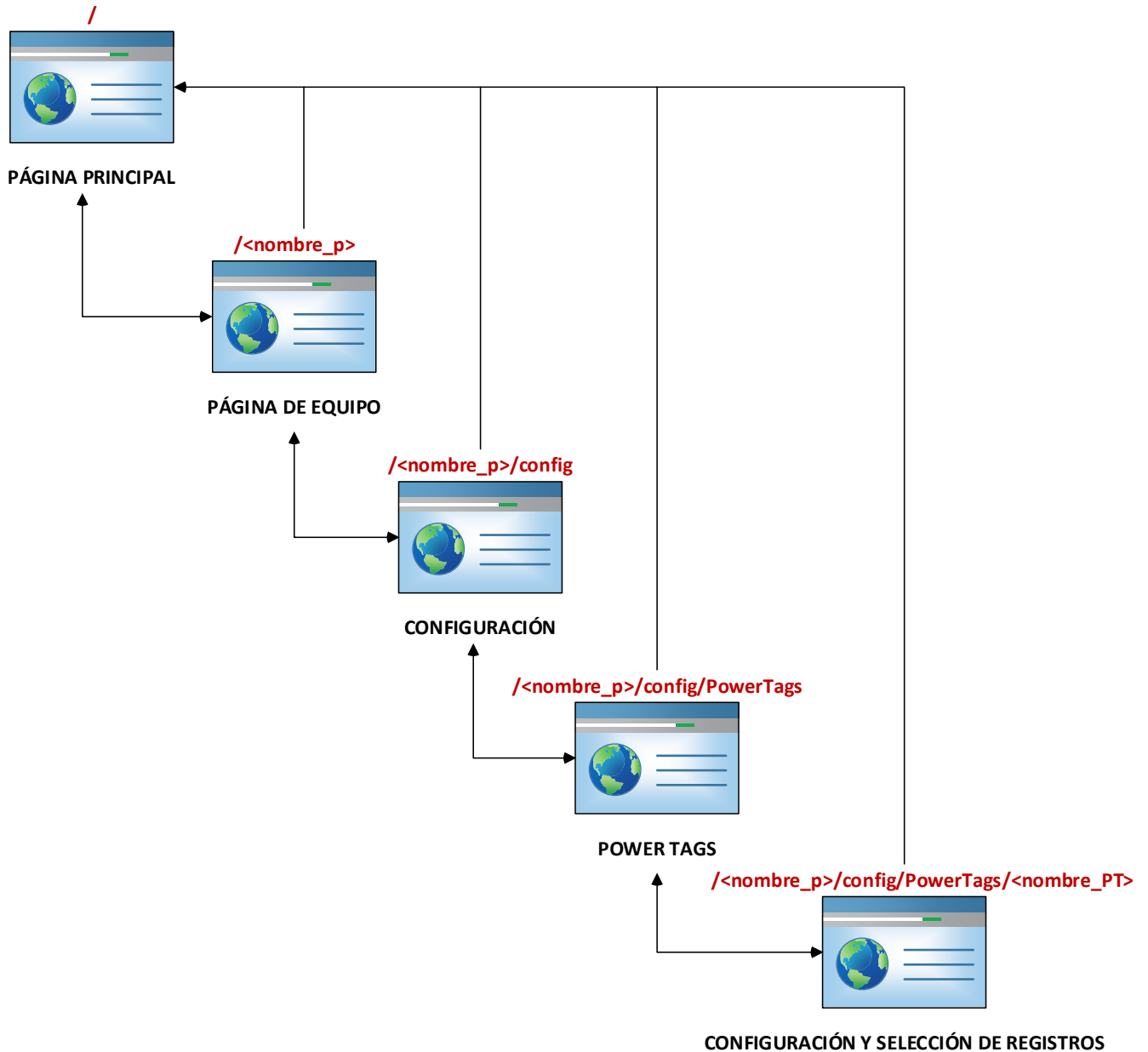


Ilustración 3.11 Estructura particular para dar soporte a pasarelas y Power Tags

Como se puede observar, la raíz de la estructura se mantiene, aunque en la página de equipo de las pasarelas, la opción para acceder a la página de parametrización de registros ya no estará disponible, puesto que los registros que se desean leer se corresponden con las Power Tags colgadas de la pasarela.

La función del resto de páginas representadas en la ilustración se detalla a continuación:



- **Página de configuración:** Además de la funcionalidad ya descrita en la estructura básica, esta página ofrecerá un botón mediante el cual el usuario podrá acceder a una página donde definir las Power Tags colgadas de la pasarela en cuestión.

## pasarela1



En este submenú, además de poderse establecer/modificar la dirección IP y el intervalo de lectura de datos, también se habilita el acceso a un nuevo submenú que permite añadir referencias a las PowerTags conectadas a la periferia, con el objetivo de establecer posteriormente los registros de las mismas que se desean leer.

El botón "Guardar" registra la dirección IP y el intervalo de lectura seleccionados, para posteriormente retornamos al menú principal del equipo, mientras que el botón "PowerTags" es el encargado de redireccionarnos al menú donde se realiza la definición de las mismas.

Ilustración 3.12 Página específica de configuración de las pasarelas

- **Página de Power Tags:** La página ofrece un pequeño menú con el cual definir nuevas Power Tags mediante la asignación de un nombre (debe ser único dentro de la pasarela) y una ID MODBUS (debe ser único dentro de la pasarela). Las Power Tags definidas se irán mostrando en pantalla de la misma manera que se mostraban los equipos en la pantalla principal de la aplicación, y el usuario podrá acceder a las mismas clicando sobre la imagen que las representa.

## pasarela1

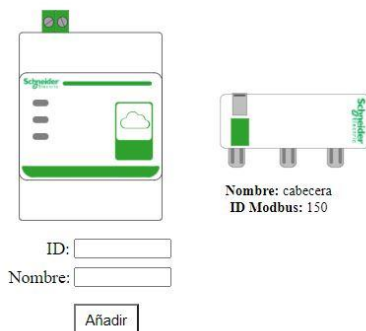


Ilustración 3.13 Página de adición y acceso a las Power Tags



- **Página de configuración y selección de registros:** Al hacer clic sobre una Power Tag determinada, se abre la actual página, la cual tiene la finalidad tanto de permitir la redefinición del nombre e ID de la Power Tag, así como ofrecer la posibilidad de modificar los registros seleccionados para su lectura, funcionando entonces la página como una combinación de la página de configuración y la página de selección de registros de la estructura básica.



Nombre:   
Modbus ID:



Rango:    
Registro:



En este submenú se realiza la selección de los registros ModBus que se desean leer, del siguiente modo:

- Rango: permite seleccionar un rango de registros de la forma [Menor, Mayor]
- Registro: permite seleccionar un único registro individual

La interfaz del equipo tiene la siguiente función:

- Añadir: almacena el nuevo rango y/o registro individual parametrizado, retornando a este mismo directorio para parametrizar nuevamente otro conjunto
- Eliminar: elimina el rango y/o registro individual parametrizado, si previamente fue cargado en memoria
- Borrar todos: elimina todos los registros en memoria
- Guardar: almacena el nuevo rango y/o registro individual parametrizado, retornando al menú principal del equipo

**Registros en memoria:**

- Rangos: [3000, 3004]
- Registros: Sin resultados

Ilustración 3.14 Página de configuración de las Power Tags y parametrización de registros

### 3.3.3 Interfaces de comunicación desarrolladas

Una vez el usuario ha definido, configurado y parametrizado correctamente los equipos mediante la aplicación web, estará listo para comunicarse con el sistema multihilo, haciendo uso del menú habilitado a tal fin y del que dispone en la página principal de la interfaz web, como se muestra en la *Ilustración 3.6 Página principal de la aplicación web*. A la hora de abordar la problemática asociada a esta comunicación entre dos recursos claramente diferenciados, se plantearon básicamente dos alternativas:

- Desarrollar un sistema monolítico y perfectamente integrado, donde Django se encargue de la comunicación con el sistema multihilo, haciendo uso de las herramientas y funcionalidades propietarias de Django y que simplifican el desarrollo.
- Elaborar una solución que utilice tecnologías estándar implementables únicamente haciendo uso del lenguaje Python y módulos multipropósito, sin necesidad de depender expresamente de un *framework* cuyo propósito es otro diferente, de tal modo que se prevea una simplificación de la integración del sistema multihilo con otro tipo de interfaces de usuario o recursos de otro tipo.

Finalmente se ha optado por la segunda solución, puesto que, pese a resultar en un mayor esfuerzo en cuanto a desarrollo se refiere, coincide mejor con el objetivo original de implementar el sistema multihilo como un servicio, contra el que el usuario de manera



esporádica pueda conectarse para realizar una configuración del mismo, y exceptuando estas ocasiones puntuales, el servicio corra de manera indefinida desarrollando su labor, como un recurso totalmente independiente.

Como se puede observar en la *Ilustración 3.2 Arquitectura conceptual de la herramienta*, el sistema multihilo se comunica con el *backend* de Django mediante dos interfaces (asociada cada una a los recursos anteriormente mencionados), y cuyo objetivo principal es racionalizar el flujo de datos a intercambiar entre ambos recursos, asegurando que el formato de los mismos es uno claramente definido. Estas dos interfaces se implementan mediante dos clases diferentes, denominadas *BackendInterface* y *BackgroundInterface*, que reúnen las siguientes características:

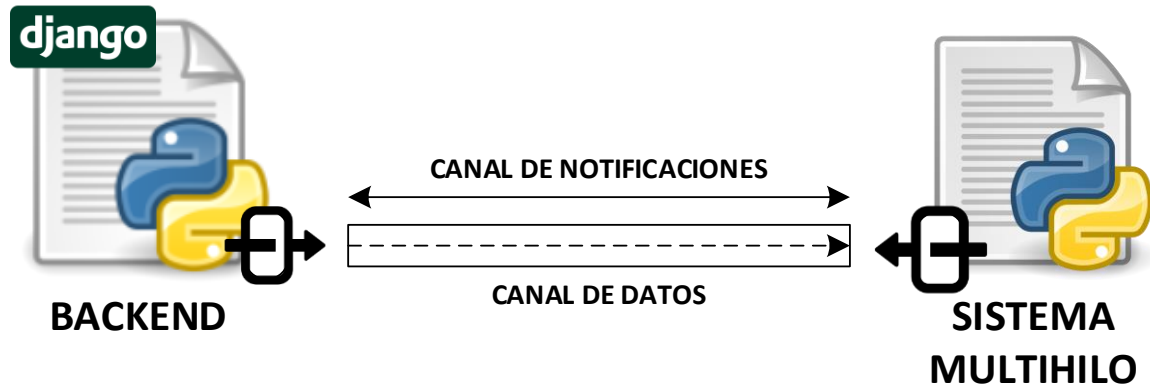
- **BackgroundInterface:** Se trata de la interfaz que se implementa en el sistema multihilo, y que da soporte al mismo en la comunicación con el *Backend*. Sus funciones se pueden resumir en las siguientes:
  - Levanta un servidor de objetos proxy al cual se conectará la interfaz del *Backend*. El objetivo de este servidor es facilitar una serie de objetos de comunicación a la interfaz de *Backend*, habilitando así el envío de datos entre diferentes procesos, e incluso diferentes máquinas de la red.
  - Permanece a la escucha de nuevos datos enviados por parte del *Backend*.
  - Una vez recibidos los datos y ejecutadas las actividades que se le ordenen, enviará una notificación de vuelta al *Backend* para ponerlo en conocimiento de su estado.
- **BackendInterface:** Como su nombre indica, se trata de la interfaz que se implementa en el lado del *Backend*. Sus funciones se resumen en las siguientes:
  - En ella se define el formato genérico de los datos que se pretenden enviar al sistema multihilo, con sus diferentes variantes, y con el objetivo de evitar el envío de datos no previstos que pudiesen producir algún problema en el correcto funcionamiento del sistema multihilo.
  - Establece conexión con el servidor de objetos proxy de la interfaz de *Background* para solicitar los objetos que dan soporte a los canales de comunicación.
  - Una vez comprobado que los datos que se desean enviar se ajustan al formato de datos esperado, se realiza el envío de los mismos.
  - Espera la respuesta del *Background* informando del recibimiento de los datos, y notificando el estado del mismo.

Los tipos de datos intercambiados entre ambas interfaces serán, por un lado, una serie de comandos y notificaciones, tanto para identificar lo que se debe hacer con los datos recibidos en el sistema multihilo, como para conocer el estado de dicho sistema en el *Backend*; mientras que, por otro lado, se enviarán todos los datos necesarios para realizar configuraciones y parametrizaciones desde el *Backend* hasta el sistema multihilo. Por lo tanto, se han habilitado dos canales: uno bidireccional para el envío de pequeñas cadenas





de texto, mientras que otro será unidireccional y por el circularán flujos de datos más complejos. La estructura conceptual de las comunicaciones entre interfaces se muestra a continuación:



*Ilustración 3.15 Funcionamiento de las interfaces de comunicación entre recursos*

Existen otras herramientas que resuelven la comunicación y son bien conocidas, como por ejemplo *Celery* [40], [41], que además está especialmente orientado a la integración multiservidor, aprovechando los recursos ociosos de diferentes máquinas para resolver las tareas que se tratan en el sistema multitarea, sin embargo, y debido al grado de complejidad del presente proyecto, se ha optado por no implementar actualmente esta solución, puesto que, pese a su gran potencial, constituye un servicio completo de cierta complejidad, que se ha de mantener.

Una vez detallados los aspectos principales de la herramienta desarrollada, así como discutidas las principales alternativas a las soluciones finalmente implementadas, se desarrollarán una serie de experimentos para mostrar los resultados obtenidos de la operación del sistema, analizándolos para verificar el cumplimiento de los objetivos inicialmente marcados.



## 4 Experimentos y resultados

Los experimentos realizados para corroborar el cumplimiento de los objetivos establecidos, así como para evaluar los resultados generales de la herramienta, harán uso de una de las maquetas de 4 variables de las que dispone el Grupo de Investigación SUPPRESS, por lo que, previamente a desarrollar la ejecución y resultados de los experimentos, se realizará una presentación del sistema físico.

### 4.1 Presentación del sistema físico

El Grupo de Investigación SUPPRESS de la Universidad de León ha diseñado y construido maquetas didácticas de control de procesos, que se encuentran en el Aula E2 del Edificio Tecnológico II (Universidad de León). Se trata de células de ensayo de estrategias de control dotadas de instrumentación industrial inteligente para medida y control de caudal, nivel, presión y temperatura.



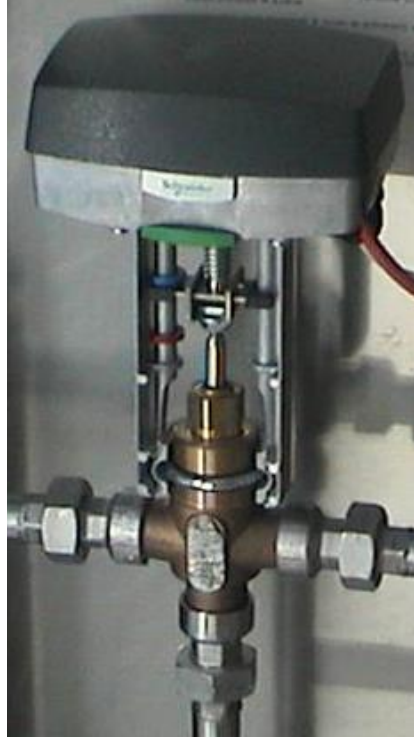
*Ilustración 4.1 Maqueta de 4 variables empleada en la experimentación[42]*

En dicha aula se dispone de un total de 3 maquetas idénticas a la anteriormente mostrada, de las cuales, 2 permanecerán inactivas durante el experimento, lo que se traduce en un consumo energético mínimo, asociado a medidores y demás consumos pasivos de los que disponen las mencionadas maquetas.

Los experimentos involucrarán algunos de los equipos de los cuales éstas maquetas disponen, como son:



- Las válvulas de tres vías encargadas de regular el caudal circulado.



*Ilustración 4.2 Válvula de 3 vías existente en la maqueta de 4 variables*

- La electroválvula que comunica los tanques de proceso.



*Ilustración 4.3 Electroválvula entre tanques de proceso*

- La bomba de recirculación del caudal de proceso



*Ilustración 4.4 Bomba de recirculación*

El esquema eléctrico conceptual del laboratorio se puede apreciar en la imagen mostrada a continuación, aunque teniendo en cuenta las anotaciones anteriormente realizadas con respecto al estado actual del laboratorio, y que la máquina de frío tampoco se encuentra encendida para los experimentos en cuestión, dicho esquema se puede simplificar para facilitar la comprensión de las lecturas de datos realizadas, como se muestra seguidamente:

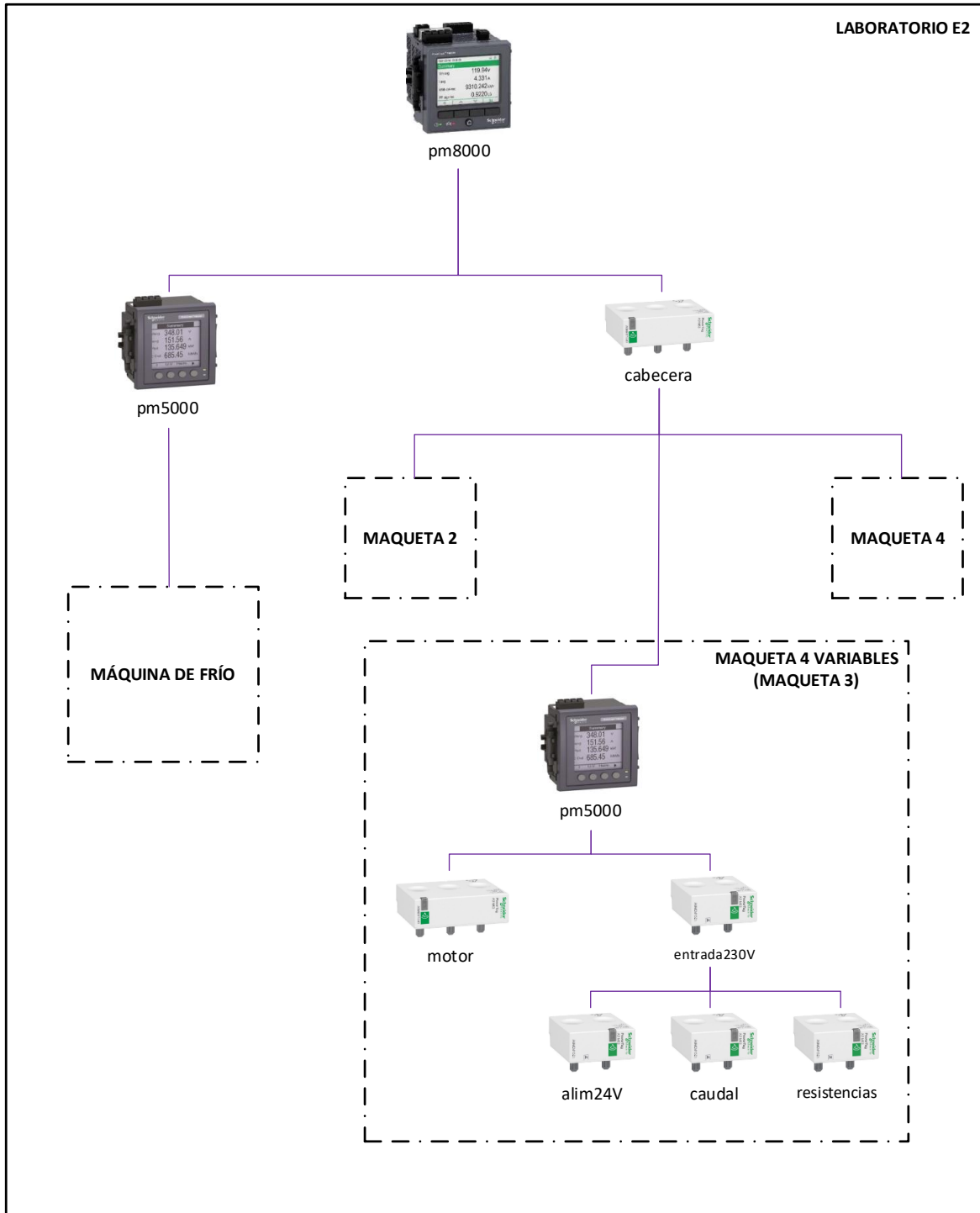


Ilustración 4.5 Esquema eléctrico completo

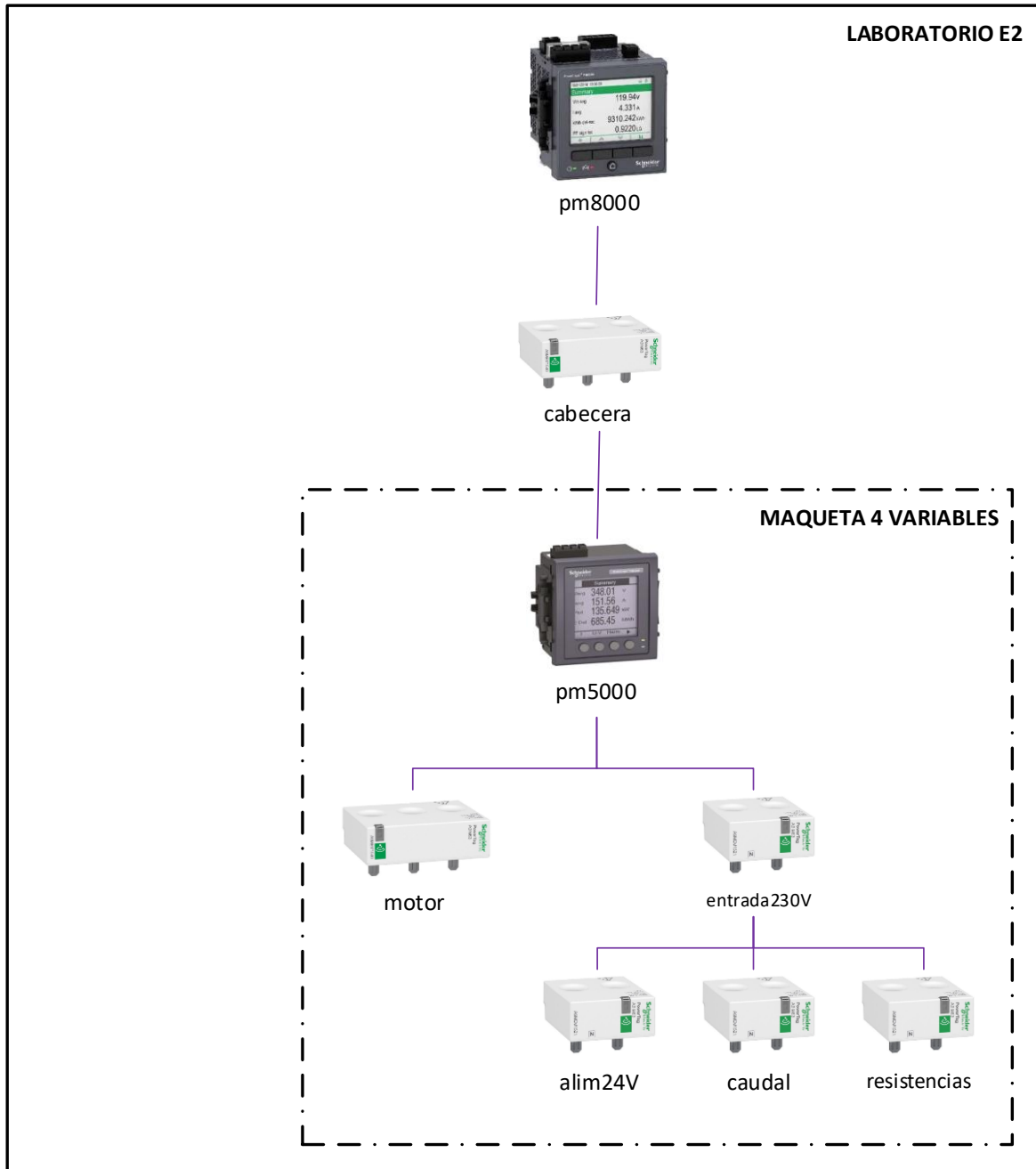


Ilustración 4.6 Esquema eléctrico simplificado



## 4.2 Experimento 1: Captura masiva de datos a elevada frecuencia

El primer experimento consistirá en hacer funcionar las válvulas de tres vías entre sus posiciones extremas un cierto número de veces. Seguidamente se realizará un arranque de la bomba de recirculación, controlada por un PID de nivel, y durante la operación de la misma se introducirá una perturbación (producida por la electroválvula).

Se capturarán datos tanto del PLC como del PM8000, PM5000, PowerTag de cabecera, PowerTag del motor de la bomba, PowerTag de la entrada de 230V y PowerTag de la alimentación de 24V, y se analizarán los datos mediante comparación. Todos los datos serán capturados con un intervalo de 2 segundos.

### 4.2.1 Representación gráfica de los datos del PLC

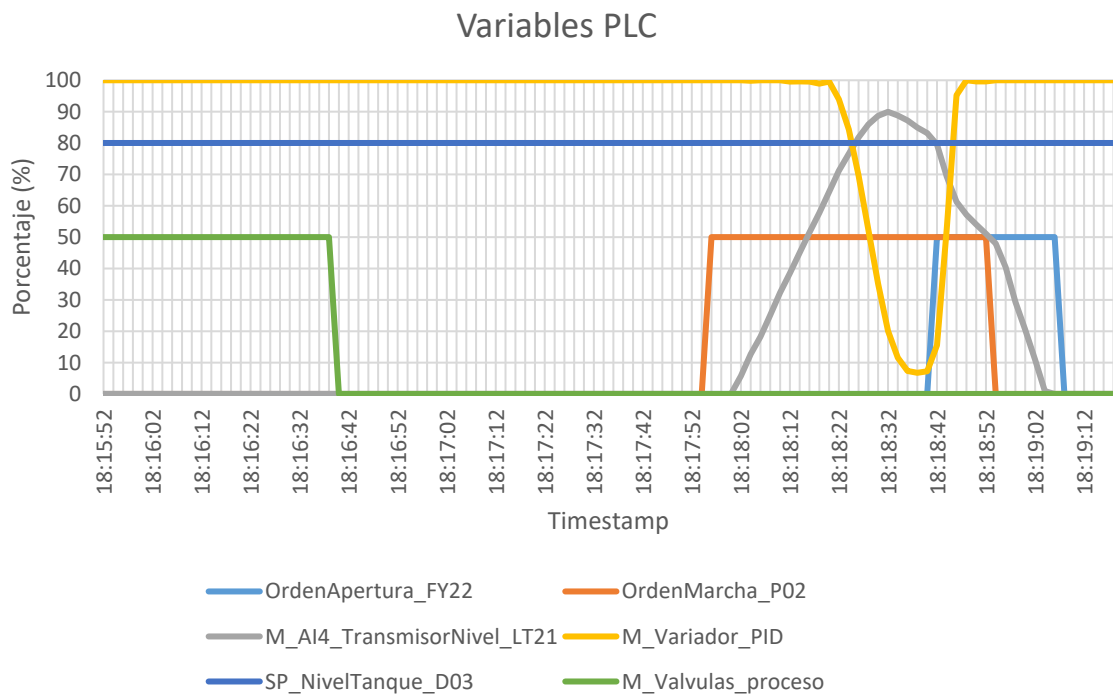


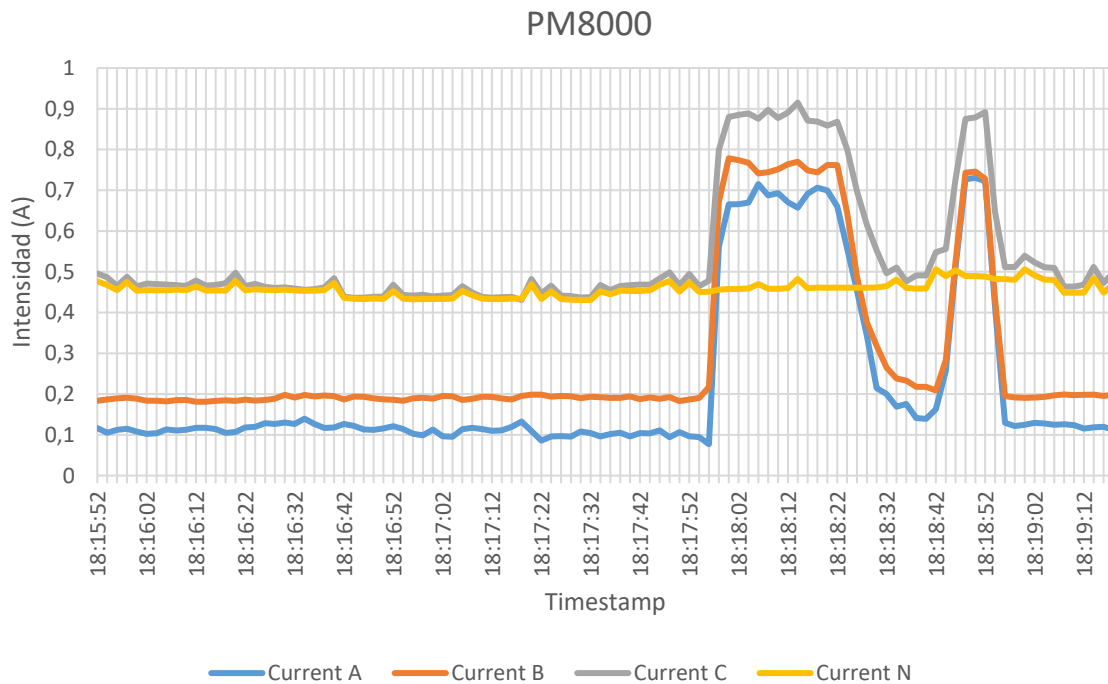
Ilustración 4.7 Datos recabados del PLC

Los datos del PLC cumplen la funcionalidad de representar qué se ha hecho sobre las diferentes variables de control para llevar a cabo el experimento. Cabe mencionar que las variables *OrdenMarcha\_P02*, *OrdenApertura\_FY22* y *M\_Valvulas\_Proceso* muestran un valor de 50% sobre la gráfica anterior, pero dicho valor solamente tiene el objetivo de facilitar su observación, puesto que se tratan únicamente de señales booleanas.

Por último, cabe destacar que la variable *M\_Variador\_PID* únicamente afecta al proceso cuando se da la orden de marcha a la bomba P02, y que las variaciones de esta variable se reflejarán sobre el consumo eléctrico de la bomba, como veremos a continuación.



#### 4.2.2 Representación gráfica de los datos del PM8000



*Ilustración 4.8 Datos recabados del PM8000*

El PM8000, al encontrarse monitorizando la totalidad de consumos del laboratorio y de la máquina de frío, ofrece una lectura muy generalista, de la cual se puede extraer información de grandes consumos, pero los más pequeños pasarán inadvertidos prácticamente.

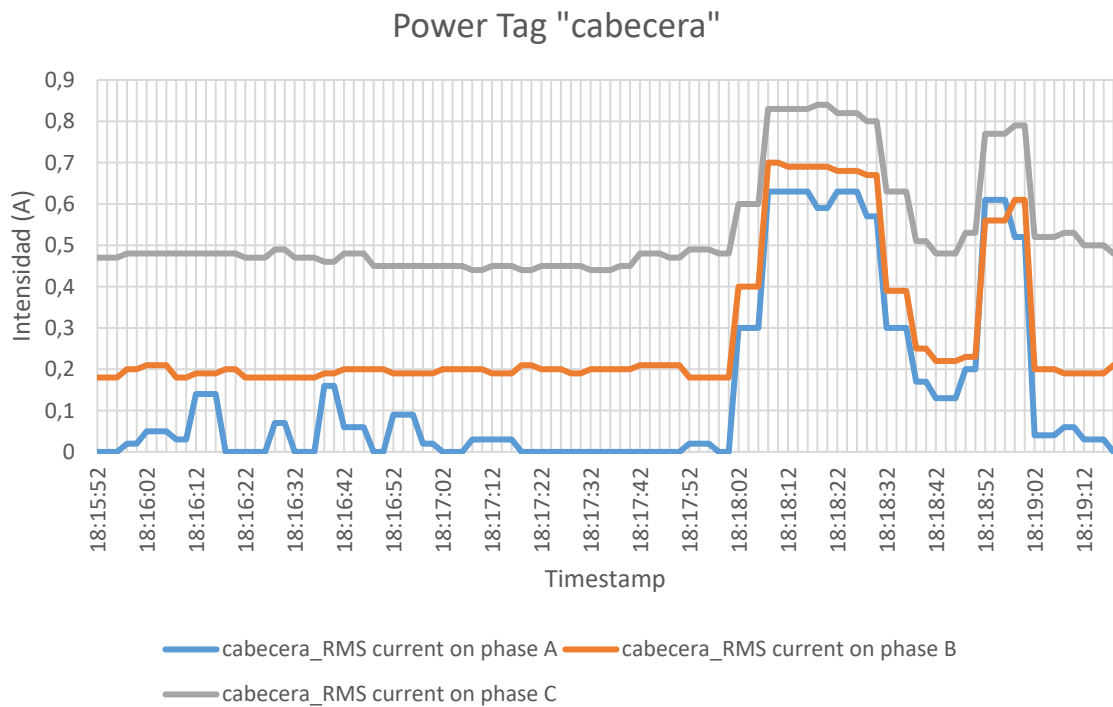
En la imagen anterior se puede apreciar:

- La mayor parte de elementos monofásicos y alimentados en corriente continua se encuentran alimentados mediante una única fase (C), debido al elevado desequilibrio que se aprecia en el sistema trifásico.
- Al permanecer todos los demás elementos del laboratorio, así como la máquina de frío, inactivos durante la operación de la maqueta, se puede apreciar perfectamente el consumo de la bomba, así como la influencia de la activación de la electroválvula, la cual produce un pico de consumo por parte de la bomba, que vuelve a trabajar a plena carga.
- El consumo introducido por las válvulas pasa completamente inadvertido.





### 4.2.3 Representación gráfica de los datos de la PowerTag de cabecera



*Ilustración 4.9 Datos recabados de la PowerTag de cabecera*

Como se puede observar, y debido a la inactividad de la máquina de frío y del resto de maquetas, los datos recabados de la PowerTag son muy similares a los recabados del PM8000. Se puede apreciar también que las lecturas ofrecidas por las PowerTag poseen una cantidad de ruido mucho más notoria que los PMs, debido a su principio de funcionamiento (sondas Hall).

Esta perturbación es muy notoria cuando menor es la variación de la corriente a medir, como se puede observar si comparamos la corriente en la fase A ofrecida por el PM8000 y por la PowerTag.

Las conclusiones extraídas y referentes al consumo coinciden con las ya mencionadas.



#### 4.2.4 Representación gráfica de los datos del PM5000

##### PM5000

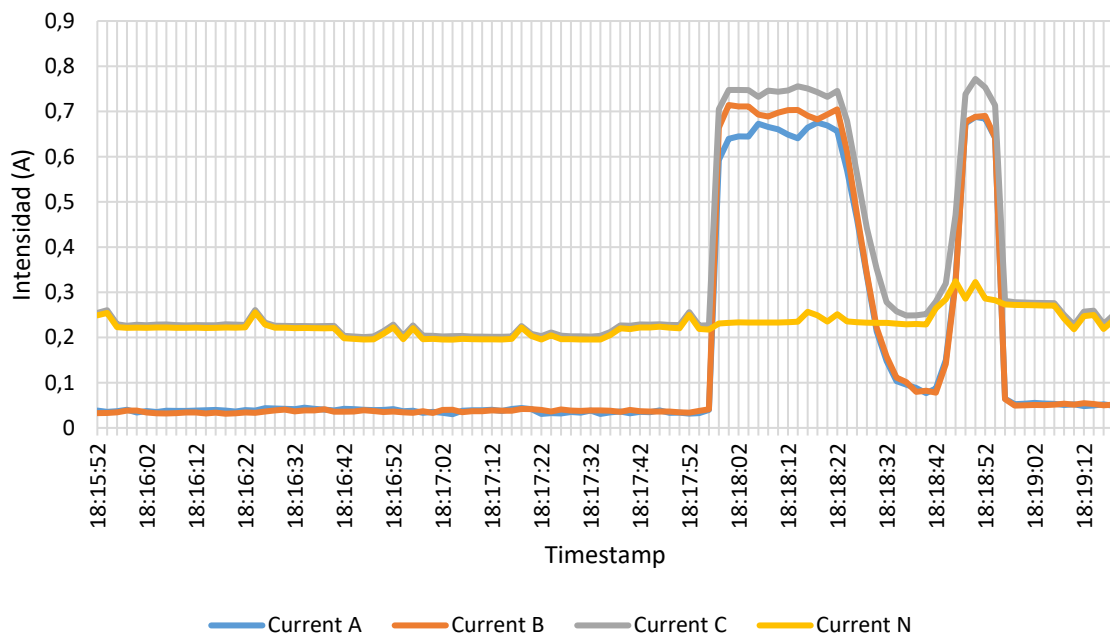


Ilustración 4.10 Datos recabados del PM5000

Se aprecia una gráfica muy similar a las anteriormente comentadas, sin embargo, se puede observar:

- Una menor distorsión armónica que en el caso de la gráfica más generalista del PM8000.
- Un desequilibrio de fases aún más acentuado, donde se confirma que la fase C se encuentra claramente desequilibrada porque las cargas monofásicas y de continua no han sido repartidas entre las tres fases.

Se puede extraer aún más información si analizamos el neutro y lo comparamos con la gráfica del PM8000:

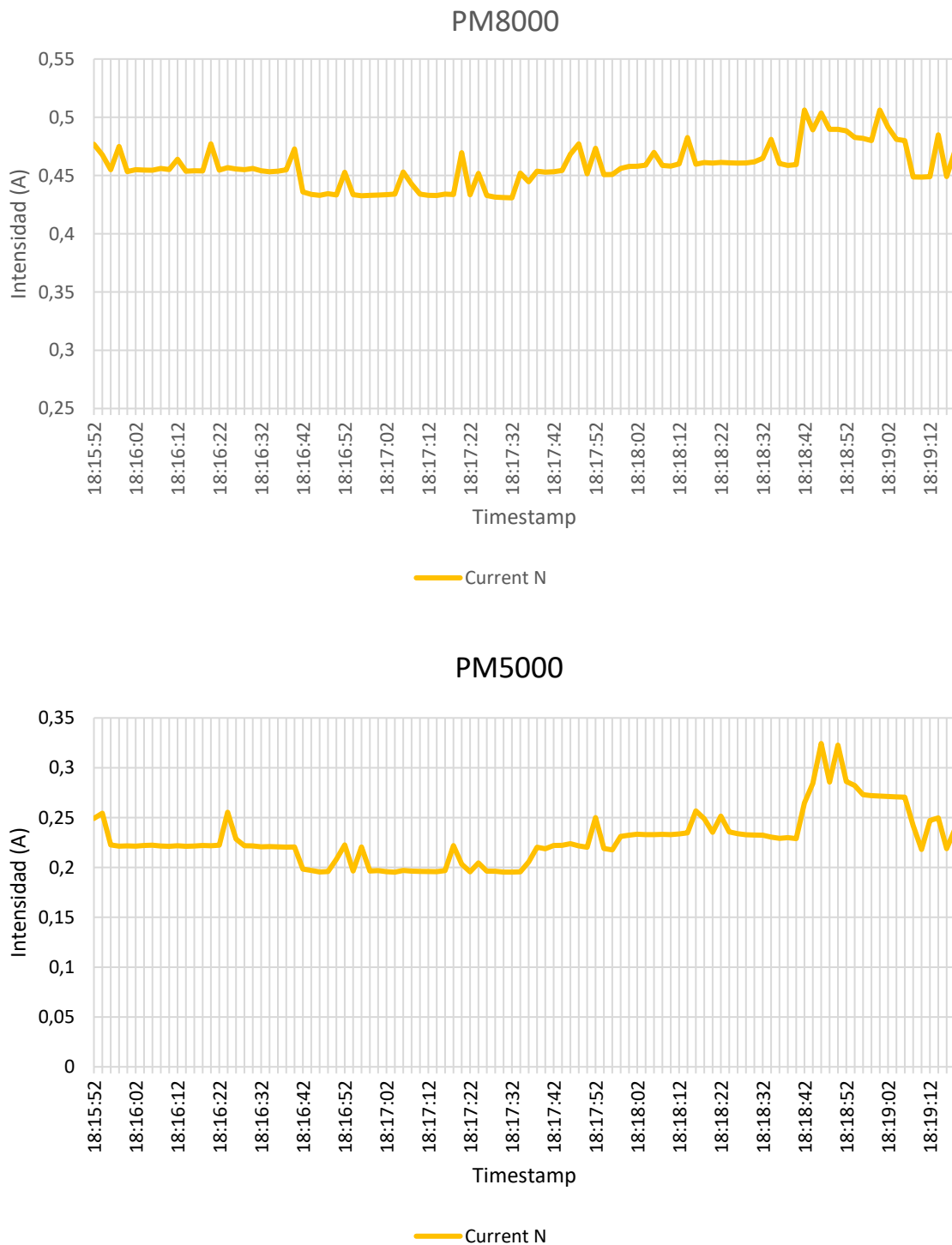


Ilustración 4.11 Comparativa entre la lectura del neutro del PM8000 y PM5000

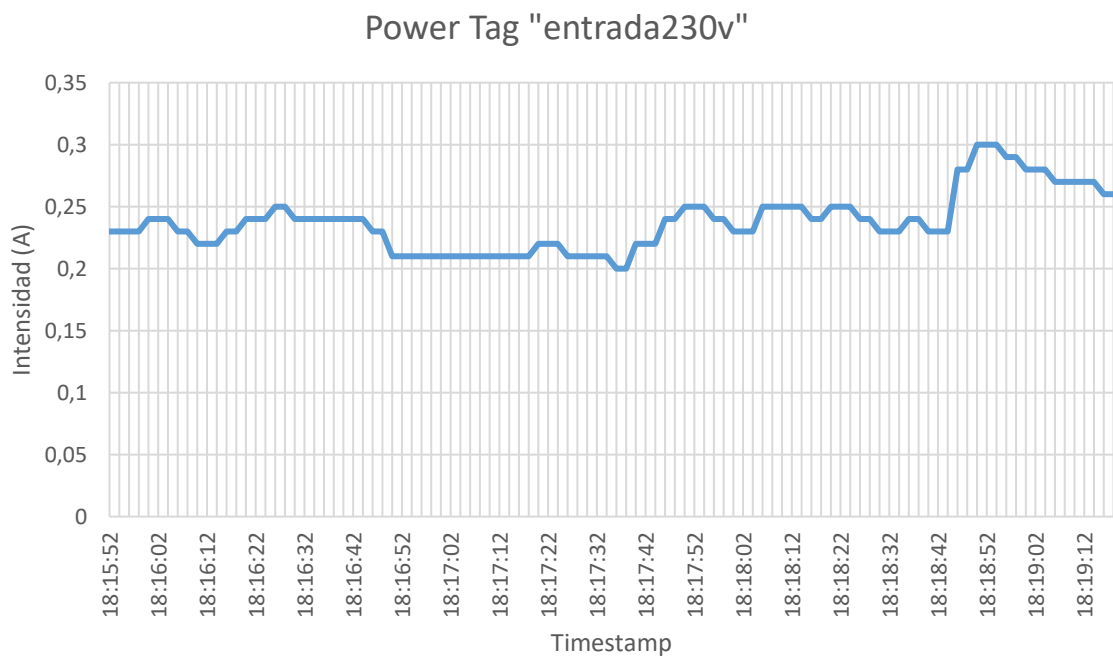
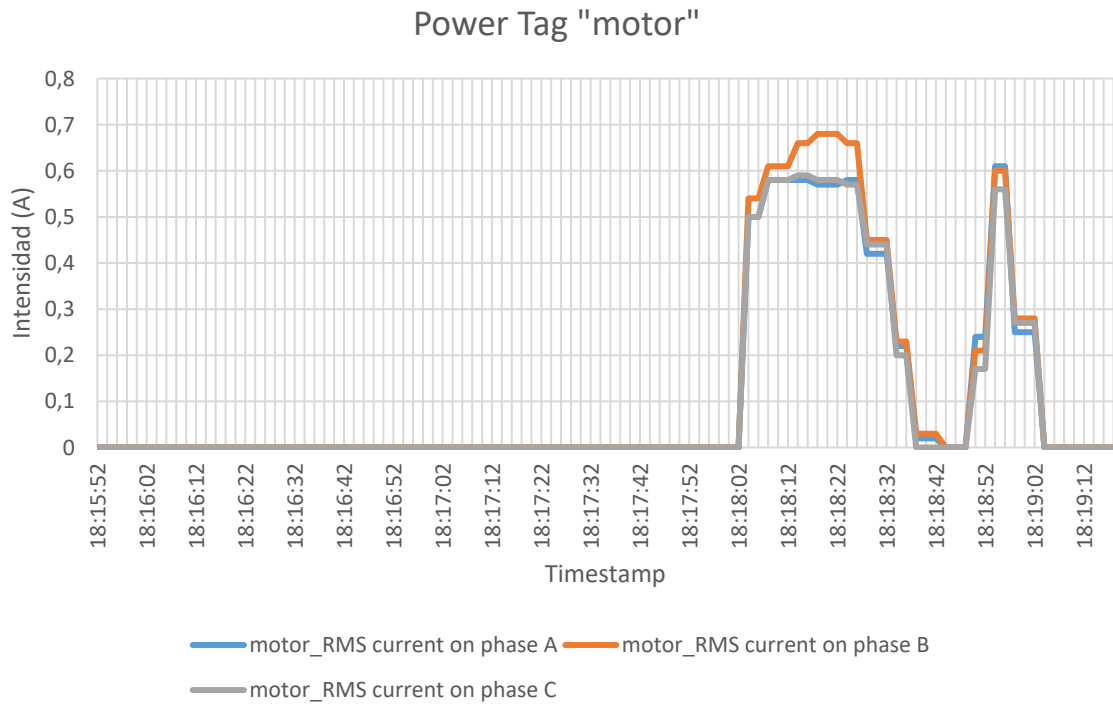
Como se puede observar en las gráficas anteriormente mostradas, el neutro acompaña a la fase C, exceptuando los picos de consumo de la bomba, los cuales no los refleja. Teniendo esto en cuenta y observando la imagen adjunta se puede apreciar que, en el PM5000, debido a una menor distorsión, se observa más claramente un comportamiento escalonado en el consumo, así como un pico de consumo en las muestras finales. Este comportamiento



escalonado es debido a las válvulas de tres vías, mientras que el pico en el consumo es debido a la activación de la electroválvula.

Esta observación se corrobora con las mediciones realizadas en las PowerTags aguas abajo del PM5000.

#### 4.2.5 Representación gráfica de los datos de las PowerTags del motor, entrada230V y alimentación de 24Vcc



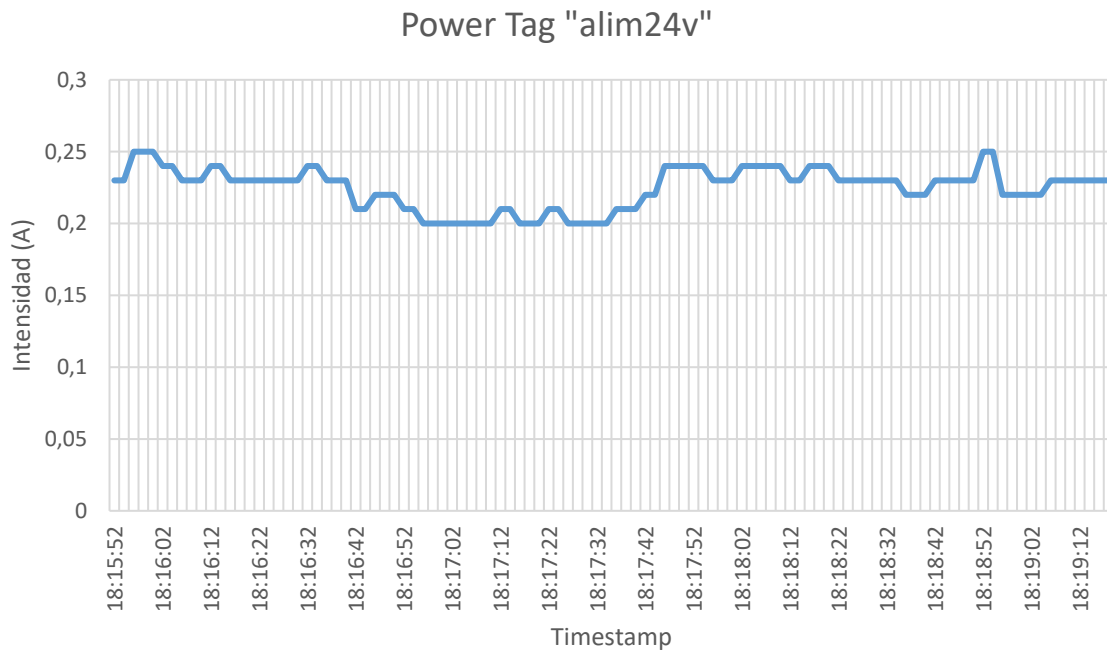


Ilustración 4.12 Datos recabados de las PowerTags motor, entrada230V y alimentación 24Vcc

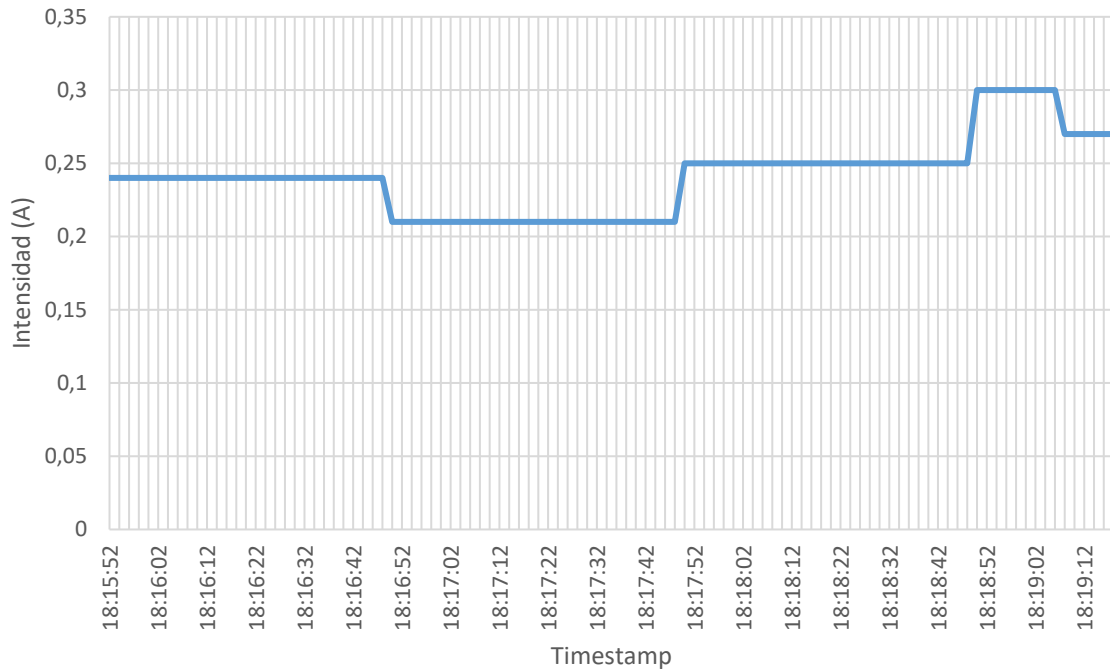
Las imágenes mostradas nos permiten identificar el desglose de consumos que ya se ha ido mencionando.

- En el caso de la PowerTag del motor, se puede apreciar el consumo de la bomba aislado respecto al resto de consumos, el cual se veía perfectamente reflejado en numerosas gráficas que ya hemos mencionado previamente.
- En el caso de las PowerTags de entrada230V y alimentación 24Vcc, se puede apreciar una gráfica muy similar a la que ya se había aislado en el PM5000, la cual evidencia lo que ya se había comentado y aún permanecía ligeramente enmascarado por el resto de consumos en la gráfica correspondiente al PM5000.
- Comparando las gráficas de la PowerTag de entrada230V y la PowerTag de alimentación 24Vcc, se puede comprobar que la electroválvula entre tanques únicamente hace un consumo monofásico, pero no de la alimentación de 24Vcc de la que dispone la maqueta, puesto que su consumo no aparece reflejado en esta PowerTag. De este modo, la PowerTag de alimentación 24Vcc representa de forma aún más clara el consumo realizado por válvulas de tres vías.

Con el objetivo de mejorar la visibilidad de los datos generados por las PowerTags, se ha realizado un postprocesado de los mismos, filtrando el ruido y distorsión introducido por los sensores de efecto Hall, y evidenciando claramente la tendencia de los datos capturados, como se muestra a continuación:



Power Tag "entrada230v"



Power Tag "alim24v"

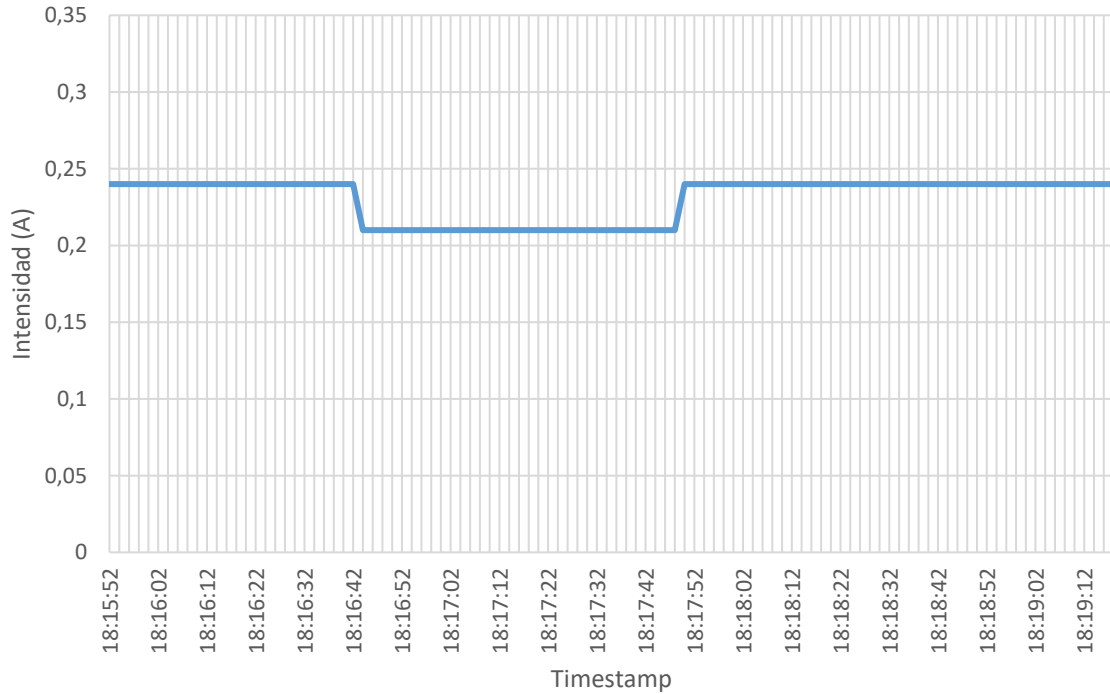


Ilustración 4.13 Gráficos filtrados de las PowerTags en cuestión

Como se puede observar, el *submetering* permite identificar estos consumos que se muestran ocultos frente a consumos más notorios y distorsiones de onda, y que aún en el PM5000 podrían pasar inadvertidos si se desconociese el comportamiento esperado.



### 4.3 Experimento 2: Comparativa entre el comportamiento de la herramienta y el comportamiento del software Power Monitoring Expert

El software Power Monitoring Expert limita la adquisición de datos a un intervalo mínimo de 1 minuto. Para evaluar el rendimiento que la herramienta presentará frente a este software, se plantea el siguiente experimento, el cual consistirá en la apertura y cierre en sucesivas ocasiones de la electroválvula entre tanques, así como el arranque puntual de la bomba.

Para realizar una comparativa, se programará la captura de datos provenientes del PM5000 a intervalos de 60 segundos (mismo comportamiento que los datos recabados con el software PME), mientras que, simultáneamente, se recabarán los datos de las PowerTags del motor de la bomba y de la entrada de 230V, a intervalos de 2 segundos, para comparar las lecturas del PM5000 con las de las PowerTags. A mayores, se ofrecen los datos recabados del PLC, con un intervalo de muestreo de 2 segundos, con el único propósito de identificar qué se está haciendo en cada instante, mediante el análisis de las variables de control.

#### 4.3.1 Representación gráfica de los datos del PLC

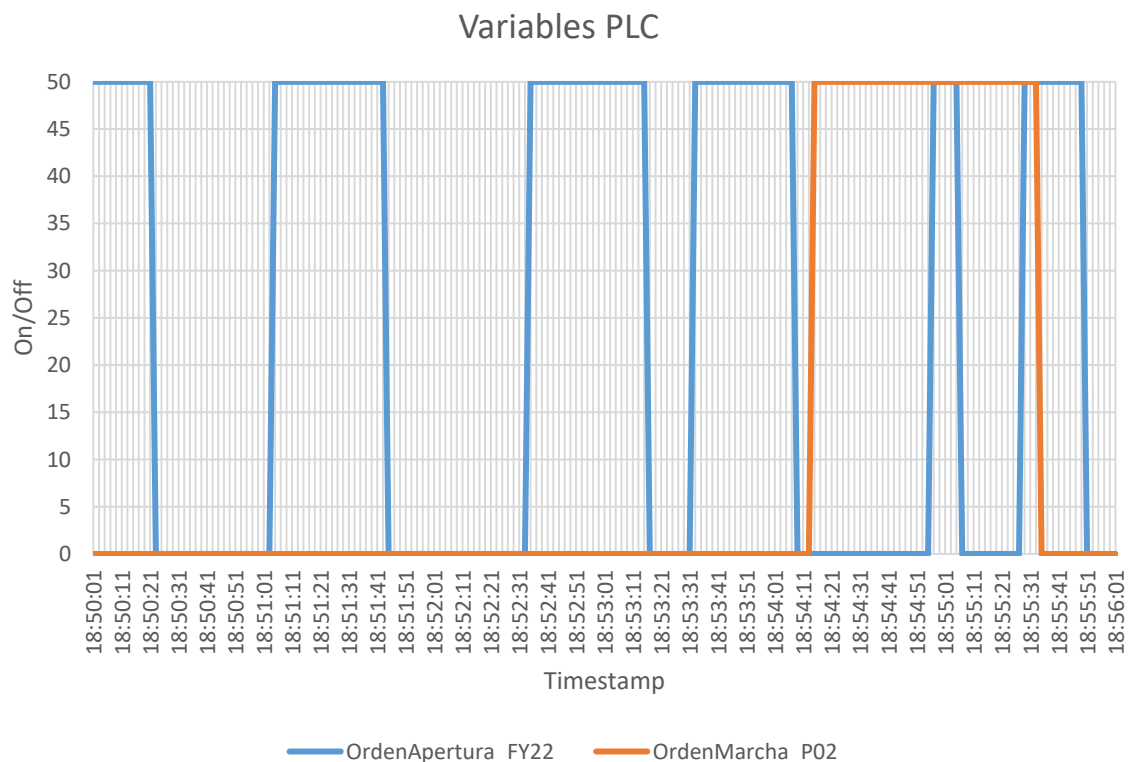


Ilustración 4.14 Datos recabados del PLC

En la gráfica adjunta se puede apreciar los instantes exactos en los que la bomba se encuentra funcionando, así como los instantes en los que la electroválvula se encuentra abierta, y en cuales se encuentra cerrada.



### 4.3.2 Representación gráfica de los datos del PM5000

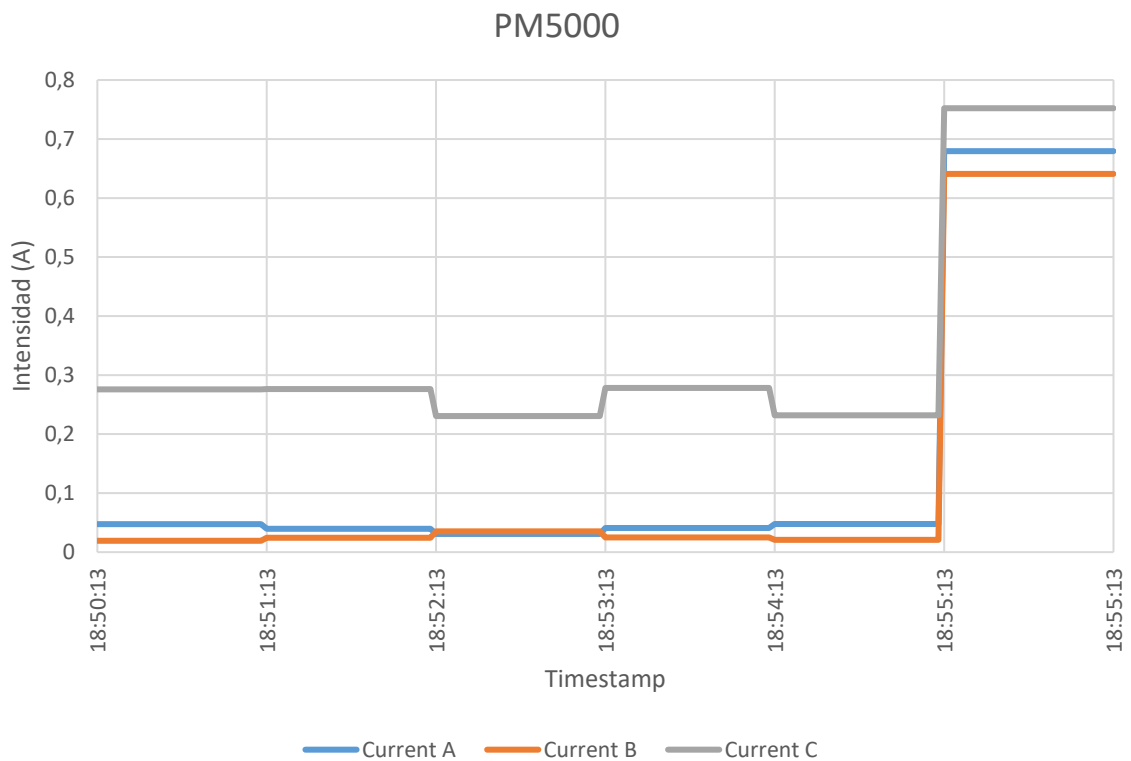


Ilustración 4.15 Datos recabados del PM5000

De la gráfica adjunta se deduce:

- Debido al intervalo de muestreo tan elevado, la información correspondiente a la distorsión presente en la onda se ha perdido por completo, por lo que no es posible realizar un análisis de ruido de la red.
- El comportamiento de la bobina deducible del consumo aquí mostrado no coincide con el comportamiento (ni con el consumo) real, puesto que durante la operación de la bomba se ha abierto la electroválvula, lo cual debería reflejarse como una perturbación de la misma.
- Los ciclos de apertura y cierre de la bomba no aparecen correctamente representados, es decir, el consumo identificado no se corresponde con el consumo realmente producido.
- Las ventajas del *submetering* se ven reducidas, puesto que, debido al elevado intervalo de captura de datos, las perturbaciones en el consumo producidas por elementos como la electroválvula durante la operación de la bomba, son eliminadas por completo.





### 4.3.3 Representación gráfica de los datos de las PowerTags del motor y de la entrada de 230V

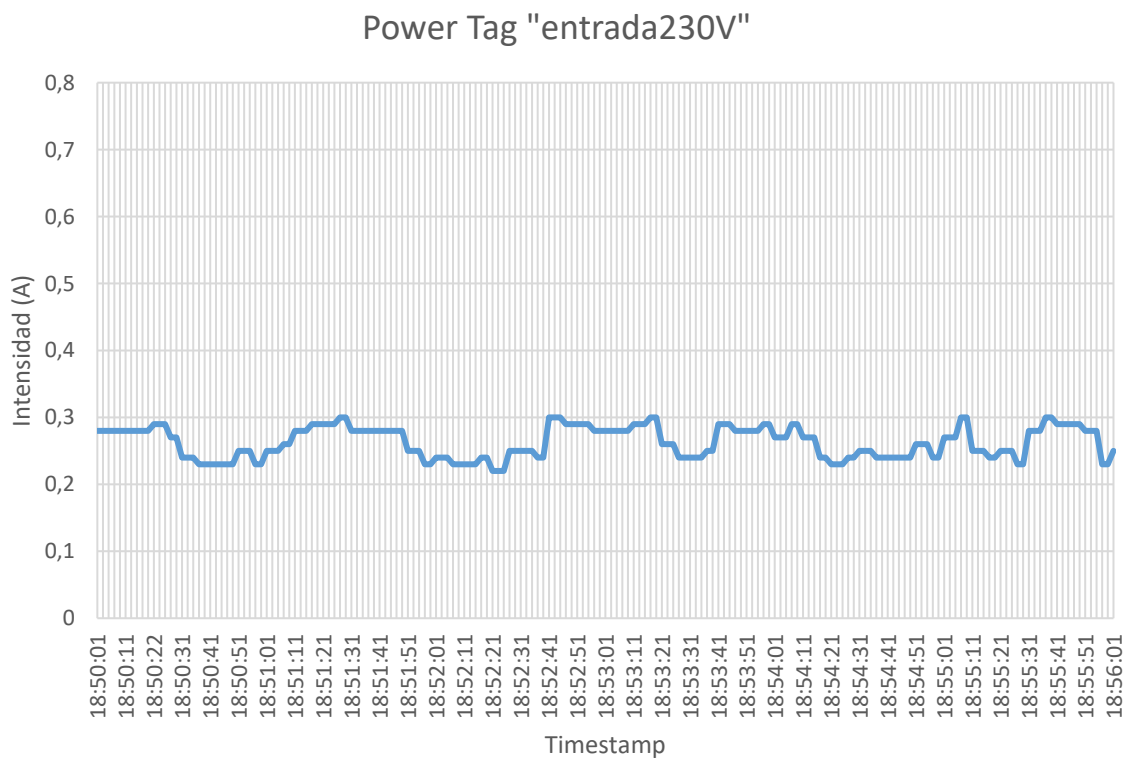
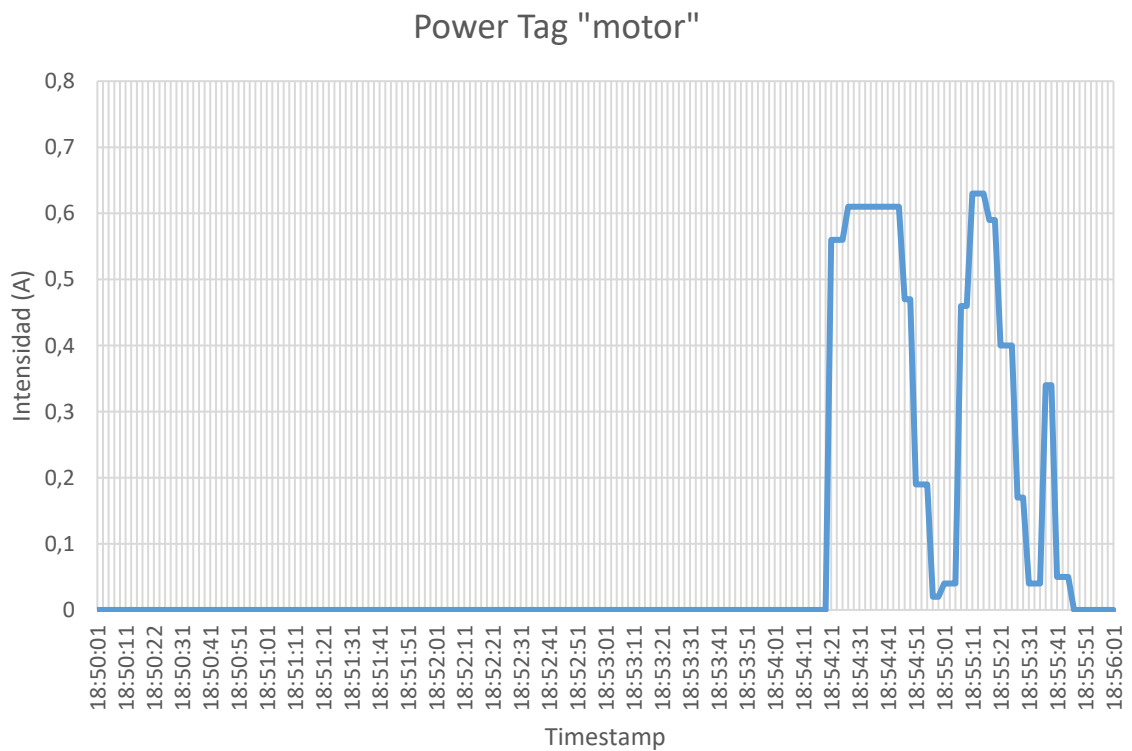


Ilustración 4.16 Datos recabados de las PowerTags del motor y de la entrada de 230V



Aunque en este caso es mucho más apreciable el comportamiento de la electroválvula dentro del consumo registrado por la PowerTag de entrada 230V, se ha realizado igualmente un postprocesado de los datos para su filtrado:

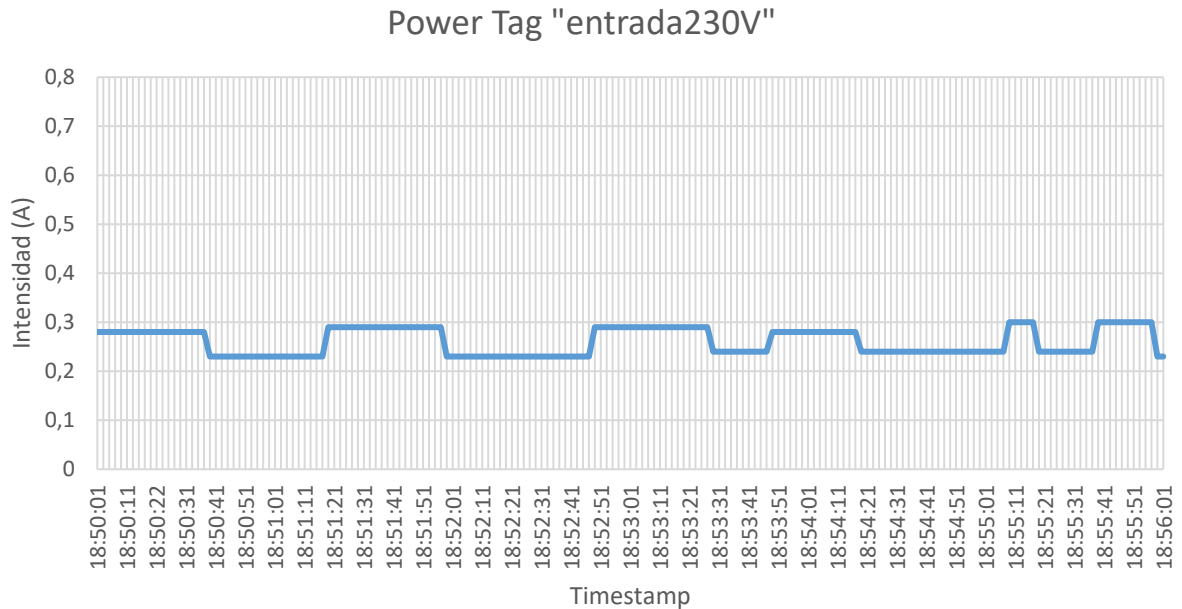


Ilustración 4.17 Datos filtrados de la PowerTag de entrada 230V

De las gráficas mostradas se puede deducir:

- Respecto a la gráfica correspondiente al PM5000, el consumo de la bomba se ve notoriamente mejor representado, quedando perfectamente reflejada la influencia de la electroválvula.
- Respecto a la gráfica correspondiente al PM5000, se representan la totalidad de ciclos de operación de la electroválvula, sin perderse ninguno como consecuencia del intervalo de captura de datos.
- Como ya se mencionó, las ventajas del *submetering* son evidentes en estos casos de elevadas frecuencias de captura de datos, puesto que, de otro modo, las perturbaciones apreciables en la gráfica de la PowerTag del motor o los ciclos más rápidos de apertura y cierre de la electroválvula, pasarían totalmente inadvertidos, falseándose el consumo de dichos dispositivos.



#### 4.4 Discusión de resultados

Los resultados derivados de los experimentos anteriores son básicamente los siguientes:

- Del primer experimento, se deriva la importancia del *Submetering*, la capacidad de la herramienta desarrollada para implementarlo eficazmente, la capacidad de la herramienta para soportar dispositivos multipropósito y la capacidad para combinar gestión energética con supervisión industrial que ofrece la herramienta, al capturar datos de diferentes fuentes.
- Del segundo experimento, se deriva que el objetivo fundamental que motiva el presente proyecto se ha alcanzado, puesto que la herramienta ofrece una serie de funcionalidades y un mejor comportamiento que las principales herramientas existentes en la actualidad.



## 5 Conclusiones

En virtud de los resultados obtenidos, se concluye que la herramienta desarrollada cumple los objetivos inicialmente impuestos, sirviendo como un elemento capaz de soportar mediciones masivas de datos en cortos intervalos de tiempo, lo que se traduce en su capacidad para hacer frente a sistemas complejos de *Submetering* permitiendo el aprovechamiento de las ventajas de los mismos, así como para evaluar cuestiones como la calidad del suministro eléctrico, mediante el análisis de los datos generados haciendo uso de la herramienta, lo cual, debido a los intervalos manejados por herramientas software actualmente disponibles en el mercado, no es una posibilidad. El diseño seguido para el desarrollo de la misma permite la posibilidad de realizar previsibles ampliaciones futuras, con el objetivo de dar soporte a diferentes tecnologías y estándares de comunicación industrial.

En general, se muestra como una herramienta que cumple con su objetivo principal de servir para la captura masiva de datos a altas frecuencias, gozando de una gran flexibilidad en dicha actividad, ya que ofrece una gran capacidad de configuración por parte del usuario, pero de un modo simple e intuitivo para el mismo.



## 6 Líneas futuras

Como líneas de investigación futura, se podrían definir las siguientes:

- Aumentar la cobertura de dispositivos soportados por el sistema actual, extendiéndose a diferentes fabricantes que, aun implementando las mismas tecnologías de comunicación empleadas, suponen un reto en la resolución de las particularidades que se vayan presentando en cada caso.
- Aumentar la cobertura de protocolos y estándares de comunicación soportados.
- Trabajar en mejorar la integración del sistema desarrollado con diferentes funcionalidades propias de la Industria 4.0, trabajando en arquitecturas orientadas a servicios donde cada servicio tiene una funcionalidad claramente definida.
- Estudiar la posibilidad de implementar el sistema desarrollado mediante otra implementación de Python, o analizar otras posibilidades para solventar el problema asociado con el GIL, que puede conllevar una limitación en sistemas con numerosos hilos de ejecución.
- Trabajar para reducir aún más los intervalos límite de captura de datos, abriéndose así a posibilidades como la reconstrucción de ondas eléctricas.



## 7 Bibliografía

- [1] M. Alonso-Rosa, A. Gil-de-Castro, R. Medina-Gracia, A. Moreno-Munoz, and E. Cañete-Carmona, “Novel internet of things platform for in-building power quality submetering,” *Applied Sciences (Switzerland)*, vol. 8, no. 8, 2018, doi: 10.3390/app8081320.
- [2] N. Iglesias and J. M. Miranda, “Las infraestructuras de recarga y el despegue del vehículo eléctrico,” *Observatorio Medioambiental*, vol. 18, pp. 57–85, 2015, [Online]. Disponible en: [http://dx.doi.org/10.5209/rev\\_OBMD.2015.v18.51285](http://dx.doi.org/10.5209/rev_OBMD.2015.v18.51285)
- [3] F. Arteaga, “Sistema de adquisición, gestión y supervisión de datos en tiempo real de un cuadro de baja tensión,” 2019. [Trabajo de Fin de Grado]
- [4] “Smart grids: electricity networks and the grid in evolution.” <https://www.i-scoop.eu/industry-4-0/smart-grids-electrical-grid/> (Última vez consultado en: Oct. 31, 2021).
- [5] Siemens, “SIMATIC Energy Manager Pro.” <https://new.siemens.com/mx/es/productos/automatizacion/industry-software/simatic-energy-management/simatic-energy-manager-pro.html> (Última vez consultado en: Nov. 02, 2021).
- [6] Siemens, “DESIGO.” <https://new.siemens.com/es/es/productos/building-technology/automatizacion/desigo.html> (Última vez consultado en: Nov. 02, 2021).
- [7] Schneider Electric, “Power SCADA Operator.” <https://www.se.com/mx/es/product-range/63067-ecostruxure-power-scada-operation/#documents> (Última vez consultado en: Nov. 03, 2021).
- [8] Schneider Electric, “Power Monitoring Expert.” <https://www.se.com/cr/es/product-range/62919-ecostruxure-power-monitoring-expert/#overview> (Última vez consultado en: Nov. 03, 2021).
- [9] Comité Técnico CTN 216 Eficiencia energética cambio climático y energías renovables UNE-EN, “UNE-EN ISO 50001:2018,” p. 47, 2018, [Online]. Disponible en: [https://www.en.aenor.com/\\_layouts/15/r.aspx?c=N0060594](https://www.en.aenor.com/_layouts/15/r.aspx?c=N0060594)
- [10] J.-M. Marcos-Fano, “Historia y panorama actual del sistema eléctrico español,” *Física y Sociedad*, vol. 13, pp. 10–17, 2010.
- [11] “Diferencias entre Smart Grids y Redes Eléctricas Convencionales.” <https://globalelectricity.wordpress.com/2013/12/19/diferencias-entre-smart-grids-y-redes-electricas-convencionales/> (Última vez consultado en: Oct. 29, 2021).
- [12] “Red Eléctrica de España.” <https://www.ree.es/es/red21/redes-inteligentes/que-son-las-smartgrid> (Última vez consultado en: Oct. 29, 2021).
- [13] “Definición de Smart Grids”, [Online]. Disponible en: <https://globalelectricity.wordpress.com/2013/12/19/definicion-de-smart-grids/>



- [14] “SUBMETERING monitorización energética avanzada.”  
<https://blog.linkener.com/blog/submetering#:~:text=Submetering se denomina a todas,o máquina donde estén instalados.> (Última vez consultado en: Nov. 01, 2021).
- [15] Jorge Marcos Rubio, “DESARROLLO E IMPLEMENTACIÓN DE ARQUITECTURA DE SUBMETERING EN MAQUETA INDUSTRIAL DE CUATRO VARIABLES,” 2020. [Trabajo de Fin de Grado]
- [16] Jorge Marcos Rubio, “COMPARATIVA ENTRE SCADAS PARA SUPERVISIÓN DE CONSUMOS ELÉCTRICOS EN MAQUETA INDUSTRIAL DE CUATRO VARIABLES,” 2020. [Trabajo de Fin de Grado]
- [17] Schneider Electric, “PowerLogic Electrical network management,” pp. 79–93, 2018.
- [18] M. de Sousa and P. Portugal, “MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3,” p. 36, 2016, doi: 10.1201/9781420041682.ch18.
- [19] Modbus-IDA, “Modbus Messaging on Tcp / Ip Implementation Guide,” pp. 1–46, 2006, [Online]. Disponible en:  
[https://www.modbus.org/docs/Modbus\\_Messaging\\_Implementation\\_Guide\\_V1\\_0b.pdf](https://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf)
- [20] “Información detallada sobre el protocolo MODBUS,” 2021. <https://www.ni.com/es-es/innovations/white-papers/14/the-modbus-protocol-in-depth.html> (Última vez consultado en: Oct. 18, 2021).
- [21] “multiprocessing — Paralelismo basado en procesos.”  
<https://docs.python.org/es/3/library/multiprocessing.html#multiprocessing.connection.Connection> (Última vez consultado en: Oct. 26, 2021).
- [22] “Diferencia entre procesos e hilos en Python.”  
<https://programmerclick.com/article/7486236779/> (Última vez consultado en: Nov. 02, 2021).
- [23] “Programación orientada a objetos”, Última vez consultado en: Oct. 27, 2021. [Online]. Disponible en: <https://desarrolloweb.com/articulos/499.php>
- [24] H. B. M. S. R. H. y S. J. Sigurdsson H., “Framework Design A Role Modeling Approach,” *Encyclopedia of volcanoes.*, no. 1995, p. 662, 2000.
- [25] “Laravel vs Lumen.” <https://belitsoft.com/laravel-development-services/microservices-architecture-development> (Última vez consultado en: Nov. 10, 2021).
- [26] “Comparativa entre Flask y Django.” <https://openwebinars.net/blog/django-vs-flask/> (Última vez consultado en: Nov. 10, 2021).
- [27] Deloitte, “Explicación ORM”, Última vez consultado en: Nov. 01, 2021. [Online]. Disponible en: <https://www2.deloitte.com/es/es/pages/technology/articles/que-es-orm.html>
- [28] “Backend y Frontend”, Última vez consultado en: Nov. 11, 2021. [Online]. Disponible en: <https://nestrategia.com/desarrollo-web-back-end-front-end/>
- [29] “El lenguaje de programación Python - Introducción”, Última vez consultado en: Oct. 16, 2021. [Online]. Disponible en:



<https://web.archive.org/web/20200224120525/https://luca-d3.com/es/data-speaks/diccionario-tecnologico/python-lenguaje>

- [30] “Historia de Python.” <https://docs.python.org/3/license.html> (Última vez consultado en: Oct. 16, 2021).
- [31] “Documentación oficial Python”, Última vez consultado en: Oct. 16, 2021. [Online]. Disponible en: <https://docs.python.org/release/3.10.2/tutorial/index.html>
- [32] “Principales implementaciones Python”, Última vez consultado en: Oct. 16, 2021. [Online]. Disponible en: <https://desarrollodesoftware.home.blog/2019/03/16/implementaciones-de-python/>
- [33] A. Sharma, “Python Global Interpreter Lock,” 2020. <https://www.datacamp.com/community/tutorials/python-global-interpreter-lock> (Última vez consultado en: Oct. 27, 2021).
- [34] Zina A. Aziz; Diler Naseradeen Abdulqader; Amira Bibo Sallow; Herman Khalid Omer, “Python Parallel Processing and Multiprocessing: A Rivew,” 2021, doi: <https://doi.org/10.25007/ajnu.v10n3a1145>.
- [35] “Pandas”, Última vez consultado en: Oct. 16, 2021. [Online]. Disponible en: [https://pandas.pydata.org/docs/user\\_guide/index.html](https://pandas.pydata.org/docs/user_guide/index.html)
- [36] “SQLAlchemy”, Última vez consultado en: Oct. 18, 2021. [Online]. Disponible en: <https://www.sqlalchemy.org/>
- [37] “PyModbus.” <https://pymodbus.readthedocs.io/en/latest/index.html> (Última vez consultado en: Oct. 18, 2021).
- [38] “APScheduler.” <https://apscheduler.readthedocs.io/en/3.x/index.html> (Última vez consultado en: Oct. 25, 2021).
- [39] “Que es el patrón MTV (Model Template View),” 2019. <https://espifreelancer.com/mtv-django.html> (Última vez consultado en: Nov. 11, 2021).
- [40] “Qué es Celery: Introducción y primeros pasos,” 2021. <https://openwebinars.net/blog/que-es-celery-introduccion-y-primeros-pasos/> (Última vez consultado en: Dec. 18, 2021).
- [41] “Celery - Distributed Task Queue.” <https://docs.celeryproject.org/en/stable/> (Última vez consultado en: Dec. 18, 2021).
- [42] J. M. R. M. Domínguez, J.J. Fuertes, P. Reguera, J. J. González, “MAQUETA INDUSTRIAL PARA DOCENCIA E INVESTIGACIÓN,” vol. 1, pp. 1–6, 2004.
- [43] W. Concentrator, M. Gateway, and E. Server, “EcoStruxure Panel Server Firmware Release Notes,” 2021.
- [44] “www.stackoverflow.com.” <https://stackoverflow.com/> (Última vez consultado en: Oct. 15, 2021).
- [45] “jQuery.” <https://jquery.com/> (Última vez consultado en: Nov. 28, 2021).





- [46] “www.w3schools.com.” <https://www.w3schools.com/python/default.asp> (Última vez consultado en: Oct. 16, 2021).
- [47] Schneider Electric, “PowerLogic PM5000 series,” 2021, [Online]. Disponible en: [https://download.schneider-electric.com/files?p\\_enDocType=User+guide&p\\_File\\_Name=EAV15105-ES09.pdf&p\\_Doc\\_Ref=EAV15105-ES](https://download.schneider-electric.com/files?p_enDocType=User+guide&p_File_Name=EAV15105-ES09.pdf&p_Doc_Ref=EAV15105-ES)
- [48] S. Electric, “A9MEM1540,” pp. 16–18, 2022.
- [49] Schneider Electric, “PowerLogic PM8000 series,” 2021, [Online]. Disponible en: <https://www.se.com/us/en/product/METSEPM8240/powerlogic-pm8000---pm8240-panel-mount-meter---intermediate-metering/?range=62252-powerlogic-pm8000-series>
- [50] “Django Project.” <https://www.djangoproject.com/> (Última vez consultado en: Nov. 11, 2021).