PROYECTO FIN DE GRADO

TÍTULO: DISEÑO DE SISTEMA DE MONITORIZACIÓN EN COMUNIDADES ENERGÉTICAS

AUTOR/A: PABLO GILSANZ HERRERO

TITULACIÓN: GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES

DIRECTOR/A: KIANE ALVES E SILVA

TUTOR/A: RITA HOGAN TEVES DE ALMEIDA

DEPARTAMENTO: DEPARTAMENTO DE INGENIERÍA TELEMÁTICA Y ELECTRÓNICA

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTE/A: MARÍA CRISTINA GUERRERO GARCÍA

TUTOR/A: RITA HOGAN TEVES DE ALMEIDA

SECRETARIO/A: FRANCISCO MARTÍNEZ MORENO

Fecha de lectura:

Calificación:

El Secretario/La Secretaria,

Resumen

El presente Proyecto de Fin de Grado tiene como objetivo el diseño e implementación de un sistema de monitorización adaptado a instalaciones fotovoltaicas de pequeña escala en comunidades energéticas rurales. Estas comunidades requieren herramientas específicas que permitan supervisar en tiempo real el estado de sus instalaciones, registrar los flujos de energía y facilitar la toma de decisiones (para maximizar el autoconsumo y la eficiencia energética).

Actualmente, la mayoría de las soluciones de monitorización están diseñadas para grandes plantas fotovoltaicas, lo que las hace costosas y poco adaptadas a las necesidades de pequeñas comunidades. Por otro lado, los sistemas de monitorización integrados en los inversores comerciales ofrecen funcionalidades limitadas y, en algunos casos, poco fiables. En este sentido, el presente proyecto busca cubrir esta brecha tecnológica mediante el diseño de una solución accesible y escalable que permita a las comunidades energéticas gestionar su producción de manera eficiente.

La solución propuesta emplea un Controlador Lógico Programable basado en ESP32, para la adquisición de datos de los distintos dispositivos del sistema. Los datos recopilados son enviados y almacenados en una base de datos de series temporales y visualizados en tiempo real mediante una herramienta de visualización. Además, se ha integrado una red privada virtual para permitir el acceso remoto seguro al sistema.

Los resultados obtenidos confirman la eficacia del sistema desarrollado, demostrando su capacidad para proporcionar datos precisos y en tiempo real sobre la generación de energía en instalaciones fotovoltaicas de pequeña y mediana escala. Se ha validado su compatibilidad con distintos modelos de inversores empleados, garantizando su funcionamiento con cada uno de ellos. Además, la flexibilidad del diseño y el uso de tecnologías de código abierto permiten su personalización según las necesidades específicas de cada usuario, así como su escalabilidad para futuras ampliaciones, facilitando la incorporación de nuevas funcionalidades y mejoras del sistema.

Abstract

The present Final Degree Project aims to design and implement a monitoring system tailored for small-scale photovoltaic installations in rural energy communities. These communities require specific tools that enable real-time supervision of their installations, recording of energy flows, and informed decision-making (to maximize self-consumption and energy efficiency).

Currently, most monitoring solutions are designed for large photovoltaic plants, making them expensive and poorly suited to the needs of small communities. Additionally, integrated monitoring systems in commercial inverters offer limited and sometimes unreliable functionalities. In this regard, the present project seeks to bridge this technological gap by designing an accessible and scalable solution that allows energy communities to efficiently manage their production.

The proposed solution employs a Programmable Logic Controller based on ESP32 for data acquisition from various system devices. The collected data is sent and stored in a time-series database and visualized in real-time through a visualization tool. Furthermore, a virtual private network has been integrated to enable secure remote access to the system.

The results obtained confirm the effectiveness of the developed system, demonstrating its ability to provide accurate and real-time data on energy generation in small- and medium-scale photovoltaic installations. Its compatibility with different inverter models has been validated, ensuring proper operation with each of them. Moreover, the flexibility of the design and the use of open-source technologies allow for customization according to each user's specific needs, as well as scalability for future expansions, facilitating the incorporation of new functionalities and system improvements.

Índice de figuras

Índice de tablas

Lista de acrónimos

Acrónimo	Descripción
ADC	Analog to Digital Converter (Convertidor analógico-digital)
ADU	Application Data Unit (Unidad de Datos de Aplicación)
CE	Comunidades Energéticas
CEC	Comunidades de Energía Ciudadana
CEM	Condiciones Estándar de Medida
CER	Comunidades de Energía Renovable
CRC	Comprobación de Redundancia Cíclica
FV	Fotovoltaicos
G	Irradiancia
GUI	Graphical User Interface (Interfaz Gráfica de Usuario)
H	Irradiación
Idc	Corriente de corriente continua
IDE	Integrated Development Environment (Entorno de desarrollo integrado)
IES	Instituto de Energía Solar
IoT	Internet Of Things (Internet de las Cosas)
Isc	Corriente de cortocircuito
JSON	JavaScript Object Notation (Notación de objeto de JavaScript)
LAN	Local Area Network (Red de área local)
M2M	Machine to Machine (máquina a máquina)
MPPT	Maximum Power Point Tracking (Seguimiento del Punto de Máxima Potencia)
MQTT	Message Queuing Telemetry Transport (Transporte de telemetría de cola de mensajes)
NAT	Network Address Translation (Traducción de Dirección de Red)
NTP	Network Time Protocol (Protocolo de Tiempo de Red)
ODS	Objetivos de Desarrollo Sostenible
OSI	Open Systems Interconnection (Interconexión de sistemas abiertos)
Pac	Potencia activa
Pdc	Potencia de corriente continua
PDU	Protocol Data Unit (Unidad de Datos de Protocolo)
PLC	Controlador Lógico Programable
QoS	Quality of Service (Calidad de Servicio)
RTC	Real Time Clock (Reloj en Tiempo Real)
Tc	Temperatura de célula
TSDB	Time Series Databases (Bases de datos de series temporales)
TSM	Time-Structured Mergetree (Árbol de Fusión Estructurado por Tiempo)
UE	Unión Europea
UTC	Universal Time Coordinated (Tiempo Universal Coordinado)
Vdc	Voltaje de corriente continua
Voc	Voltaje en circuito abierto
VPN	Virtual Private Network (Red Privada Virtual)

Índice de contenidos

1. Introducción

1.1 Marco y motivación del proyecto

El auge de las energías renovables ha impulsado la proliferación de instalaciones fotovoltaicas (FV) y la formación de comunidades energéticas (CE), especialmente en entornos rurales [1]. Un ejemplo de estas es el proyecto JALON [2], una iniciativa europea coordinada por el Grupo de Sistemas Fotovoltaicos del Instituto de Energía Solar (IES) de la Universidad Politécnica de Madrid y financiada por el Programa LIFE de la Comisión Europea. JALON tiene como objetivo gestionar una CE regional a gran escala en la región rural de Calatayud. El proyecto busca involucrar a 5,000 ciudadanos en 87 pueblos, además de organizaciones y empresas locales para generar, gestionar y consumir conjuntamente energía renovable a través de sistemas FV distribuidos especialmente en sistemas de autoconsumo individual y colectivo. Con esta iniciativa, se promueve la sostenibilidad, el ahorro energético y la autonomía local, sirviendo además como ejemplo para el surgimiento y crecimiento de proyectos similares en toda Europa.

Estas comunidades, caracterizadas por instalaciones de menor escala, requieren sistemas de monitorización específicos que permitan:

- Supervisar el estado de las instalaciones para facilitar tareas de operación y mantenimiento eficientes.
- Registrar los flujos de energía, incluyendo la producción, el autoconsumo y la energía exportada a la red.

Los sistemas de monitorización diseñados para grandes plantas FV no son adecuados para estas aplicaciones debido a diferencias técnicas y económicas. Por ello, es necesario desarrollar soluciones adaptadas a las particularidades de las comunidades energéticas rurales.

En este contexto, el presente trabajo se centrará en el desarrollo de un sistema de monitorización adaptado a las necesidades de las CE rurales, con enfoque en sistemas de autoconsumo FV, contribuyendo así a la eficiencia y sostenibilidad de estas iniciativas.

1.2 Objetivos técnicos y académicos

Este proyecto tiene como objetivo principal desarrollar un sistema de monitorización adaptado a las necesidades específicas de las CE rurales con instalaciones FV de autoconsumo, asegurando su viabilidad tanto técnica como económica.

El sistema, permitirá la supervisión en tiempo real del estado operativo de las instalaciones, permitiendo la detección de cualquier anomalía o fallo en el sistema facilitando tareas de operación y mantenimiento eficientes.

Además, el sistema se centrará en registrar los flujos de energía, incluyendo la producción y la energía exportada a la red, con el propósito de optimizar el rendimiento y la gestión

energética de la comunidad. Esta capacidad de análisis permite a los gestores de la comunidad energética comprender mejor los patrones de generación y consumo, facilitando la toma de decisiones para mejorar la eficiencia energética y maximizar el autoconsumo.

Otro objetivo es garantizar la compatibilidad y escalabilidad del sistema de monitoreo con diversas configuraciones de instalaciones fotovoltaicas de autoconsumo presentes en entornos rurales. Para ello, se trabajará con tres tipos distintos de inversores de marcas utilizadas por las empresas locales, de manera que, realizando pequeños ajustes en el código, el sistema sea plenamente funcional.

1.3 Estructura del resto de la memoria

El contenido del resto de la memoria está distribuido en los siguientes capítulos:

- Capítulo 1: Introducción

Este capítulo establece el contexto general del proyecto, presentando los objetivos técnicos y académicos, así como la motivación para su desarrollo. Además, se ofrece una visión general del resto de la memoria.

- Capítulo 2: Marco Tecnológico

En este apartado se analizan los fundamentos teóricos y el estado del arte de las comunidades energéticas y las plataformas de monitorización existentes, contextualizando la problemática y justificando la necesidad del proyecto.

- Capítulo 3: Especificaciones y Restricciones de Diseño

En este apartado se definen las especificaciones técnicas y las restricciones bajo las cuales se ha desarrollado el proyecto. Se detallan los requisitos funcionales y no funcionales, los criterios de rendimiento y las limitaciones técnicas.

- Capítulo 4: Descripción de la Solución Propuesta

Este capítulo describe en profundidad la solución desarrollada, incluyendo el diseño del sistema de monitorización, las tecnologías empleadas y la arquitectura implementada. También se detallan las herramientas utilizadas y los procesos llevados a cabo para integrar los diferentes elementos del sistema.

- Capítulo 5: Resultados

En este capítulo se describen todas las pruebas realizadas para comprobar el correcto funcionamiento del sistema. Además, se presentan los resultados obtenidos durante las pruebas realizadas, respaldados con datos y gráficos ilustrativos.

- Capítulo 6: Presupuesto

En este apartado se realiza un análisis de los costos asociados al proyecto, tanto en términos de costes materiales como los costes por mano de obra.

- Capítulo 7: Impacto del Proyecto

Se examinan las implicaciones del proyecto en ámbitos sociales, económicos, ambientales y tecnológicos. Además, se analiza la contribución del proyecto a los objetivos del desarrollo sostenible (ODS) y su papel en la transición hacia un modelo sostenible.

- Capítulo 8: Conclusiones y Trabajos Futuros

Este capítulo recopila las principales conclusiones del proyecto, evaluando el cumplimiento de los objetivos planteados. También se proponen posibles líneas de investigación y desarrollo futuro para ampliar y mejorar el sistema desarrollado.

- Capítulo 9: Referencias

Contiene la lista de todas las fuentes bibliográficas utilizadas a lo largo del proyecto, organizadas de forma que permitan identificar fácilmente el origen de la información.

- Anexo: Manual de usuario

En esta parte de la memoria se detalla información adicional relevante para la replicación futura del proyecto.

2. Marco tecnológico

En este apartado se analizarán los conceptos clave relacionados con las CE y la monitorización de FV, fundamentales para el desarrollo del proyecto.

2.1 Comunidades energéticas

2.1.1 Definición de Comunidades Energéticas

Las CE han emergido como pilares fundamentales en la transición energética europea, impulsadas por la necesidad de integrar a los ciudadanos en el proceso de generación, distribución y consumo energético [3]. Estas comunidades son organizaciones colectivas, formadas por individuos, empresas locales y autoridades, con el objetivo principal de fomentar la sostenibilidad ambiental, mejorar la eficiencia energética y generar beneficios sociales y económicos para sus miembros y comunidades circundantes [4].

Actualmente no existe una definición estándar para las CE. Los términos utilizados tienden a superponerse o variar según el contexto. Sin embargo, el Paquete de Energía Limpia de la Unión Europea (UE) define dos tipos de comunidades energéticas [5]:

- Comunidades de Energía Ciudadana (CEC): Entidad legal basada en participación abierta y voluntaria, controlada por personas físicas, autoridades locales o pequeñas empresas. Su propósito principal es proporcionar beneficios sociales, ambientales o económicos a la comunidad en lugar de obtener ganancias financieras. Puede participar en actividades como generación, suministro, almacenamiento de energía, servicios de eficiencia energética y carga de vehículos eléctricos.

- Comunidades de Energía Renovable (CER): Entidad legal similar a las CEC, pero centradas en proyectos de energía renovable y con un enfoque geográfico limitado. Sus miembros deben encontrarse en proximidad a las instalaciones energéticas, lo que refuerza la conexión local y el impacto directo en el entorno.

Ambas definiciones comparten principios fundamentales como la participación abierta y voluntaria, el control efectivo por parte de sus miembros, y un enfoque en beneficios sociales y ambientales más que en ganancias financieras. Sin embargo, las CER se diferencian por su compromiso con proyectos de energía renovables y su vinculación con el [6].

Este trabajo se centra en las CER, considerando su enfoque en un modelo energético descentralizado basado en fuentes renovables y su papel en la producción y gestión de energía dentro de comunidades locales.

2.1.2 Motivación

La formación de CE responde a la necesidad de agrupar a pequeños productores para facilitar su integración en los mercados eléctricos. Este modelo mejora su capacidad de gestión colectiva y reduce las pérdidas energéticas asociadas a la transmisión a larga distancia. En este

contexto, emerge la figura de los prosumidores, ciudadanos que, anteriormente, eran consumidores pasivos y que ahora actúan como productores y consumidores activos de la energía que generan [7].

Según un estudio [8] realizado a miembros de diversas CE en Europa, las principales motivaciones para participar o fundar una CE son de naturaleza ambiental, social y comunitaria, superando a factores como la reducción de costos de energía o la generación de ingresos locales. Los ciudadanos reconocen que estas comunidades aportan valores no monetarios a su entorno, como el impulso a la transición energética, la orientación en el uso de la energía y la promoción de iniciativas sociales y ambientales. En este sentido, su disposición a participar en una CE está vinculada a la percepción de beneficios para la comunidad y el medio ambiente.

Además, la confianza entre los miembros y el sentido de pertenencia son aspectos clave para el éxito de estas iniciativas. Las CE no solo se perciben como una herramienta para promover la sostenibilidad local, sino también como un mecanismo para fortalecer la autonomía energética. Este enfoque colaborativo impulsa un modelo más participativo, democrático y sostenible, adaptado a las necesidades locales.

2.1.3 Comunidades energéticas en Europa

En el contexto europeo, las CE han experimentado un notable desarrollo gracias al marco normativo impulsado por el Paquete de Energía Limpia de la Unión Europea y programas de investigación como Horizon 2020. Estas iniciativas no solo han fomentado la transición hacia modelos energéticos más sostenibles, sino que también han promovido la participación ciudadana en la gestión energética. A continuación, se presentan algunos ejemplos representativos de CE en Europa [9]:

- Enercoop (Francia).

Fundada en 2005, es una de las mayores cooperativas energéticas de Europa, ofreciendo energía 100% renovable a un precio justo. Es gestionada por 11 cooperativas regionales y compra electricidad directamente a productores de fuentes como hidroeléctrica, solar, eólica y biomasa. Su objetivo es promover un modelo energético descentralizado y democrático liderado por ciudadanos [10].

- EWS Schönau (Alemania).

EWS Schönau nació como respuesta al desastre de Chernóbil y es un referente en la generación y distribución de energía renovable. Su enfoque multi-energético incluye eólica, solar, biomasa y biogás, además de innovadores servicios de gestión energética. Fue pionera en tomar el control de la red local en los años 90, consolidándose como un modelo de autonomía energética y sostenibilidad [11].

- Isla de Eigg (Escocia).

La Isla de Eigg es la primera comunidad en implementar un sistema eléctrico completamente aislado de la red de Reino Unido, basado en energía renovable. Combina hidroeléctrica, eólica y solar para ofrecer un suministro constante y reducir el uso de generadores fósiles. Este modelo demuestra cómo incluso comunidades remotas pueden ser sostenibles y resilientes energéticamente [12].

- Comunidad Energética Hortícola (España)

La Comunidad Energética ECOHORTÍCOLA, situada en Andalucía (España), apuesta por un modelo agroindustrial más sostenible mediante la generación fotovoltaica y el uso compartido de vehículos eléctricos. El proyecto combina autoconsumo renovable y movilidad eléctrica para reducir la dependencia de la red convencional. Esta iniciativa muestra cómo pequeñas sociedades pueden liderar la transición energética desde el sector agrícola[13].

2.1.4 Comunidades energéticas en el contexto español

En España, las CE están surgiendo como actores clave en la transición hacia un modelo energético descentralizado y sostenible, aunque su desarrollo sigue siendo menor en comparación con otros países europeos [14]. El último informe del Observatorio Nacional de Comunidades Energéticas [15] contabilizó 353 CE en España, considerando datos hasta junio de 2023. Según el estudio, estas comunidades se caracterizan por la participación ciudadana y han adoptado el autoconsumo fotovoltaico como su eje principal, promoviendo la generación y el consumo de energía renovable a nivel local.

El marco regulatorio en España ha experimentado una evolución importante en los últimos años para fomentar el desarrollo de las CE. En 2010, la legislación permitió por primera vez que las cooperativas de energía comercializaran electricidad. Paralelamente, el Real Decreto 1699/2011 [16], estableció el marco inicial para la generación distribuida, regulando el autoconsumo, aunque sin mecanismos de compensación por la energía excedente inyectada en la red. A partir de 2012 se produjo un cambio significativo en las políticas gubernamentales, suprimiendo las primas a las nuevas instalaciones de energía renovable lo que marcó una etapa de mayor alineación con los intereses de las comunidades energéticas. A pesar de estos avances, en 2015, el Real Decreto 900/2015 [17] introdujo restricciones al autoconsumo, incluyendo el controvertido "impuesto al sol", que limitó el autoconsumo y retrasó el desarrollo de proyectos de energía solar comunitarios.

El panorama mejoró sustancialmente en 2018 con la aprobación del Decreto-Ley 15/2018 [18], que eliminó estas barreras y permitió el autoconsumo compartido. En el año siguiente, el Real Decreto 244/2019 [19] reguló el autoconsumo y reconoció legalmente a las CE. Con eso, muchas CE han podido desarrollar modelos descentralizados basados en el autoconsumo FV, fomentando la participación ciudadana y la generación local de energía renovable. No obstante, persisten obstáculos como la complejidad burocrática y la limitada financiación pública, que ralentizan su expansión. El desarrollo y análisis de las CE en España se enfrentan,

además, a la falta de una base de datos centralizada y homogénea, lo que dificulta su seguimiento y estudio sistemático. Algunos ejemplos destacados de comunidades energéticas solares son los siguientes:

- Som Energía.

Es la cooperativa de energía renovable más grande de España. Fundada en 2010 se centra en la generación y comercialización de energía renovable, incluyendo proyectos solares. Som Energia ha desarrollado plantas fotovoltaicas que producen más de 5 GWh al año y ha sido pionera en modelos de financiación participativa en la que cualquier particular, empresa o administración puede unirse a la cooperativa [20].

- EnergÉtica.

Operando en Castilla y León desde 2014, es una cooperativa de consumidores sin ánimo de lucro que quiere definir su propio modelo energético. EnergÉtica ha promovido varios proyectos de energía solar, incluyendo instalaciones fotovoltaicas en cooperativas agrícolas y asociaciones vecinales [21]

- Comunidad energética del Campus Sur.

Es un proyecto cooperativo y colaborativo de la Universidad Politécnica de Madrid en el que estudiantes, personal de técnico, de gestión y de administración y servicios, profesores e investigadores participan de forma activa en una comunidad energéticamente sostenible y autosuficiente. Surge dentro del Proyecto AURORA y pretendía ser un referente en toda Europa instalando tejados fotovoltaicos en todas las azoteas del campus mediante micro inversiones de sus miembros. Como esta solución no fue posible por cuestiones burocráticas, finalmente se va a hacer la instalación FV, así como dos sistemas de aerotermia, en el colegio Centro Cultural Palomeras [22]

- CERCA

CERCA es la comunidad energética en desarrollo impulsada por el Proyecto JALÓN en la Comarca de Calatayud (Zaragoza). Bajo un modelo cooperativo sin ánimo de lucro, promueve la producción y el consumo compartido de energía solar para ofrecer electricidad limpia y asequible a hogares, pymes y ayuntamientos, incluso sin tejado propio. Su enfoque comarcal refuerza la gestión colectiva, reduce costes y activa iniciativas sociales y ambientales que revitalizan el territorio. CERCA ejemplifica cómo la cooperación local puede transformar el modelo energético rural hacia uno más justo y sostenible [23].

2.2 Sistemas Fotovoltaicos de Autoconsumo

Los sistemas FV de autoconsumo son instalaciones de generación de energía solar diseñadas para cubrir, total o parcialmente, el consumo eléctrico de uno o varios usuarios. Su objetivo principal es reducir la dependencia de la red eléctrica convencional, optimizando el uso de

fuentes renovables y promoviendo un modelo energético más sostenible y descentralizado [4].

El autoconsumo puede ser individual, cuando un único consumidor genera y consume la energía producida por su propia instalación fotovoltaica, o colectivo, cuando varios consumidores comparten la electricidad generada por una misma instalación fotovoltaica, siempre que estén dentro de un área determinada y conectados a la misma red de distribución [24]. El modelo de autoconsumo colectivo ha facilitado el desarrollo de las CE al permitir una gestión compartida de la energía.

La Figura 1 presenta un esquemático base de un sistema de autoconsumo. Entre sus principales elementos están:

- Módulos fotovoltaicos para generación de energía eléctrica.
- Inversor para transformar la corriente continua (CC) producida por los módulos fotovoltaicos en corriente alterna (CA), para su uso en las viviendas conectados al sistema.
- Contador inteligente para medir la cantidad de energía generada, consumida y vertida a la red.
- Red eléctrica para compra de energía cuando la producción FV es insuficiente, así como para venda de excedentes cuando hay producción FV sobrante.
- Consumidor(es), que utilizan la energía generada por el sistema FV.

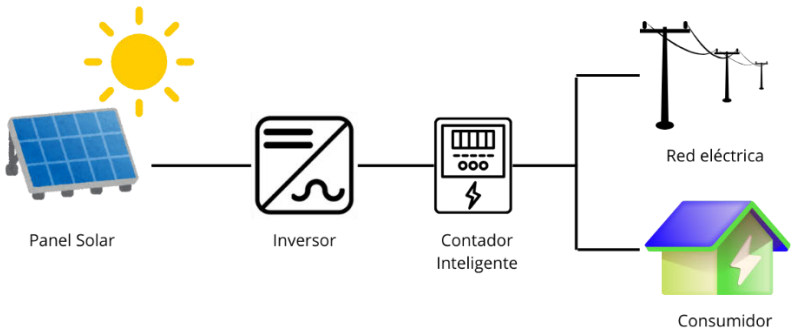

Figura 1: Esquemático base de un sistema fotovoltaico de autoconsumo

En los sistemas de autoconsumo colectivo, se emplea también un sistema de gestión y monitorización para control en tiempo real de la producción FV, el consumo individual de cada miembro, la compra de energía de la red y los excedentes generados para posterior compensación [25]. El seguimiento de esos parámetros permite mejorar el uso del sistema, reducir la dependencia de la red eléctrica (y con ello la factura energética) y favorecer la sostenibilidad mediante la reducción de emisiones.

2.3 Monitorización de sistemas fotovoltaicos

La monitorización de los sistemas FV en una CE se presenta como un elemento clave para garantizar el uso eficiente y sostenible de la energía solar [26] [27]. Estos sistemas permiten

supervisar y analizar en tiempo real el comportamiento de las instalaciones, optimizando su rendimiento, reduciendo pérdidas y detectando posibles fallos de manera temprana. La capacidad de recopilar datos precisos y utilizarlos para mejorar la operación de los sistemas es fundamental para maximizar la rentabilidad y asegurar la estabilidad del suministro energético en estas comunidades.

Un sistema de monitorización FV consta de varios módulos clave, cada uno con funciones específicas que contribuyen al control y análisis integral del sistema [28]. El diagrama de bloques general de un sistema de monitorización FV típico se muestra en la Figura 2.

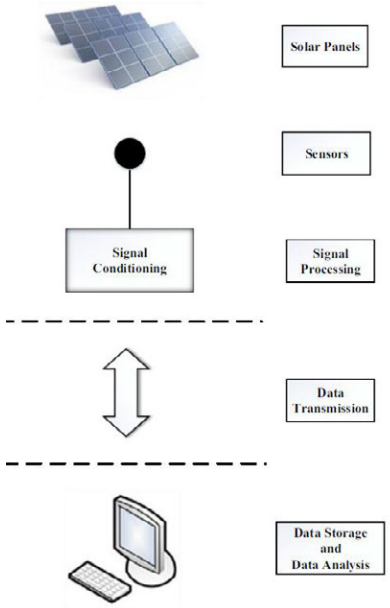

Figura 2: Diagrama de bloques general de un sistema FV [28]

A continuación, se describen las principales funciones y componentes de estos módulos:

- Módulos FV: Capturan la energía solar y la convierten en electricidad empleando el efecto fotovoltaico.
- Sensores: Instrumentos que miden parámetros clave como corriente, voltaje, temperatura y radiación solar. Entre los sensores comunes se encuentran el piranómetro para medir la irradiancia y los termopares o RTD para la temperatura.
- Acondicionador de señales: Realiza la amplificación y el filtrado de las señales analógicas obtenidas por los sensores, mejorando la calidad y precisión de los datos.
- Controlador: Gestiona las señales obtenidas, digitaliza los datos mediante convertidores analógico-digitales (ADC) y los transmite a una base de datos situada en un ordenador o en la nube para su almacenamiento.
- Transmisión de datos: Los sistemas modernos emplean métodos avanzados como protocolos de comunicación como Modbus o comunicación inalámbrica para garantizar la integridad y velocidad del flujo de información.
- Almacenamiento y análisis de datos: Un ordenador o sistema central analiza los datos empleando herramientas de software permitiendo representar estos en gráficos para una mejor visualización.

La integración de los módulos de monitorización en sistemas fotovoltaicos permite supervisar el funcionamiento de las instalaciones de manera detallada, lo que resulta fundamental para identificar problemas como la degradación de los módulos solares, fallos en los inversores o el impacto de condiciones meteorológicas adversas. Estos sistemas contribuyen significativamente a optimizar la eficiencia operativa, ya que permiten realizar mantenimientos preventivos y correctivos de manera oportuna, minimizando las pérdidas energéticas y reduciendo costes a largo plazo.

Sin embargo, el alto coste y la complejidad de estos sistemas de monitorización restringen su implementación principalmente a plantas FV de gran escala. Este desafío ha llevado a una creciente investigación en torno a tecnologías de monitorización que sean más accesibles y adaptadas a plantas fotovoltaicas de pequeña y mediana escala. Para estas instalaciones, como las asociadas a las CE, resulta fundamental disponer de sistemas más simples, que puedan ofrecer un monitoreo continuo de la producción energética, garantizar una operación estable y predecir o detectar posibles condiciones adversas que comprometan el rendimiento.

Como alternativa a los sistemas de monitorización complejos mencionados anteriormente, una de las soluciones más simples y accesibles para monitorizar la producción de energía fotovoltaica son los sistemas integrados en los inversores. Desarrollados por los propios fabricantes, estos sistemas están diseñados para facilitar el control y el seguimiento de la generación energética, siendo ampliamente utilizados en viviendas particulares e instalaciones de pequeña escala. Principalmente, permiten registrar y analizar la energía producida por los paneles solares, así como la energía inyectada a la red eléctrica.

Un ejemplo destacado de esta tecnología es la solución implementada por fabricantes como Huawei con su Solar App, una plataforma digital disponible tanto aplicación móvil como en la web que integra herramientas de monitorización en tiempo real. Estas aplicaciones permiten a los usuarios visualizar gráficas detalladas de generación de energía, evaluar el rendimiento del sistema, supervisar el estado de la batería, monitorizar el flujo de energía entre los distintos componentes del sistema y la posibilidad de recibir alertas en caso de fallos o anomalías. Puede verse un ejemplo de estas funcionalidades en la figura 3.

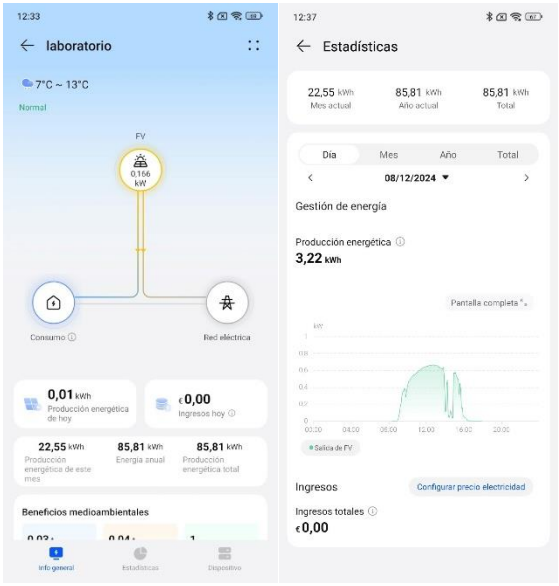

Figura 3: Aplicación Solar App de Huawei

Si bien estos sistemas son ideales para instalaciones pequeñas o usuarios particulares, muestran importantes limitaciones cuando se desean implementar estos en situaciones más complejas como las de una CE. Una de las principales restricciones radica en que los datos recopilados se limitan únicamente a aquellos que son gestionados por el inversor, lo que puede resultar en una perspectiva incompleta. Por ejemplo, en un sistema de autoconsumo colectivo es muy importante conocer el consumo de cada consumidor y de cada prosumidor. Asimismo, otros parámetros como la temperatura son obtenidos a través de fuentes externas, como internet, lo que deriva en información imprecisa y poco representativa de las condiciones reales.

Además, la necesidad de una conexión constante a internet para almacenar y procesar los datos representa un desafío adicional en proyectos localizados en zonas con conectividad limitada. Por ello, aunque estos sistemas resultan útiles en aplicaciones específicas, no son la solución más adecuada para las exigencias técnicas y operativas de las CE.

3. Especificaciones y restricciones de diseño

El diseño del sistema de monitorización debe cumplir con una serie de especificaciones y restricciones que aseguren su funcionalidad y eficiencia a largo plazo. Este apartado describe dichas especificaciones y restricciones, organizadas según los requisitos clave del sistema

3.1 Idea inicial del proyecto

Teniendo en cuenta el esquema que presentan los sistemas de monitorización complejos y los integrados en los inversores vistos en el apartado anterior, se propone una combinación de ambos para lograr satisfacer las diferentes necesidades de los sistemas FV que podremos encontrarnos en las CE rurales. Por ello, dividiremos el sistema a desarrollar en 3 etapas claramente diferenciadas como se observa en la Figura 4:

- Adquisición y envío de datos: En esta etapa, se llevará a cabo la recopilación de los datos que se consideren relevantes provenientes del inversor fotovoltaico, el contador y el módulo de referencia. Además, deberá incluir mecanismos para garantizar la integridad de los datos recopilados y prevenir la pérdida de información al realizar el envío de estos.
- Almacenamiento: La información recolectada por los equipos se transfiere a una base de datos para su almacenamiento indefinido, garantizando su disponibilidad en caso de que se deseen realizar consultas.
- Visualización: Los datos almacenados se presentan mediante herramientas de visualización intuitivas y claras, que permitan a los usuarios finales y gestores de la CE interpretar la información de manera efectiva y rápida.

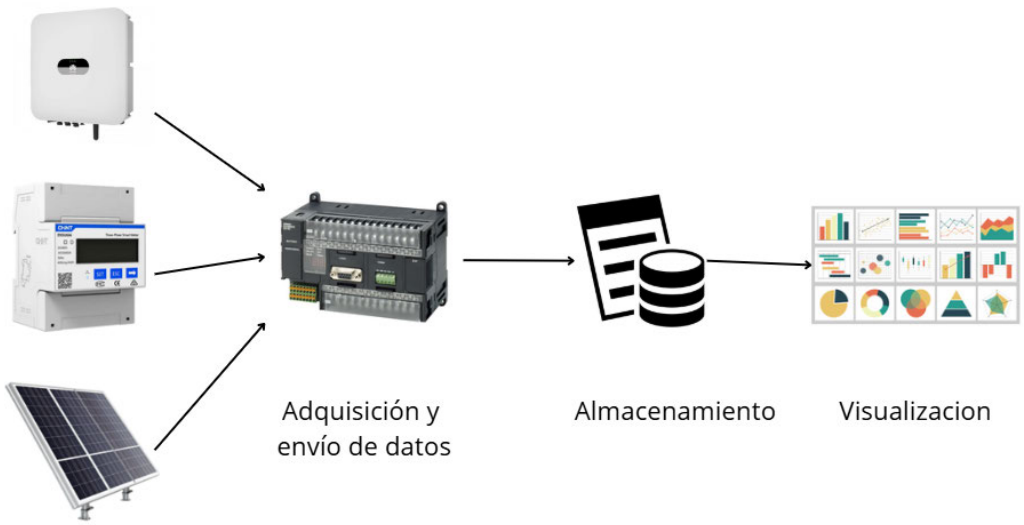

Figura 4: Concepto inicial del proyecto

3.2 Especificaciones técnicas

Se consideran las siguientes especificaciones técnicas:

- Los datos que el sistema debe ser capaz de monitorizar son los siguientes:
    - Vdc (tensión de corriente continua) e Idc (corriente de corriente continua) provenientes de cada uno de los *strings* conectados al inversor.
    - Pdc (potencia de corriente continua) total que recibe el inversor (suma de las potencias de los *strings*).
    - Pac (potencia activa) que inyecta el inversor a la red eléctrica. Este parámetro será extraído desde un contador externo.
    - Voc (tensión en circuito abierto) e Isc (corriente de cortocircuito) para calcular temperatura de célula (Tc) e irradiancia (G). Estos datos se obtienen desde un módulo de referencia previamente calibrado para ello.
- La monitorización de todos los parámetros debe realizarse con un intervalo de 5 minutos para permitir un seguimiento preciso de la instalación.
- Los datos recolectados deben almacenarse indefinidamente en una base de datos o similar, asegurando su disponibilidad para visualización y análisis en cualquier momento.
- El sistema debe ser capaz de operar tanto con conexión a Internet como sin ella, garantizando la recopilación y almacenamiento local de datos en ausencia de acceso a la red.
- Los dispositivos empleados para la realización del sistema deben ser del mismo tipo de inversores y contadores utilizados por las empresas locales de Calatayud.

3.3 Restricciones de diseño

A la hora de diseñar el sistema de monitorización se tiene que garantizar que:

- El sistema sea económicamente viable para pequeñas CE, con un costo inicial asequible y bajos costos de mantenimiento.
- Todo el software utilizado sea de código abierto para reducir costos y facilitar la personalización según las necesidades específicas de cada instalación.
- El sistema sea fácil de instalar por personal técnico no especializado, con un diseño que facilite su mantenimiento y minimice los costos operativos.
- La plataforma de monitorización y el interfaz de usuario sean intuitivos y fáciles de entender, permitiendo que los usuarios puedan interpretar los datos de forma clara con un simple vistazo.

4. Descripción de la solución propuesta

Para el desarrollo del sistema de monitorización propuesto, se ha diseñado una solución integral que combina elementos hardware y software.

Como se puede observar en la Figura 5, el sistema final interconecta todos los equipos entre si empleando diversas técnicas y protocolos de comunicación. Podemos distinguir los siguientes elementos:

- El generador FV que incluye los Módulos de referencia, el inversor solar y el contador como elementos de monitorización encargados de realizar la captura de los datos a monitorizar.
- Un Controlador Lógico Programable (PLC) donde se implementa el código para la recopilación, procesado y envío de los datos (mediante el protocolo MQTT a un ordenador para su almacenamiento y visualización).
- Un ordenador servidor, que recibe los datos del PLC y aloja el bróker MQTT, la base de datos de influxDB y la herramienta de visualización Grafana. Este ordenador estará conectado a una red privada virtual (VPN) mediante el programa Tailscale, lo que permite acceder a todos los programas mencionados anteriormente de forma remota desde cualquier otro dispositivo u ordenador remoto.

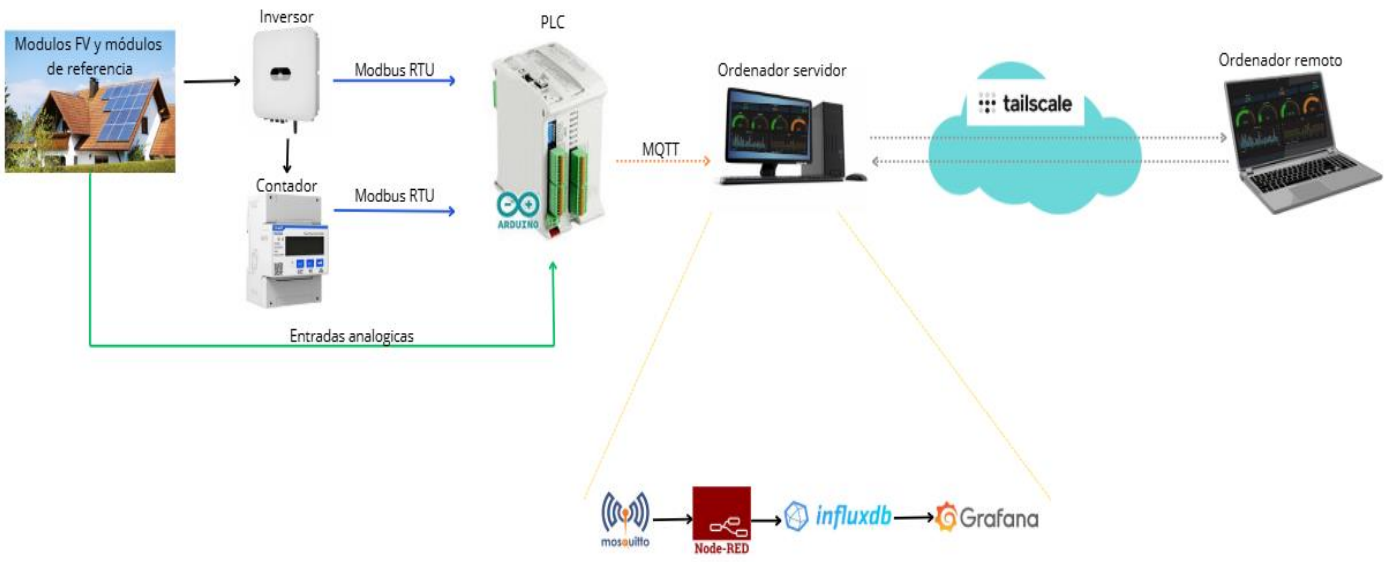

Figura 5: Esquema general de la solución propuesta

El sistema de monitorización operará de la siguiente manera: cada cinco minutos, el PLC realiza las lecturas de los registros correspondientes a los parámetros a monitorizar mediante el protocolo Modbus RTU, obteniendo las medidas de los dispositivos de monitorización. Simultáneamente, el PLC lee los valores de las entradas analógicas de los módulos de referencia para calcular parámetros como la temperatura de célula e irradiancia. Tras la adquisición de todos estos datos, el PLC los almacena en una tarjeta SD integrada como respaldo, asegurando su disponibilidad en caso de fallos en la comunicación. Posteriormente,

los datos se convierten al formato JSON (*JavaScript Object Notation*) para facilitar su envío. Utilizando el protocolo MQTT, el PLC transmite los datos al *bróker* Mosquitto alojado en el ordenador que actúa como servidor local. Una vez recibidos, Node-RED procesa y adapta los datos para su almacenamiento en la base de datos InfluxDB. Finalmente, Grafana, ejecutándose en el servidor, visualiza los datos en gráficos y paneles. El acceso remoto a Grafana, se facilita mediante una VPN establecida con Tailscale, permitiendo a los usuarios autorizados supervisar el sistema desde cualquier ubicación.

Este flujo de operaciones asegura una monitorización continua y fiable, proporcionando a el usuario final una herramienta efectiva para supervisar y analizar el rendimiento del sistema FV en tiempo real.

4.1 Hardware

A continuación, se describen todos los sistemas hardware como el PLC, los inversores, el contador y los módulos de referencia con los que se ha trabajado a lo largo del proyecto.

4.1.1 ESP32 PLC 21 +

El dispositivo seleccionado para el desarrollo del código es el ESP32 PLC 21+, un autómata industrial de la marca Industrial Shields, reconocido por su fiabilidad y calidad en aplicaciones de monitorización en entornos industriales [29]. Inicialmente, se consideró utilizar un microcontrolador común basado en ESP32 que es más barato que la opción escogida, sin embargo, se optó por el PLC debido a sus características avanzadas y robustez. Asimismo, se evaluó el modelo ESP32 PLC 14, pero se descartó debido a la falta de módulos esenciales, como el del RTC (Real Time Clock) y el de la SD, necesarios para cumplir con las especificaciones del sistema del sistema.

El ESP32 PLC 21+ cuenta con 21 I/O (entradas/salidas), proporcionando una amplia capacidad para gestionar múltiples señales. Ofrece diversos sistemas de comunicación, incluyendo I2C, SPI y RS485, lo que le confiere gran flexibilidad y control en la integración con otros dispositivos y sistemas. Este autómata puede programarse a través del puerto USB utilizando lenguajes C/C++ mediante la plataforma de código abierto Arduino IDE. Además, Industrial Shields proporciona un paquete de código con las librerías y herramientas necesarias para facilitar el desarrollo del código en sus dispositivos. En la figura 6 se puede observar el aspecto de este PLC.

Figura 6: ESP32 PLC 21 + [29]

Entre sus características, se destacan las siguientes (que serán de vital importancia para el desarrollo del proyecto):

- Tensión de alimentación entre 12 y 24 Vdc y una corriente mínima de 2 A
- Entradas analógicas con un rango de voltaje de 0 a 10 Vdc y una resolución de 11 bits, permitiendo lecturas precisas con valores que oscilan entre 0 y 2047.
- Comunicación RS485 mediante transceptor MAX485 en configuración half-duplex, facilitando la comunicación robusta y eficiente a través del protocolo Modbus RTU.
- Conexiones Wi-Fi en la banda de 5 GHz proporcionando una comunicación inalámbrica rápida y fiable.
- Módulo RTC, que mantiene una temporización precisa, asegurando la sincronización de eventos y tareas programadas, incluso en ausencia de conexión a internet.
- Ranura para tarjetas MicroSD de hasta 32 GB, ofreciendo una solución fiable y compacta para el almacenamiento de datos, esencial para guardar información como medio de respaldo.

4.1.2 Inversores

Los inversores seleccionados han sido elegidos por ser los más comunes con los que trabajan empresas locales de la zona de Calatayud. A continuación, se detallan las características de cada uno:

- **Inversor Huawei SUN2000-2KTL-L1**

El Huawei SUN2000-2KTL-L1 es un inversor monofásico híbrido con una potencia de salida de 3kW. Cuenta con una eficiencia máxima del 96,7% y dispone de 2 seguidores MPPT (Maximum Power Point Tracking), lo que permite optimizar la producción en instalaciones con diferentes orientaciones. Cuenta con su propio sistema de monitorización mediante la aplicación FusionSolar y puede establecer comunicación a través de WLAN, Ethernet y RS485. Se puede visualizar el aspecto de este inversor en la Figura 7.

Figura 7: Inversor Huawei SUN2000-2KTL-L1 [30]

- **Inversor Fronius PRIMO 3.0-1**

El Fronius PRIMO 3.0-1 es un inversor monofásico sin transformador con una potencia de salida de 3.000 W. Incorpora 2 MPPT, permitiendo la conexión de hasta dos *strings* de paneles FV por cada uno. Dispone de una interfaz de usuario que permite consultas en tiempo real y la modificación de parámetros sin necesidad de aplicaciones externas. Además, posee su propio sistema de monitorización mediante Fronius Solar Web y cuenta con interfaces WLAN y RS485 para la comunicación. El aspecto de este inversor se ilustra en la Figura 8.

Figura 8: Inversor Fronius PRIMO 3.0-1[31]

- **Inversor SAJ R5-1.5K-S1**

El SAJ R5-1.5K-S1 es un inversor monofásico diseñado para pequeñas instalaciones residenciales, con una potencia de salida de 1.500 W. Está equipado con un único MPPT y cuenta con su propio sistema de monitorización mediante una aplicación llamada eSolar. Incluye un *datalogger* esencial para su funcionamiento, que permite la visualización de datos. Dispone de puertos de comunicación RS232 (USB), RS485 (RJ45), Wi-Fi y Bluetooth. En la Figura 9 se representa la apariencia de este inversor.

Figura 9: Inversor SAJ R5-1.5K-S1[32]

4.1.3 Contador

El contador escogido ha sido el CONTAX D-6041-BUS, un contador digital monofásico de energía activa y reactiva diseñado para sistemas de corriente alterna monofásicos con una tensión de 230 V y una corriente máxima de 60 A. este dispositivo se conecta directamente al circuito eléctrico y está diseñado para operar de manera continua. Además, está equipado con una pantalla LCD de cinco dígitos que proporciona una visualización clara y precisa de los datos de consumo y otros parámetros eléctricos relevantes. También cuenta con funciones de analizador de redes y comunicación mediante protocolo Modbus a través de una salida RS-485. La Figura 10 muestra el aspecto de este contador.

Figura 10: Contador CONTAX D-6041-BUS [33]

4.1.4 Módulos de referencia

Los módulos de referencia son paneles solares del mismo tipo que los utilizados en el generador fotovoltaico, pero con una conexión independiente. A diferencia de los módulos conectados a la instalación, estos no forman parte del sistema de generación de energía, sino que sirven como punto de referencia para evaluar el rendimiento del resto de los paneles. Su función principal es medir parámetros de funcionamiento, como la tensión en circuito abierto (Voc) y la corriente de cortocircuito (Isc), bajo situaciones distintas a las condiciones estándar de medida (CEM), permitiendo detectar posibles desviaciones en el rendimiento de los paneles de la instalación.

Para garantizar mediciones representativas, estos módulos se instalan en condiciones similares a las del conjunto de paneles solares de la instalación, lo que permite obtener datos precisos sobre las condiciones ambientales y de operación del sistema fotovoltaico. En este caso, los módulos de referencia empleados son como los de la figura 11, los cuales son del tipo ESPMC050 de la marca ATERSA.

Figura 11: Módulo ESPMC050[34]

El uso de módulos fotovoltaicos calibrados como sensores de irradiancia solar no solo representa una opción más económica para instalaciones pequeñas, sino que también ofrece ventajas técnicas significativas [35]. Respecto a su respuesta espectral, esta es más representativa del comportamiento de los propios módulos fotovoltaicos, mejora la precisión en la modelización del rendimiento. A diferencia de los piranómetros, no requieren correcciones espectrales ni angulares, lo que simplifica el monitoreo y minimiza posibles fuentes de error. Además, su tiempo de respuesta más rápido permite obtener mediciones más dinámicas y precisas en aplicaciones específicas. Otro beneficio clave es su capacidad para medir con mayor precisión la temperatura de la célula, superando a métodos como los sensores RTD [28], aunque estos últimos siguen siendo una alternativa más asequible.

4.2 Protocolos de comunicación

Para la interconexión de los diferentes sistemas que actúan dentro del proyecto, se han empleado los siguientes protocolos de comunicación:

4.2.1 Protocolo Modbus RTU

El protocolo Modbus es un estándar de mensajería en la capa de aplicación, correspondiente al nivel 7 del modelo OSI (del inglés *Open Systems Interconnection* o interconexión de sistemas abiertos). Este protocolo facilita la comunicación cliente/servidor o maestro/esclavo entre dispositivos conectados a diferentes tipos de buses o redes. Existen tres variantes principales del protocolo Modbus: Modbus ASCII, Modbus RTU y Modbus TCP/IP. Sin embargo, para este proyecto, solo se considerará Modbus RTU debido a sus características específicas y requerimientos del sistema.

Modbus RTU es un protocolo de comunicación serie asíncrono y de código abierto, diseñado para operar bajo una arquitectura maestro-esclavo [36] [37]. En este sistema, solo el

dispositivo maestro (en este caso el PLC) tiene la capacidad de iniciar la comunicación, mientras que los dispositivos esclavos (inversores y contadores) responden a las solicitudes realizadas por el maestro a través de una conexión serial.

El protocolo Modbus RTU se apoya sobre RS-485, este es un estándar de comunicación que define el modelo eléctrico para la transmisión de datos en serie. RS-485 es ampliamente utilizado en entornos industriales debido a su robustez y capacidad para transmitir datos a largas distancias con alta inmunidad al ruido. A diferencia de RS-232, que permite solo la comunicación punto a punto, RS-485 permite la conexión de múltiples dispositivos en un mismo bus, lo que facilita la implementación de arquitecturas maestro-esclavo como la de Modbus RTU. Este estándar utiliza una señal diferencial balanceada, lo que minimiza las interferencias electromagnéticas y permite alcanzar distancias de hasta 1200 metros con velocidades de transmisión de hasta 10 Mbps. Para garantizar una comunicación fiable, Modbus RTU en este proyecto operará sobre RS-485 con una velocidad de transmisión de 9600 bps, 8 bits de datos, sin paridad y 1 bit de parada (8N1). Estos parámetros han sido elegidos por ser la configuración más estándar y la que es capaz de emplear todos los dispositivos. La velocidad de transmisión de 9600 bps proporciona un equilibrio entre rapidez y fiabilidad, evitando posibles errores debidos a interferencias en largas distancias. Los 8 bits de datos garantizan que se pueda transmitir la información completa sin pérdidas, mientras que la elección de sin paridad simplifica la configuración y reduce la sobrecarga en la comunicación. Finalmente, el uso de 1 bit de parada asegura la correcta finalización de cada trama de datos, facilitando la sincronización entre el maestro y los esclavos. Además, para asegurar la estabilidad de la comunicación, es recomendable instalar resistencias de terminación de 120Ω en ambos extremos del bus RS-485. Esto minimiza las reflexiones de señal y mejora la fiabilidad de la transmisión de datos, especialmente en redes con cables largos o múltiples dispositivos conectados.

La estructura de mensajes de este protocolo sigue un formato binario donde los datos se organizan en una Unidad de Datos de Protocolo (PDU, por sus siglas en inglés), que constituye el núcleo del mensaje. La PDU se combina con un encabezado (dirección del dispositivo esclavo) y una cola (para verificación de errores), formando la ADU (*Aplication Data Unit* o Unidad de Datos de Aplicación) [38]. La estructura de la ADU se muestra en la Figura 12, donde se observan los siguientes campos:

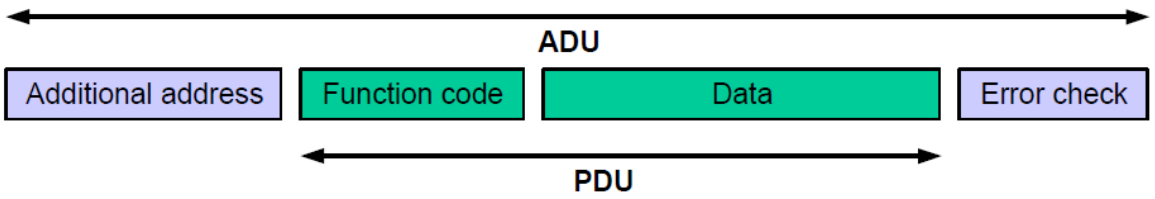

Figura 12: Trama Modbus [38]

- Dirección del Dispositivo (1 byte): Contiene la dirección del esclavo al que se dirige el mensaje. Los valores permitidos son del 1 al 247, mientras que los valores de 248 a 255 están reservados y el 0 se emplea para mensajes de difusión (broadcast).
- Código de Función (1 byte): Define la acción específica que debe realizar el dispositivo esclavo. Por ejemplo, el código 0x03 es empleado para leer múltiples registros y el 0x10 para escribir en múltiples registros.
- Campo de Datos (N bytes): Contiene información adicional requerida por el código de función, como direcciones de memoria, numero de registros a leer o escribir y el valor de estos.
- Comprobación de errores (2 bytes): Es un grupo de bits diseñado para asegurar la integridad de los datos transmitidos o almacenados. Típicamente se usa el mecanismo CRC (Comprobación de redundancia cíclica).

El funcionamiento de este protocolo es muy simple y puede verse reflejado en la figura 13, el dispositivo maestro inicia una comunicación enviando una trama compuesta por los campos anteriores y únicamente el esclavo con la misma dirección que la que tiene el campo dirección de dispositivo de la trama, procesa la solicitud y responde al maestro enviando otra trama con los datos solicitados o una confirmación de la acción realizada empleando el mismo código de función que el maestro empleo en la trama original.

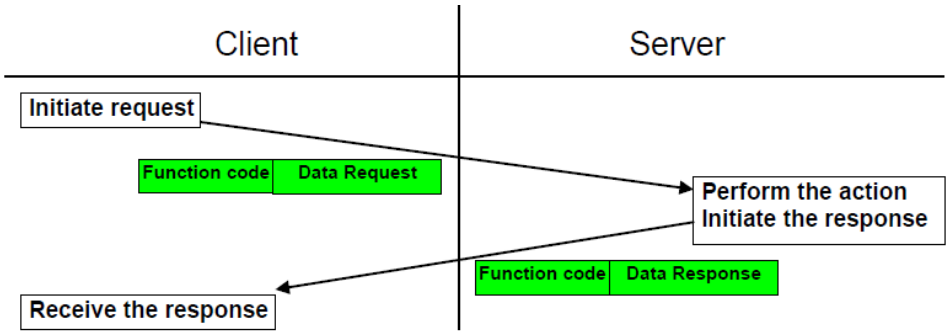

Figura 13: Intercambio de información mediante Modbus [38]

El principal motivo por el cual se seleccionó el protocolo Modbus RTU frente a otros es debido a su independencia de una conexión a internet y su compatibilidad con la interfaz eléctrica RS-485, utilizada por todos los dispositivos mencionados previamente. Empleando este protocolo, se puede crear un único bus compuesto por 2 cables (A+ y B-) que interconecte el contador, inversor y PLC. Cuando se quieran leer los datos, el PLC únicamente tendrá que enviar la trama con la dirección del dispositivo en concreto y este le responderá sin involucrar en la comunicación al otro dispositivo esclavo.

4.2.2 Protocolo MQTT

El protocolo MQTT (Message Queuing Telemetry Transport o Transporte de telemetría de cola de mensajes) es una tecnología de comunicación diseñada específicamente para facilitar el intercambio de mensajes entre dispositivos en sistemas de comunicación máquina a máquina (M2M, por sus siglas en inglés). Este protocolo está orientado a la transferencia de datos en

tiempo real, haciendo que sea especialmente adecuado para entornos donde la conectividad puede ser intermitente [39].

Entre las características más destacadas de MQTT se incluyen:

- Modelo de mensajería basado en colas: Permite que las aplicaciones se comuniquen sin necesidad de mantener una conexión directa o constante, incluso cuando se ejecutan en diferentes momentos o plataformas.
- Uso de la pila TCP: MQTT utiliza el protocolo TCP (*Transmission Control Protocol*) como base para garantizar una transmisión de datos confiable, incluyendo mecanismos de control de errores y entrega en orden. Cada conexión TCP se mantiene abierta durante toda la sesión, reutilizando eficientemente el canal para reducir la sobrecarga de establecer y cerrar conexiones repetidamente.
- Medidas de seguridad: Incluye soporte para SSL/TLS para cifrar las comunicaciones, autenticación mediante usuario y contraseña, y control de acceso a los mensajes.
- Calidad de Servicio (QoS): Proporciona tres niveles de QoS que determinan las garantías de entrega entre el cliente y el bróker:
  -QoS 0: (Como máximo una vez) El mensaje se envía una única vez sin confirmación de recepción. No se garantiza la entrega, y los mensajes pueden perderse si ocurre un fallo durante la transmisión. Este nivel es adecuado cuando la pérdida ocasional de mensajes es aceptable y se prioriza la eficiencia.
  -QoS 1: (Al menos una vez) El mensaje se envía repetidamente hasta que el remitente recibe una confirmación de recepción (ACK) del receptor. Esto asegura que el mensaje llegue al menos una vez, aunque pueden ocurrir duplicados en el receptor. Es útil cuando es crucial que el mensaje llegue, y el receptor puede manejar duplicados.
  -QoS 2: (Exactamente una vez) Se implementa un protocolo de *handshake* de dos niveles entre el remitente y el receptor para garantizar que cada mensaje se entregue una y solo una vez. Este nivel es el más seguro en términos de entrega, pero también el más costoso en términos de recursos y tiempo debido al mayor número de intercambios necesarios.

Además, MQTT se basa en un modelo de mensajería conocido como *publisher/subscriber* o pub-sub. Este modelo, se diferencia del esquema cliente-servidor al eliminar la necesidad de que los dispositivos interesados en un intercambio de información mantengan una comunicación directa entre ellos. En su lugar, los mensajes se envían a través de un intermediario central denominado *broker*. Este modelo presenta las siguientes entidades principales [40]:

- *Publishers*: Son los dispositivos que generan y envían mensajes con información. Por ejemplo, en la aplicación desarrollada, el PLC actúa como *publisher* enviando los datos de forma periódica al bróker instalado en el ordenador.
- *Subscribers*: Son los dispositivos o aplicaciones que reciben mensajes en función de sus intereses. En el proyecto desarrollado, la aplicación Node-Red será la encagada de extraer los mensajes del *broker* para procesar los datos.

- *Broker*: Es el servidor central responsable de gestionar las comunicaciones entre publicadores y suscriptores. El *broker* recibe los mensajes de los *publishers* y los reenvía únicamente a los *subscribers* que han mostrado interés en esos datos.

El funcionamiento del protocolo se encuentra ilustrado en la figura 14. Los dispositivos interactúan entre si mediante *topics* (temas), que son canales lógicos utilizados para organizar y clasificar los mensajes. Los *publishers* envían mensajes asociados a un *topic* específico, pudiendo ser estos de cualquier tipo, mientras que los *subscribers* se suscriben a uno o varios *topics* para recibir la información. Esto permite que los dispositivos reciban únicamente los mensajes que les interesan, siendo el *broker* el encargado de hacer llegar los mensajes a los *subscribers* únicamente de los *topics* a los que están suscrito [41].

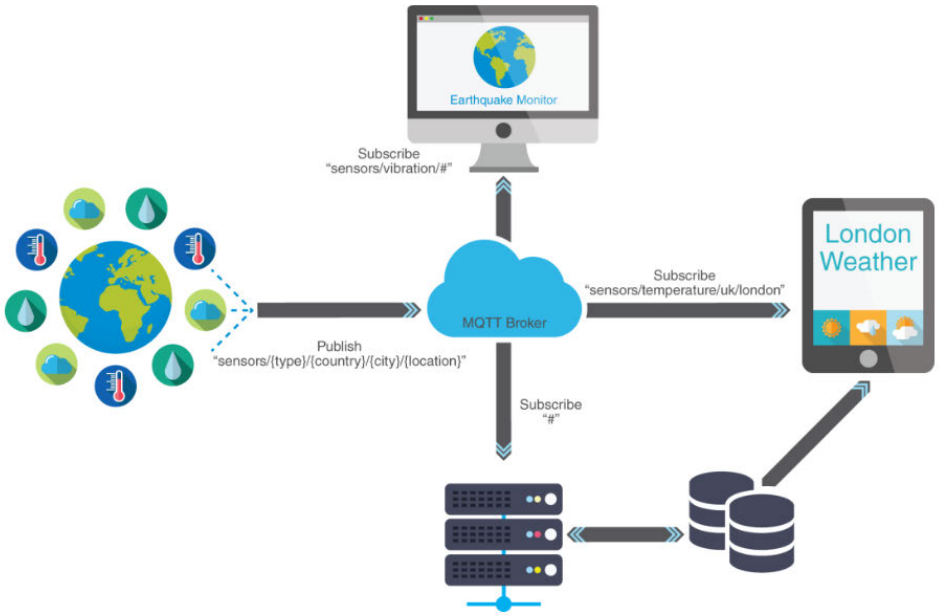

Figura 14: Esquema de comunicación en MQTT[42]

El protocolo MQTT fue el protocolo seleccionado para enviar los datos desde el PLC al *broker* Mosquitto alojado en el ordenador debido a sus múltiples ventajas. En primer lugar, MQTT está diseñado específicamente para dispositivos con recursos limitados, como los PLCs, y para entornos donde la red puede ser inestable. Además, su arquitectura se basa en mensajes ligeros, lo que reduce significativamente el uso de ancho de banda y optimiza el rendimiento en sistemas con limitaciones de comunicación. Otra característica clave es su capacidad para reconectarse automáticamente en caso de pérdida de conexión, reanudando la transmisión de datos y ofreciendo opciones de persistencia para garantizar que la información no se pierda. Estas características hacen que MQTT sea una opción más eficiente y confiable frente a otros protocolos como HTTP [43], justificando su implementación en este proyecto.

4.3 Software

En esta sección se describen las herramientas de software utilizadas durante el desarrollo del proyecto explicando su propósito y características principales.

4.4.1 Arduino IDE

Arduino *Integrated Development Environment* (IDE) es una plataforma de desarrollo de código abierto diseñada para programar dispositivos compatibles con Arduino [44]. Disponible para sistemas operativos como macOS, Windows y Linux, proporciona un entorno intuitivo que permite escribir, compilar y cargar código en microcontroladores.

El código se escribe en los lenguajes C/C++ dentro de archivos con extensión .ino llamados *sketches*. Cada *sketch* debe contener, al menos, una función de inicialización llamada setup() y una función principal denominada loop(), además de las declaraciones de funciones y variables que el usuario desee. El IDE de Arduino está compuesto por un editor de código, un compilador, un depurador y una interfaz gráfica de usuario (GUI) lo que facilita el desarrollo y corrección de errores en los proyectos. La figura 15 incluye un ejemplo de un programa encargado de leer una entrada analógica.

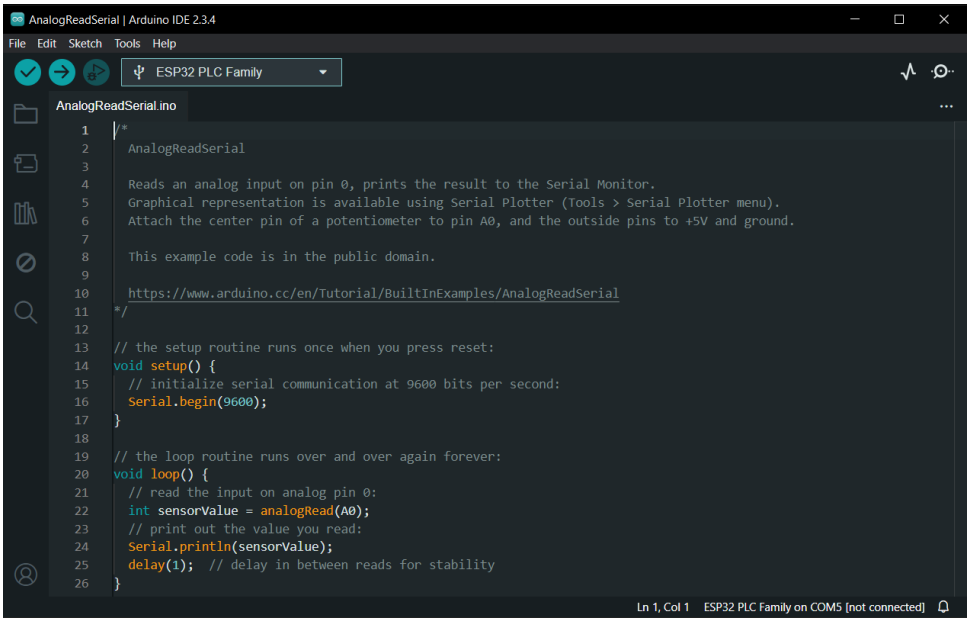

Figura 15: Ejemplo de sketch en Arduino IDE

La elección de esta aplicación como plataforma de desarrollo del código se debe a la recomendación de los propios fabricantes del PLC. Además, gracias a su carácter gratuito y a la activa comunidad que la respalda, es posible programar la placa sin necesidad de poseer amplios conocimientos de programación. El lenguaje es sencillo y existen innumerables bibliotecas adaptadas a diversas necesidades, lo que facilita la implementación de funcionalidades específicas en los proyectos.

4.4.2 Eclipse Mosquitto

Eclipse Mosquitto es un *broker* MQTT de código abierto y alta eficiencia diseñado para facilitar la comunicación en aplicaciones de Internet de las Cosas (IoT) [45]. Es ampliamente conocido y está disponible para sistemas operativos como Windows, macOS y diversas distribuciones de Linux. Para ello Mosquitto implementa las versiones 5.0, 3.1.1 y 3.1 del protocolo MQTT, proporcionando un método ligero para la mensajería mediante un modelo de publicación/suscripción. Su bajo consumo de recursos lo hace adecuado para dispositivos con recursos limitados, como sensores de baja potencia, teléfonos móviles o microcontroladores como Arduino.

Además, Mosquitto proporciona una biblioteca en C para implementar clientes MQTT, así como las herramientas de línea de comandos mosquitto_pub y mosquitto_sub para publicar y suscribirse a mensajes respectivamente desde la terminal de comandos del sistema que se esté empleando.

La elección de Eclipse Mosquitto y su preferencia sobre otros *brokers* como HiveMQ para este proyecto se debe a su facilidad de uso, a la abundante información disponible debido a que la mayoría de los proyectos IoT lo usan y a su ligereza, que lo hace adecuado para dispositivos de baja potencia como en este caso.

4.4.3 Node-RED

Node-RED es una herramienta de programación visual basada en flujos para la creación de aplicaciones de IoT (Internet de las Cosas), automatización y procesamiento de datos [46]. Fue desarrollada por IBM y permite a los usuarios conectar dispositivos y servicios de manera sencilla utilizando un enfoque de "arrastrar y soltar".

Esta plataforma proporciona un editor al que se accede desde un navegador web, facilitando la creación de flujos mediante una amplia gama de cajas o nodos predefinidos con propósitos específicos, como recibir, transformar y enviar datos. Los usuarios pueden arrastrar y soltar estos nodos para construir aplicaciones que integren diversos servicios y dispositivos sin necesidad de escribir código extenso. Además, Node-RED permite la creación de funciones personalizadas en JavaScript, ofreciendo flexibilidad para adaptarse a necesidades específicas.

Una característica destacada de Node-RED es su capacidad para integrarse con una variedad de protocolos y servicios, incluyendo MQTT, HTTP, WebSockets y bases de datos, lo que la convierte en una herramienta versátil para el desarrollo de soluciones IoT. La paleta de nodos puede ser fácilmente ampliada instalando nuevos nodos creados por la comunidad. Además, los flujos creados son almacenan en formato JSON, lo que facilita su exportación, intercambio y reutilización en diferentes proyectos. En la figura 16 puede verse un ejemplo de un diagrama de flujo en Node-Red.

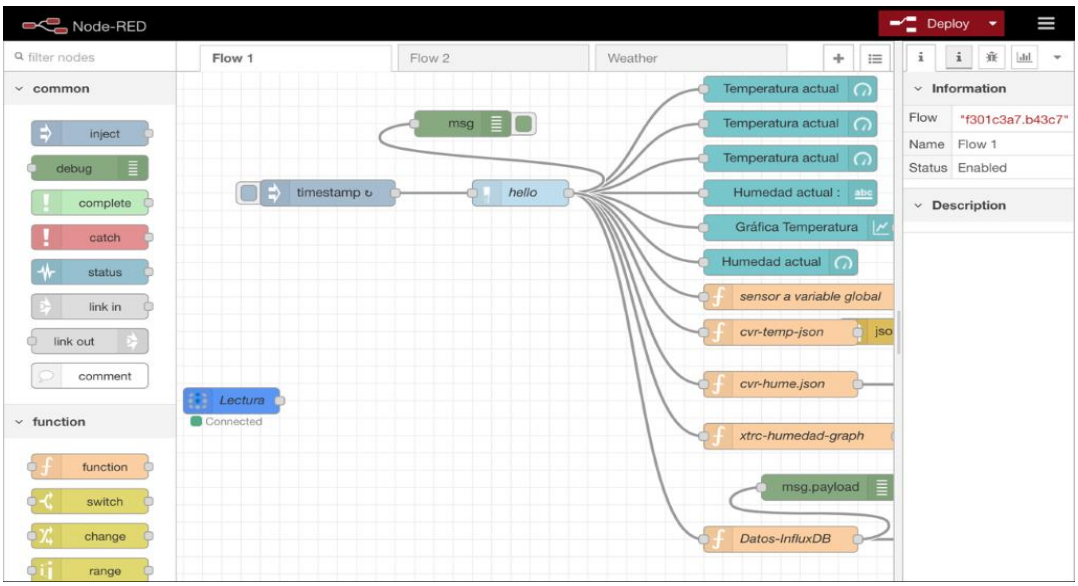

Figura 16: Ejemplo de flujo en Node-RED

La elección de Node-RED en este proyecto se debe a su facilidad de uso, flexibilidad y la amplia comunidad de desarrolladores que la respalda, siendo compatible con el resto de los programas que se emplearan.

4.4.4 InfluxDB

InfluxDB es una base de datos NoSQL de series temporales de código abierto, diseñada específicamente para gestionar y analizar grandes volúmenes de datos con marcas de tiempo (*timestamp*), como los generados por sensores y dispositivos en aplicaciones IoT [47]. Es compatible con múltiples sistemas operativos, incluyendo distribuciones basadas en Raspberry Pi OS FreeBSD, macOS y Windows. Además, existe una versión en la nube llamada InfluxDB Cloud.

Las bases de datos de series temporales, también conocidas como *Time Series Databases* (TSDB), son sistemas de software optimizados para almacenar y gestionar datos indexados por tiempo. Una serie temporal es una colección de puntos de datos recopilados en intervalos sucesivos y registrados en orden cronológico. Estas bases de datos permiten clasificar grandes y complejas cantidades de datos, haciendo que la información sea más accesible que si se almacena en una base de datos tradicional. En las TSDB, los datos se insertan de manera continua, sin que sea posible realizar actualizaciones a los datos que ya se encuentran en la misma, facilitando el seguimiento y análisis de tendencias a lo largo del tiempo.

InfluxDB emplea el motor de almacenamiento *Time-Structured Merge Tree* (TSM), que optimiza la escritura secuencial y permite la compactación eficiente de datos. Este enfoque reduce la cantidad de almacenamiento necesario y mejora la velocidad de consulta, logrando un alto rendimiento en la gestión de datos indexados por tiempo. En InfluxDB, los datos se organizan en *buckets* que son ubicaciones nombradas donde se almacena la información de series temporales. Cada *bucket* tiene políticas de retención de datos, lo que permite definir el tiempo durante el cual se conservará la información.

Para realizar consultas, InfluxDB proporciona un lenguaje de consulta similar a SQL, denominado InfluxQL, que facilita la realización de análisis complejos sobre los datos temporales como puede verse en la figura 17. Además, la interfaz de usuario de InfluxDB ofrece herramientas para construir *dashboards* personalizados, permitiendo la visualización en tiempo real de los datos a través de diversos tipos de gráficos y paneles. Estos *dashboards* son configurables y pueden exportarse o importarse en formato JSON a distintos proyectos.

Figura 17: Ejemplo estructura de consultas en influxDB con Flux

La elección de InfluxDB en este proyecto se debe a su capacidad para gestionar eficientemente grandes volúmenes de datos de series temporales, superando en rendimiento a las bases de datos tradicionales como MySQL en este ámbito. Razones como su flexibilidad para adaptarse a diferentes casos de uso y su simplicidad de uso también fueron motivos determinantes además de ofrecer integración nativa con Node-RED y Grafana.

4.4.5 Grafana

Grafana es una plataforma de código abierto diseñada para la visualización y el análisis de datos en tiempo real. Puede ser implementada localmente en sistemas operativos como Ubuntu, macOS y Windows, o en configuraciones híbridas que combinan nubes privadas y públicas con el servicio Grafana Cloud. Permite a los usuarios crear y agrupar paneles interactivos y personalizables denominados *dashboards*, que facilitan la interpretación de métricas, registros y otros tipos de datos provenientes de diversas fuentes. Además, ofrece la funcionalidad de crear alertas útiles para la monitorización [48].

Una de las principales fortalezas de Grafana es su capacidad para integrarse con una amplia variedad de fuentes de datos, incluyendo bases de datos SQL y NoSQL, sistemas de monitoreo como Prometheus y Graphite, y servicios en la nube como AWS CloudWatch y Google Stackdriver. La plataforma proporciona diversas opciones de visualización como puede verse en la figura 18, como gráficos de líneas, barras y mapas de calor, lo que permite adaptar la presentación de los datos a las necesidades específicas de cada usuario. Además, los *dashboards* pueden ser exportados e importados fácilmente en formato JSON facilitando su reutilización y compartición.

Figura 18: Ejemplo dashboard en Grafana

La decisión de emplear Grafana en este proyecto se debe a su capacidad para proporcionar una visualización clara y concisa de los datos almacenados en InfluxDB, permitiendo la creación de paneles de control que facilitan el análisis de los datos.

4.4.6 Tailscale

Tailscale es una solución de red privada virtual (VPN) que simplifica la creación de conexiones seguras entre dispositivos a través de Internet [49]. Basada en el protocolo de código abierto WireGuard, Tailscale permite establecer redes privadas sin la complejidad de las VPN tradicionales. El servicio es compatible con múltiples sistemas operativos, incluyendo Windows, macOS, Linux, Android, iOS y dispositivos basados en ARM como Raspberry Pi.

Una de las características más destacadas de Tailscale es su capacidad para establecer conexiones punto a punto sin necesidad de abrir puertos del *router*. Esto se logra mediante técnicas de NAT (*Network Address Translation*) traversal y el uso de servidores DERP (*Designated Encrypted Relay for Packets*), que facilitan la comunicación incluso cuando los dispositivos están detrás de NAT permitiendo que los dispositivos se conecten entre sí sin estar en la misma red como puede verse en la figura 19.

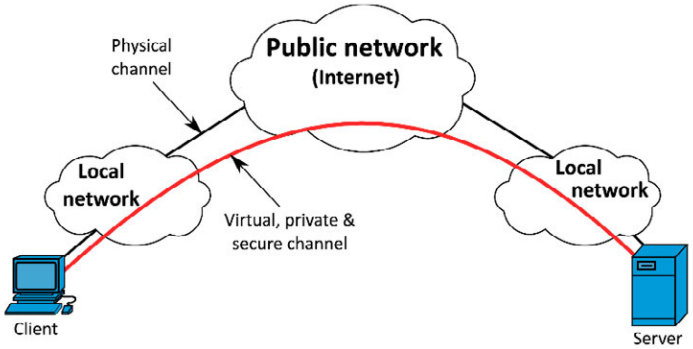

Figura 19: Ejemplo conexión VPN

La elección de Tailscale en este proyecto se debe a su capacidad para crear una red segura y privada entre dispositivos remotos, facilitando el acceso a programas alojados en un ordenador que actúa como servidor local desde otros dispositivos, sin la necesidad de configuraciones complejas o la apertura de puertos en el *router*. Esto es especialmente útil ya que el *router* que estamos empleando en el laboratorio no permite la configuración de VPNs tradicionales o la apertura de puertos necesarios para su funcionamiento.

4.4 Desarrollo de la solución

En esta sección del proyecto, se detallará el proceso de desarrollo de la solución propuesta, explicando las decisiones tomadas desde la obtención de los datos hasta la visualización de los mismos empleando las herramientas mencionadas anteriormente.

4.4.1 Extracción de datos del inversor y contador

Una vez puestos en marcha el inversor y el contador, y tras verificar el correcto funcionamiento de ambos dispositivos, se procedió a conectarlos al PLC siguiendo las indicaciones detalladas en el Manual de Usuario incluido en el anexo de este documento.

La comunicación entre el PLC, el inversor y el contador se realizó mediante el protocolo Modbus RTU, que emplea una arquitectura maestro-esclavo. En esta configuración, el PLC opera como maestro, siendo el encargado de iniciar todas las solicitudes de datos, mientras que el inversor y el contador actúan como esclavos, respondiendo exclusivamente a las consultas específicas dirigidas a ellos. Cada dispositivo en el sistema cuenta con una dirección única dentro del bus Modbus, lo que permite al PLC identificar y comunicarse de manera individual con cada uno de ellos.

Para extraer la información de los dispositivos, se utilizó la función 0x03, definida en el estándar Modbus como la encargada de leer los registros de retención (*holding registers*). Estos registros contienen los datos clave a monitorizar, como parámetros eléctricos y de estado. La ubicación exacta de cada registro fue obtenida de los documentos específicos de cada dispositivo, denominados mapas de registros Modbus, proporcionados por los fabricantes. Estos documentos detallan el mapeo de registros, es decir, las direcciones de memoria asociadas a cada parámetro disponible para lectura.

Es importante destacar que cada parámetro a leer puede ocupar más de un registro consecutivos. Afortunadamente, la función 0x03 permite leer múltiples registros contiguos en una sola solicitud, facilitando la recolección eficiente de datos.

Para garantizar una comunicación fluida y confiable entre el PLC y los dispositivos conectados, se emplearon dos librerías fundamentales que permiten implementar el protocolo Modbus RTU sobre la interfaz RS485:

- ModbusRTUMaster.h: Esta librería proporciona las herramientas necesarias para implementar comunicación maestro-esclavo basada en el protocolo Modbus RTU. Permite configurar parámetros de comunicación, como velocidad (*baudrate*) y paridad, y ofrece funciones para leer y escribir registros en dispositivos esclavos. En el código, esta librería se utiliza para establecer la comunicación entre el PLC y los dispositivos como el inversor y el contador, enviando solicitudes y gestionando respuestas de manera eficiente.
- RS485.h: Es la encargada de manejar la interfaz RS485 del hardware, esencial para la comunicación serie *half-duplex* en entornos industriales. Facilita la transmisión y recepción de datos entre el PLC y los dispositivos conectados, gestionando automáticamente el control de dirección en el bus RS485. En el código, se inicializa con los parámetros necesarios para interactuar con los esclavos en la red Modbus.

Por otro lado, el proceso de lectura de registros Modbus en el código se divide en varias funciones clave que garantizan la extracción, validación y manejo eficiente de los datos:

- executeModbusReadings(), ver Figura 20: Esta función se encarga de orquestar el proceso completo de lectura Modbus, recorriendo los diferentes registros configurados. Utiliza un ciclo para enviar solicitudes mediante sendModbusRequest() y esperar respuestas válidas con waitForModbusResponse(). Si ocurre un error en el proceso, se invoca handleRetry() para gestionar los intentos fallidos. Las variables clave son currentRegister, que rastrea el registro actual, y retryCount, que controla el número de intentos realizados. Este flujo garantiza que cada registro sea leído de manera eficiente y que, en caso de fallo, el programa no se quede bloqueado.

```
246  void executeModbusReadings() {
247    currentRegister = 0;
248    retryCount = 0;
249
250    while (currentRegister < MODBUS_READS) {
251      if (sendModbusRequest()) {
252        if (!waitForModbusResponse()) {
253          handleRetry();
254        }
255      } else {
256        Serial.println("Error al enviar solicitud Modbus.");
257        handleRetry();
258      }
259    }
260  }
```

Figura 20: Función executeModbusReadings()

- sendModbusRequest(), véase Figura 21: Esta función envía solicitudes Modbus a dispositivos esclavos utilizando la biblioteca ModbusRTUMaster.h. Hace uso de la función master.readHoldingRegisters(), que permite leer registros de retención en los dispositivos esclavos especificando la dirección Modbus del dispositivo, la dirección inicial del registro y la cantidad de registros a leer.

```
bool sendModbusRequest() {
  switch (currentRegister) {
    case 0:
      return master.readHoldingRegisters(DIR_MODBUS_INV, DIR_REG_VOLT_CURR_DC_INV, FOUR_REGISTER_TO_READ);
    case 1:
      return master.readHoldingRegisters(DIR_MODBUS_INV, DIR_REG_PDC_INV, TWO_REGISTER_TO_READ);
    case 2:
      return master.readInputRegisters(DIR_MODBUS_CONTADOR, DIR_REG_ACTIVE_POWER_CONT, ONE_REGISTER_TO_READ);
  }
  return false;
}
```

Figura 21: Función sendModbusRequest()

- waitForModbusResponse(), consultar Figura 22: Esta función espera una respuesta válida a la solicitud Modbus enviada previamente. Monitorea continuamente el estado del maestro (master.isWaitingResponse()) y, si se recibe una respuesta, verifica si tiene errores o contiene datos válidos. En caso de respuesta válida, invoca processModbusResponse() para procesar los datos; si ocurre un error o se excede el tiempo de espera (MODBUS_TIMEOUT_MS), se registra un mensaje de error y devuelve false, indicando un fallo en la operación.

```
bool waitForModbusResponse() {
  unsigned long startTime = millis();
  while (millis() - startTime < MODBUS_TIMEOUT_MS) {
    if (master.isWaitingResponse()) {
      ModbusResponse response = master.available();
      if (response) {
        if (response.hasError()) {
          Serial.print("Error en la respuesta: ");
          Serial.println(response.getErrorCode());
          return false;
        } else {
          processModbusResponse(response);
          return true;
        }
      }
    }
  }
  Serial.println("Timeout esperando respuesta.");
  return false;
}
```

Figura 22: Función waitForModbusResponse()

- processModbusResponse(ModbusResponse response), tal como se aprecia en la Figura 23: Procesa las respuestas válidas de las solicitudes Modbus, extrayendo y escalando los datos para almacenarlos en variables globales. Al finalizar, incrementa currentRegister para avanzar al siguiente registro y continuar con la iteración.

```
void processModbusResponse(ModbusResponse response) {
  switch (currentRegister) {
    case 0:
      for (int i = 0; i < MPPT; i++) {
        Vdc_MPPT[i] = response.getRegister(i * 2) * VOLTAGE_SCALING_FACTOR;
        Idc_MPPT[i] = response.getRegister(i * 2 + 1) * CURRENT_SCALING_FACTOR;
      }
      break;
    case 1:
      Pdc_inv = ((response.getRegister(0) << 16) | response.getRegister(1)) * POWER_SCALING_FACTOR;
      break;

    case 2:
      Pac_cont = response.getRegister(0) * ACTIVE_POWER_SCALING_FACTOR;
      break;
  }

  currentRegister++;
}
```

Figura 23: Función processModbusResponse()

- handleRetry(), representada en la Figura 24: Esta función gestiona los reintentos en caso de errores durante la comunicación Modbus. Si el número de intentos no ha alcanzado el máximo permitido (MAX_RETRIES), incrementa retryCount y registra un mensaje indicando que se reintentará. Si los reintentos se agotan, avanza al siguiente registro incrementando currentRegister y reinicia el contador de intentos, asegurando que el proceso continúe con el próximo registro sin bloquearse.

```
342 void handleRetry() {
343   if (retryCount < MAX_RETRIES) {  // Si no se ha alcanzado el numero maximo de reintentos, se vuelve a intentar la lectura
344     retryCount++;                  //Actualizamos reintentos
345     Serial.println("Reintentando solicitud...");
346   } else {
347     Serial.println("Error: No se obtuvo respuesta tras varios intentos,pasamos a la siguiente lectura.");
348     currentRegister++;  // Pasa al siguiente registro si se agotaron los reintentos
349     retryCount = 0;     // Pone a 0 el contador de reintentos
350   }
351 }
```

Figura 24: Función handleRetry()

4.4.2 Lectura de las entradas analógicas del módulo de referencia

Para realizar el cálculo de las medidas de irradiancia (G) y temperatura de célula (T_C), fue necesario medir la tensión en circuito abierto (V_{OC}) y la corriente de cortocircuito (I_{sc}) provenientes de los módulos de referencia. Estos módulos, habían sido empleados en otro proyecto donde fueron calibrados con los valores de la tabla 1. En el laboratorio donde se desarrolló el proyecto, ambas señales (V_{OC} e I_{sc}) llegaban a través de un bornero diseñado para tal fin.

Tabla 1: Valores de los parámetros de los módulos de referencia calibrados

	Isc* (A)	Voc* (V)	α (%/°C)	β (%/°C)	β (V/°C)	Shunt
Módulo 872 (G sensor)	2.85	–	0.165	–	–	$I = 2.5$ A $V = 156.3$ mV
Módulo 869 (Tc sensor)	–	22.341	–	−0.287	−0.064	–

La corriente de cortocircuito fue convertida en un voltaje proporcional mediante una resistencia shunt (R_{shunt}). Este voltaje (al que se le llamo V_{shunt}), del orden de milivoltios, estaba dentro del rango de entrada analógica del PLC y no requería acondicionamiento adicional.

La tensión en circuito abierto era considerablemente mayor, rondando los 20 V, lo que excedía el rango máximo de 0 a 10 V soportado por las entradas analógicas del PLC (V_{PLC}) que el propio fabricante declara en su *datasheet*. Para solucionar este problema, se decidió implementar un divisor de tensión que redujera el voltaje a un nivel compatible con el PLC. Este divisor de tensión permitió acondicionar la señal, preservando su proporción original y asegurando una lectura precisa por parte del PLC. Posteriormente, en la realización de la programación del código fue necesario tener esto en cuenta y adaptar el valor medido por el PLC al valor real de la señal.

El diseño del divisor de tensión se basó en la ecuación (1), empleada para calcular V_{oc} en V en función de T_c:

$$V_{OC} = V_{OC}^{*}\,[1 + \beta * (T_c - T_C^{*})] \qquad V \qquad (1)$$

Donde:

- V_{OC}^{*} : Tensión en circuito abierto calibrada.
- β: Coeficiente de temperatura de la célula.
- T_C^{*}: Temperatura de célula de referencia (25ºC).

Para determinar los valores del divisor, se calculó V_{oc} considerando un rango de variación de la Tc de entre -10 °C y 100 °C. En este análisis se observó que V_{oc} aumenta a medida que T_c disminuye, alcanzando su valor máximo en condiciones extremas de baja temperatura. Asumiendo una T_c hipotética de -25 °C, un caso poco realista pero conservador, se estimó que V_{oc} podría alcanzar un valor máximo de 25,55 V. Para simplificar los cálculos, se utilizó un valor de 25 V como referencia de diseño.

El divisor de tensión debía reducir esta V_{oc} máxima de 25 V al rango soportado por el PLC, es decir, 10 V. Esto se modeló mediante la ecuación (2) del divisor de tensión:

$$V_{PLC\_max} = V_{OC} * \frac{R_2}{R_1 + R_2} \qquad V \qquad (2)$$

De donde se obtiene la relación:

$$\frac{R_2}{R_1 + R_2} = \frac{V_{PLC}}{V_{OC}} = \frac{10}{25} = 0{,}4$$

Pudiendo elegir el valor de resistencias para R_1 y R_2 que se desee siempre que cumpla la expresion anterior como por ejemplo $R_1 = 15\ k\Omega$ y $R_2 = 10\ k\Omega$.

Una vez que ya le llegan las señales de V_{OC} e I_{sc} al PLC , se emplean estas para calcular la T_c y G respectivamente.Para el calculo de la T_c , es posible utilizar la ecuación (1) anteriormente empleada y despejar de ella la T_c en ºC quedando la siguiente ecuacion (3):

$$T_c = T_C^{*} + \frac{1}{\beta} * \left(\frac{V_{OC}}{V_{OC}^{*}} - 1\right) \qquad ºC \qquad (3)$$

Para el cálculo de G, se emplea la expresión (4) que permite calcular la intensidad ante variaciones de la irradiancia y la temperatura de célula [50]:

$$I_{SC} = I_{SC}^{*} * \left(1 + \alpha\ *\ (T_c - T_C^{*})\right) * \frac{G}{G^{*}} \qquad A \qquad (4)$$

Donde:

- I_{SC}^{*}: Corriente de cortocircuito en condiciones estándar.

- G^*: Irradiancia en condiciones estándar (1000W/m^2).
- α: Coeficiente de temperatura de la corriente.

Donde al despejar la irradiancia y reemplazar Isc por $I_{SC} = \frac{V_{shunt}}{R_{shunt}}$ nos queda la expresión (5) con el cálculo de G:

$$G = G^* * \frac{V_{shunt}}{I_{SC}^* * R_{shunt}} * \frac{1}{1 + \alpha * (T_c - T_C^*)} \qquad W/m^2 \qquad (5)$$

Para implementar el cálculo de T_c y G en el código se creó la función readGandTc() tal y como se aprecia en la Figura 25, la cual se encarga de calcular los valores de T_c y G basándose en las señales analógicas de entrada (V_{oc} y V_{shunt}) medidas por el PLC. Primero, ajusta V_{oc} utilizando el divisor de tensión y calcula T_c empleando la ecuación (3). Luego, determina I_{SC} dividiendo el voltaje medido en la resistencia shunt entre su valor conocido (R_{shunt}) para finalmente calcular G con la ecuación (5). Los valores finales se redondean a dos decimales para mayor claridad en los datos procesados

```
void readGandTc() {
  int analog1 = analogRead(ANALOG_INPUT_1);
  int analog2 = analogRead(ANALOG_INPUT_2);
  float Vshunt = (analog1 * VOLTAGE_SCALE_ANALOG_PIN) / ADC_RESOLUTION;
  float Vplc = (analog2 * VOLTAGE_SCALE_ANALOG_PIN) / ADC_RESOLUTION;
  // Ajustar voltaje dividido por el factor de atenuación de la entrada
  float Voc_measured = Vplc / DIV_TENS;
  // Calcular temperatura de la célula (Tc)
  float fraction = (Voc_measured / Voc_cal) - 1;  // (V_oc / V_oc^*) - 1
  Tc = T_cem + (1 / beta) * fraction;             // T_c^* + (100/β) * ((V_oc/V_oc^*) - 1)
  // Calcular corriente de cortocircuito (Isc)
  float Isc = Vshunt / RSHUNT;
  // Ajuste de temperatura
  float adjustment = 1 / (1 + (alfa * (Tc - T_cem)));
  G = (G_cem * Isc * adjustment) / Isc_cal;
  //Redondeo a 2 decimales
  G = round(G * 100) / 100.0;
  Tc = round(Tc * 100) / 100.0;
}
```

Figura 25: Función readGandTc()

4.4.3 Modulo RTC del PLC

Este módulo integrado en el PLC es una herramienta esencial para proporcionar una temporización precisa y confiable al sistema de monitorización. Dicho componente, permite al PLC realizar lecturas de datos cada 5 minutos, asegurando una periodicidad constante e independiente de la conectividad a internet o de otros dispositivos externos. La integración de este módulo es fundamental para garantizar la correcta correlación temporal de los datos capturados, lo que resulta crucial para el análisis histórico y la comparación de eventos en el sistema FV.

En caso de que el sistema esté conectado a internet, la hora del RTC se sincroniza automáticamente con un servidor NTP (*Network Time Protocol*) como “es.pool.ntp.org”, garantizando una hora actualizada y precisa. Esto proporciona redundancia al sistema: el RTC

mantiene la hora local en caso de fallos en la conectividad, mientras que la sincronización NTP asegura una precisión constante cuando la red está disponible.

Para implementar el módulo RTC en el sistema, se emplearon las librerías RTClib.h y time.h, que ofrecen herramientas completas y flexibles para gestionar la hora, programar eventos temporales y manejar las funcionalidades del reloj en tiempo real. Se optó por estas librerías en lugar de la proporcionada por el fabricante del PLC, ya que, aunque ambas están diseñadas para trabajar con el mismo tipo de RTC (DS1307), RTClib.h y time.h son más completas, con una mayor compatibilidad y funciones avanzadas que facilitan el desarrollo. Más detalladamente:

- RTClib.h: Desarrollada por Adafruit [51], está diseñada específicamente para manejar dispositivos de reloj en tiempo real como los modelos DS1307 y DS3231. Esta librería permite inicializar y configurar el RTC de manera sencilla, ofreciendo funciones avanzadas para la obtención y manipulación de fechas y horas. Sus capacidades incluyen la generación de marcas de tiempo en múltiples formatos, cálculos de intervalos temporales y manejo estructurado de información temporal.
- time.h: Es una librería estándar de temporización en sistemas C/C++ y está mantenida por Arduino en su plataforma de desarrollo. Proporciona funciones para gestionar estructuras de tiempo y realizar cálculos basados en segundos desde la época UNIX (1 de enero de 1970). time.h permite realizar operaciones como el manejo de diferencias temporales, formateo avanzado de marcas de tiempo y sincronización con otras fuentes de tiempo.

La integración del módulo RTC en el sistema se logró mediante dos funciones principales:

- RTCSetup(), se ilustra en la Figura 26: Inicializa el RTC y ajusta la hora a la de compilación del código. Además, invoca a la función syncRTCWithNTP() para realizar una sincronización con un servidor NTP si hay conectividad a internet, asegurando que la hora inicial del RTC sea precisa desde el inicio del sistema.

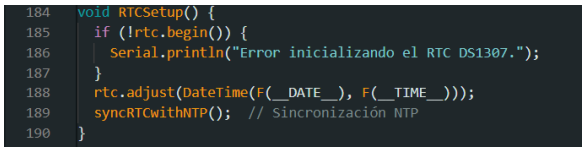

Figura 26: Función RTCSetup()

- syncRTCWithNTP(), representada en la Figura 27: Configura la sincronización del RTC con un servidor NTP, como "es.pool.ntp.org", utilizando la función configTime() para establecer la hora UTC (*Universal Time Coordinated*) sin ajustes de zona horaria. Posteriormente, utiliza getLocalTime() para obtener la hora actual desde el servidor NTP y, si la conexión es exitosa, ajusta el RTC con los valores obtenidos mediante rtc.adjust(). En caso de falla en la conexión, se emite un mensaje de error para identificar el problema.

```
384  void syncRTCwithNTP() {
385    //Serial.println("Sincronizando con el servidor NTP...");
386    struct tm time;
387    configTime(0, 0, ntpServer);  // Sin ajuste de zona horaria
388    if (!getLocalTime(&time)) {
389      Serial.println("Error al obtener la hora de NTP.");
390    } else {
391
392      // Ajustar el RTC con la hora de NTP
393      rtc.adjust(DateTime(
394        time.tm_year + 1900,  // Ajuste del año (se suma 1900 porque tm_year es años desde 1900)
395        time.tm_mon + 1,      // Mes (tm_mon es 0-11, por eso se suma 1)
396        time.tm_mday,
397        time.tm_hour,
398        time.tm_min,
399        time.tm_sec));
400      Serial.println("RTC sincronizado con NTP.");
401    }
402  }
```

Figura 27: Función syncRTCWithNTP()

En el bucle principal del programa como se muestra en la Figura 28, se verifica cada segundo si se cumple el intervalo de tiempo configurado (5 minutos) utilizando las lecturas del RTC. Si la condición se cumple, se actualiza el último minuto procesado para evitar redundancias, y se sincroniza nuevamente con el servidor NTP en caso de que haya conexión Wi-Fi. Además, se ejecutan las tareas principales del sistema, como realizar lecturas Modbus de los inversores y el contador, capturar valores de irradiancia y temperatura, enviar los datos en formato JSON al servidor y almacenar los mismos en la tarjeta SD como respaldo.

```
136    now = rtc.now();
137    if (now.minute() % TIME_BETWEEN_READS == 0 && now.minute() != lastMinute) {
138      lastMinute = now.minute();  // Actualizar el último minuto procesado
139      if (WiFi.status() == WL_CONNECTED) {
140        syncRTCwithNTP();
141      }
142      executeModbusReadings();
143      readGandTc();
144      sendJson();
145      saveDataSD();
146    }
147    delay(1000);
148  }
```

Figura 28: Loop principal del programa

4.4.4 Módulo SD del PLC

El módulo SD integrado en el PLC es una herramienta fundamental para el almacenamiento local de los datos capturados por el sistema de monitorización. Este componente permite guardar los datos en tiempo real en una tarjeta microSD asegurando que ninguno de estos se pierda y proporcionando un respaldo confiable en caso de fallos en la conectividad con el servidor o interrupciones en el envío de datos.

Para implementar el módulo SD, se empleó la librería SD.h, que está incluida en el entorno Arduino [52]. Esta librería proporciona un conjunto de funciones específicas para la lectura, escritura y gestión de archivos en la tarjeta microSD. Ofrece compatibilidad con sistemas de archivos FAT16 y FAT32, asegurando un funcionamiento confiable con tarjetas de diferentes capacidades. Su simplicidad y robustez la convierten en una opción ideal para proyectos que requieren almacenamiento local.

La integración del módulo SD en el sistema se logró mediante las funciones SDSetup(), writeFile(), appendFile() y saveDataSD(), que gestionan la inicialización, escritura y almacenamiento de datos. A continuación, se describen brevemente estas funciones:

- SDSetup(), revisar Figura 29: Inicializa el módulo SD al inicio del sistema y verifica que la tarjeta SD esté correctamente instalada. Si no se detecta la tarjeta u ocurre un error durante la inicialización, se genera un mensaje de error para informar al usuario. Además, se crea un archivo datos.csv y se escribe su encabezado para organizar los datos almacenados. El formato de almacenamiento utilizado fue CSV anotado, debido a su simplicidad y facilidad de uso, permitiendo que el archivo datos.csv pueda introducido directamente en una base de datos influxdb.

```
197  void SDSetup() {
198    Serial.println("Inicializando tarjeta SD...");
199    if (!SD.begin()) {
200      Serial.println("Error al inicializar la tarjeta SD");
201      return;
202    }
203    Serial.println("Tarjeta SD inicializada correctamente.");
204
205    // Escribe el encabezado en el archivo
206    writeFile(SD, "/datos.csv", "#group,false,false,false,false,true,true\n");
207    appendFile(SD, "/datos.csv", "#datatype,string,long,dateTime:RFC3339,double,string,string\n");
208    appendFile(SD, "/datos.csv", "#default,_result,,,,,\n");
209    appendFile(SD, "/datos.csv", ",result,table,_time,_value,_field,_measurement\n");
210  }
```

Figura 29: Función SDSetup()

- writeFile() y appendFile(), puede observarse en la Figura 30: Estas funciones manejan la escritura y anexado de datos en la tarjeta SD. La función writeFile() crea un archivo nuevo y escribe en él, mientras que appendFile() agrega datos al final de un archivo existente. Ambas funciones verifican posibles errores, como la imposibilidad de abrir el archivo, y aseguran que el contenido se almacene correctamente. Ambas funciones fueron extraídas de uno de los códigos que el fabricante del PLC proporciona como ejemplo por lo que no fue necesaria su codificación.

```
455  void writeFile(fs::FS& fs, const char* path, const char* message) {
456    //Serial.printf("Writing file: %s\n", path);
457    File file = fs.open(path, FILE_WRITE);
458    if (!file) {
459      Serial.println("Failed to open file for writing");
460      return;
461    }
462    if (file.print(message)) {
463      //Serial.println("File written");
464    } else {
465      Serial.println("Write failed");
466    }
467    file.close();
468  }
```

```
478  void appendFile(fs::FS& fs, const char* path, const char* message) {
479    //Serial.printf("Appending to file: %s\n", path);
480    File file = fs.open(path, FILE_APPEND);
481    if (!file) {
482      Serial.println("Failed to open file for appending");
483      return;
484    }
485    if (file.print(message)) {
486      //Serial.println("Message appended");
487    } else {
488      Serial.println("Append failed");
489    }
490    file.close();
491  }
```

Figura 30: Funciones writeFile() y appendFile()

- saveDataSD(), se ilustra en la Figura 31: Organiza los datos capturados en un formato específico y los almacena en la tarjeta SD mediante la función appendFile(). Cada dato es guardado con una marca de tiempo, el valor medido y el nombre del parámetro al que corresponde dicha medición. Esta función se invoca para guardar los datos en la tarjeta SD en el bucle principal, después de realizar las lecturas y enviar los datos al servidor.

```
498 void saveDataSD() {
499
500   char SDdata[250];
501   snprintf(SDdata, sizeof(SDdata), ",,0,%s,%.2f,G,PLC\n", timebuffer, G);
502   appendFile(SD, "/datos.csv", SDdata);
503   snprintf(SDdata, sizeof(SDdata), ",,1,%s,%.2f,Idc_MPPT1,PLC\n", timebuffer, Idc_MPPT[0]);
504   appendFile(SD, "/datos.csv", SDdata);
505   snprintf(SDdata, sizeof(SDdata), ",,2,%s,%.2f,Idc_MPPT2,PLC\n", timebuffer, Idc_MPPT[1]);
506   appendFile(SD, "/datos.csv", SDdata);
507   snprintf(SDdata, sizeof(SDdata), ",,3,%s,%.2f,Pac,PLC\n", timebuffer, Pac_cont);
508   appendFile(SD, "/datos.csv", SDdata);
509   snprintf(SDdata, sizeof(SDdata), ",,4,%s,%.2f,Pdc,PLC\n", timebuffer, Pdc_inv);
510   appendFile(SD, "/datos.csv", SDdata);
511   snprintf(SDdata, sizeof(SDdata), ",,5,%s,%.2f,Tc,PLC\n", timebuffer, Tc);
512   appendFile(SD, "/datos.csv", SDdata);
513   snprintf(SDdata, sizeof(SDdata), ",,6,%s,%.2f,Vdc_MPPT1,PLC\n", timebuffer, Vdc_MPPT[0]);
514   appendFile(SD, "/datos.csv", SDdata);
515   snprintf(SDdata, sizeof(SDdata), ",,7,%s,%.2f,Vdc_MPPT2,PLC\n", timebuffer, Vdc_MPPT[1]);
516   appendFile(SD, "/datos.csv", SDdata);
517 }
```

Figura 31: Función saveDataSD()

4.4.5 Envío de los datos del PLC al ordenador a través de MQTT

El envío de datos desde el PLC al ordenador se realiza mediante el protocolo MQTT. En este sistema, el ordenador tiene instalado en localhost el *broker* Mosquitto, que se encarga de gestionar la comunicación. Para que esta conexión funcione, tanto el PLC como el ordenador deben estar conectados a la misma red local (LAN) aunque esta no disponga de acceso a internet. Este protocolo permite que el PLC actúe como un *publisher*, publicando los datos recogidos en *topics* específicos, mientras que el ordenador opera como un *subscriber*, recibiendo y procesando dichos datos. Este mecanismo garantiza la transferencia fiable y eficiente de la información capturada por el PLC hacia el ordenador que procesara y visualizara los datos.

Para implementar la comunicación MQTT en el sistema, se utilizó la librería PubSubClient, una solución popular para gestionar conexiones MQTT en dispositivos Arduino y ESP32 [53]. Esta librería ofrece funciones completas para conectar al *broker* MQTT, publicar datos y suscribirse a *topics*. También es necesaria la librería Wi-Fi para que dote al PLC de una dirección IP haciendo que este sea accesible dentro de la red local y pueda comunicarse con el bróker MQTT:

- PubSubClient: Permite establecer una conexión segura entre el PLC y un *broker* MQTT, como Mosquitto. Incluye funciones para gestionar la conexión, autenticar al cliente, publicar datos en *topics* específicos y manejar reconexiones automáticas en caso de fallos.
- WiFi.h: Es la librería estándar para establecer conexiones Wi-Fi en dispositivos ESP32. Proporciona las herramientas necesarias para que el PLC se conecte a la misma LAN que el ordenador, incluso si esta no tiene acceso a internet. La librería permite configurar el nombre de la red (SSID) y la contraseña, así como verificar el estado de la conexión.

Para la implementación del envío de datos MQTT en el código se utilizan las siguientes funciones:

- wifiSetup(), ver figura 32: Se encarga de conectar el PLC a la red Wi-Fi local. Configura e inicia la conexión utilizando el SSID y la contraseña de la red. En un bucle, verifica continuamente el estado de la conexión hasta que esta se establece o se supera un tiempo máximo de espera (15 segundos). Si no logra conectarse dentro del límite, se genera un mensaje de error y la función finaliza. En caso de éxito, se imprime en el monitor serial la red a la que está conectado el dispositivo y su dirección IP local, por la que otros dispositivos de la misma LAN podrán establecer conexión con él.

```
158  void wifiSetup() {
159    WiFi.mode(WIFI_STA);
160    WiFi.begin(ssid, password);
161    /* Wait for connection */
162    unsigned long TimeoutWifi = millis();
163    while (WiFi.status() != WL_CONNECTED && millis() - TimeoutWifi < 15000) {
164      delay(500);
165      Serial.print(".");
166    }
167    if (WiFi.status() != WL_CONNECTED) {
168      Serial.println("No se pudo conectar al Wi-Fi. Reintentando más tarde...");
169      return;  // Salir de la función wifiSetup y continuar
170    }
171    Serial.println("");
172    Serial.print("Connected to ");
173    Serial.println(ssid);
174    Serial.print("IP address: ");
175    Serial.println(WiFi.localIP());
176  }
```

Figura 32: Función wifiSetup()

- reconnectMQTT(), consultar Figura 33: Se utiliza para establecer o restablecer la conexión con el bróker MQTT en caso de pérdida de la misma. En un bucle controlado por un tiempo límite (15 segundos), el PLC intenta conectarse al bróker usando un identificador único (clientId) y suscribiéndose al topic configurado. Si el intento falla, se genera un mensaje indicando el estado de la conexión y se espera un segundo ante de reintentar. Si después del tiempo límite no se logra la conexión, se imprime un mensaje de error y la función se detiene. Esto asegura que la comunicación MQTT sea lo más robusta posible frente a interrupciones.

```
217  void reconnectMQTT() {
218    unsigned long TimeoutBrokerMQTT = millis();
219    while (!client.connected() && millis() - TimeoutBrokerMQTT < 15000) {
220      Serial.print("Intentando conectar al broker MQTT...");
221      String clientId = "PLCESP32";
222      if (client.connect(clientId.c_str())) {
223        Serial.println("connected");
224        client.subscribe(mqtt_topic);
225      } else {
226        Serial.print("Fallo de conexión, estado: ");
227        Serial.println(client.state());
228        delay(1000);
229      }
230    }
231    if (!client.connected()) {
232      Serial.println("No se pudo conectar al broker MQTT. Reintentando más tarde...");
233      return;  // Salir de la función
234    }
235  }
```

Figura 33: Función reconnectMQTT()

- sendJson(), como se muestra en la Figura 34: Recopila los datos del sistema y los organiza en un documento JSON que será enviado al *broker* MQTT. Primero, se obtiene

la marca de tiempo actual a partir del RTC, que se serializa junto con los datos medidos. El JSON resultante se serializa en una cadena de texto y se intenta publicar en el *topic* Home/DatosPLC. Si el cliente MQTT está conectado, la publicación se realiza y se imprime el mensaje en el monitor serial. En caso contrario, se intenta restablecer la conexión llamando a reconnectMQTT().

```
409  void sendJson() {
410
411    now = rtc.now();
412    snprintf(timebuffer, sizeof(timebuffer), "%04d-%02d-%02dT%02d:%02d:00Z",
413             now.year(), now.month(), now.day(),
414             now.hour(), now.minute());
415
416    StaticJsonDocument<300> jsonDoc;
417    jsonDoc["Timestamp"] = timebuffer;
418    jsonDoc["Vdc_MPPT1"] = Vdc_MPPT[0];
419    jsonDoc["Idc_MPPT1"] = Idc_MPPT[0];
420    jsonDoc["Vdc_MPPT2"] = Vdc_MPPT[1];
421    jsonDoc["Idc_MPPT2"] = Idc_MPPT[1];
422    jsonDoc["Pdc"] = Pdc_inv;
423    jsonDoc["Pac"] = Pac_cont;
424    jsonDoc["G"] = G;
425    jsonDoc["Tc"] = Tc;
426
427    // Serializar el JSON a un string
428    char jsonString[300];
429    serializeJson(jsonDoc, jsonString);
430
431    // Publicar en MQTT si está conectado
432    if (client.connected()) {
433      if (!client.publish(mqtt_topic, jsonString)) {
434        Serial.println("Error: No se pudo publicar el JSON en MQTT.");
435        Serial.print("Estado MQTT: ");
436        Serial.println(client.state());
437      } else {
438        //Serial.println("Mensaje publicado exitosamente en broker MQTT:");
439        Serial.println(jsonString);
440      }
441    } else {
442      Serial.println("Error: No conectado a MQTT.");
443      reconnectMQTT();  // Intenta reconectar
444    }
445  }
```

Figura 34: Función sendJson()

En el bucle principal del programa según se observa en la Figura 35, se verifica si el dispositivo sigue conectado a la red Wi-Fi, si no, intenta restablecer la conexión mediante wifiSetup(). Del mismo modo, verifica la conexión con el *broker* MQTT y llama a reconnectMQTT() si es necesario. Una vez aseguradas ambas conexiones, mantiene activa la comunicación MQTT con client.loop(). Posteriormente, cuando se realice la captura de los datos, mandara ejecutar la función sendJson() para mandar estos al *broker*.

```
126  void loop() {
127    if (WiFi.status() != WL_CONNECTED) {
128      Serial.println("Wi-Fi desconectado. Intentando reconectar...");
129      wifiSetup();  // Reconectar al Wi-Fi
130    }
131    if (!client.connected()) {
132      reconnectMQTT();  // Reconectar al broker MQTT
133    }
134    client.loop();
135    // Verificar si ha pasado el tiempo para una nueva lectura
```

Figura 35: Reconexiones en el loop principal del programa

4.4.6 Recepción y procesamiento de los datos en Node-RED

Una vez que los datos llegan al *broker* Mosquitto mediante el protocolo MQTT, se procede a procesarlos y adaptarlos para su almacenamiento en la base de datos InfluxDB y su visualización. Para ello, se empleó Node-RED, una herramienta visual basada en flujos. Este

entorno simplifica la manipulación de datos gracias a su amplia gama de nodos preconfigurados y su capacidad para integrar servicios y protocolos de manera eficiente [54]. A continuación, la figura 36 describe la solución desarrollada en esta aplicación:

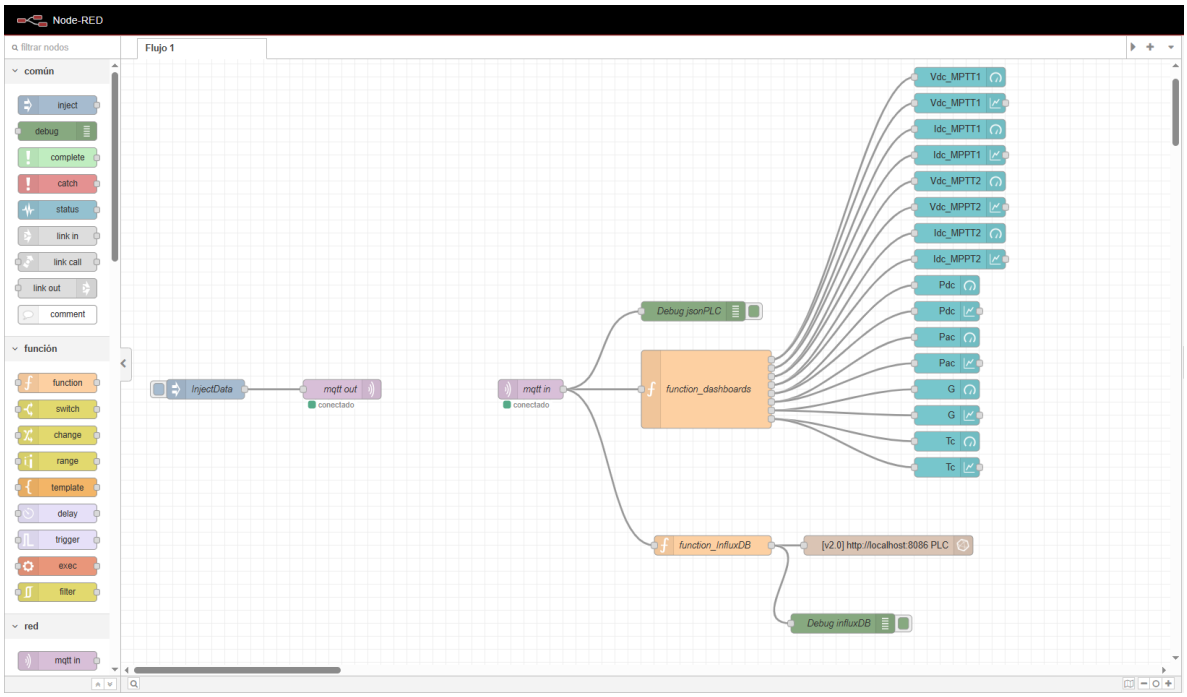

Figura 36: Flujo implementado en Node-RED

Se utilizaron tres librerías principales para implementar el flujo en Node-RED:

- Node-red (nodos básicos): Esta librería es el núcleo de Node-RED e incluye nodos esenciales para construir y gestionar flujos. Estos nodos son fundamentales para realizar tareas como el control del flujo, la manipulación de datos y la integración de sistemas externos.
- Node-red-contrib-influxdb: Es una extensión de Node-RED que proporciona nodos específicos para interactuar con la base de datos InfluxDB. Estos nodos permiten configurar la conexión con la base de datos, especificar el *bucket* de destino y enviar puntos de datos con *tags, fields* y marcas temporales. La librería facilita la integración de datos monitorizados en InfluxDB, asegurando su organización y disponibilidad para análisis y visualización en tiempo real.
- Node-red-dashboard: Esta librería habilita la creación de interfaces gráficas personalizables directamente desde Node-RED, lo que permite construir *dashboards* interactivos para la visualización de datos en tiempo real. Ofrece una amplia gama de *widgets*, como gráficos de línea, barras, medidores y tablas, que pueden configurarse para mostrar métricas clave de manera intuitiva, así como una representación visual clara de los datos monitorizados.

Los nodos clave utilizados en Node-RED se integraron para formar un flujo completo y eficiente:

- Nodo inject: Este nodo fue fundamental durante el desarrollo y las pruebas del flujo, ya que permite generar mensajes de prueba de manera manual. En este proyecto, se utilizó para simular datos provenientes del PLC en las fases iniciales, facilitando la validación de las configuraciones en los nodos mqtt out y influxdb out. Para ello, se configuró el nodo inject para enviar un objeto msg con la misma estructura JSON que el PLC envía y al mismo *topic* en que el PLC publicara los datos, simulando así una publicación real del dispositivo. La configuración de este nodo puede verse en la figura 37:

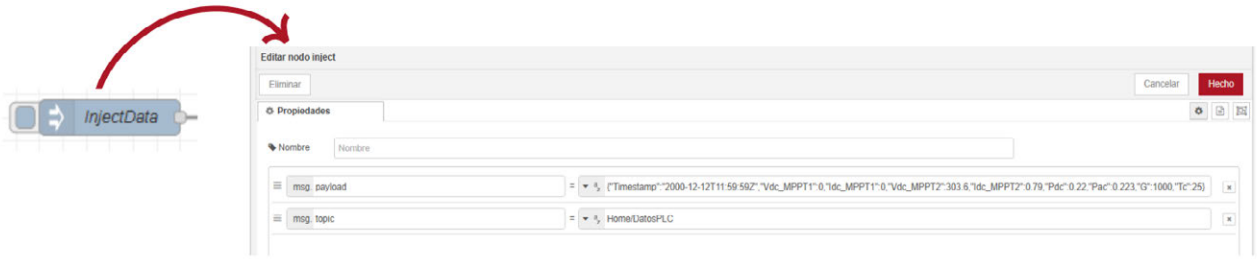

Figura 37: Nodo inject

- Nodos debug: El nodo debug es una herramienta esencial para la depuración y el monitoreo del flujo en tiempo real. Este nodo permite visualizar el contenido de los mensajes en la ventana de depuración de Node-RED, proporcionando información valiosa sobre el formato de los datos y posibles errores. En el contexto del proyecto, el nodo debug fue utilizado para verificar que los datos recibidos desde los nodos mqtt in estaban correctamente formateados y que las transformaciones realizadas en los nodos function producían los resultados esperados antes de almacenarlos en InfluxDB. En la figura 38 se detalla cómo está configurado este nodo:

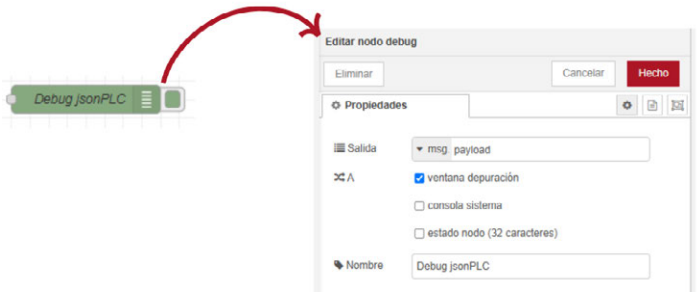

Figura 38: Nodo Debug

- Nodo mqtt in: Este nodo recibe los datos enviados por el PLC al *broker* Mosquitto mediante el protocolo MQTT. Se configura para suscribirse al *topic* #, lo que le permite recibir todos los mensajes publicados en cualquier tema del bróker MQTT. La configuración del nodo especifica el servidor del bróker como localhost:1883, indicando que el bróker MQTT (Mosquitto) se ejecuta en la misma máquina que Node-RED y utiliza el puerto estándar 1883. Además, se habilita la salida en formato JSON y se establece una Calidad de Servicio (QoS) de nivel 2 para garantizar que los datos se

entreguen al menos una vez sin duplicados. La figura 39 ejemplifica la configuración de este nodo:

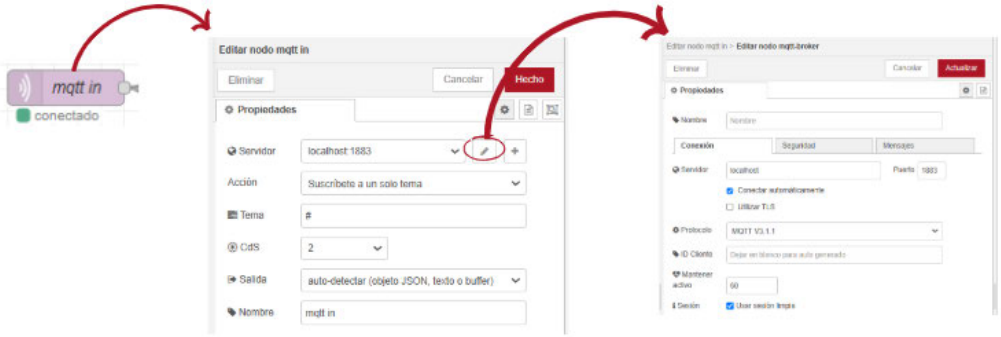

Figura 39: Nodo mqtt in

- Nodo mqtt out: Este nodo se utiliza para enviar mensajes al *broker* MQTT. En el proyecto, aunque su principal uso fue complementario, el nodo mqtt out se configuró para publicar los datos introducidos mediante el nodo inject en el *broker*. Al igual que el nodo mqtt in, se conecta al *broker* ubicado en localhost:1883. Véase la figura 40 para la configuración de este nodo:

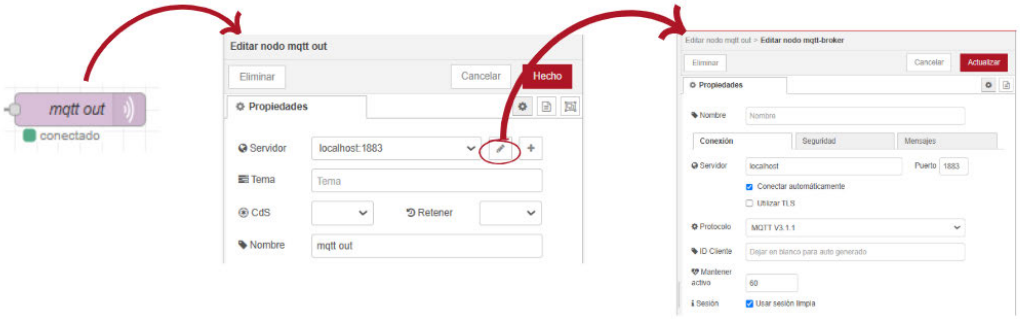

Figura 40: Nodo mqtt out

- Nodo function: El nodo function permite ejecutar código JavaScript personalizado sobre los mensajes que recibe brindando una flexibilidad total para procesar y transformar datos según las necesidades específicas del flujo. Al recibir un mensaje, el nodo function puede acceder y modificar sus propiedades, como msg.payload o msg.topic, y luego devolver uno o más mensajes para continuar con el procesamiento en el flujo. Esta capacidad es especialmente útil cuando se requieren operaciones lógicas, cálculos, manipulación de cadenas o cualquier otra tarea que no se puede realizar con los nodos predefinidos. En concreto, en la solución adoptada se han empleado dos nodos function:

    - Function_dashboard: Este nodo toma el msg.payload entrante, que contiene el JSON con los datos procedentes del PLC para separarlos y devolver una matriz de mensajes individuales, cada uno con un *topic* y *payload* correspondientes a cada parámetro. Esta estructura facilitara la integración con nodos de visualización en el *dashboard*, permitiendo

una representación clara y organizada de cada métrica. El código que ejecuta dicho nodo puede verse reflejado en la figura 41:

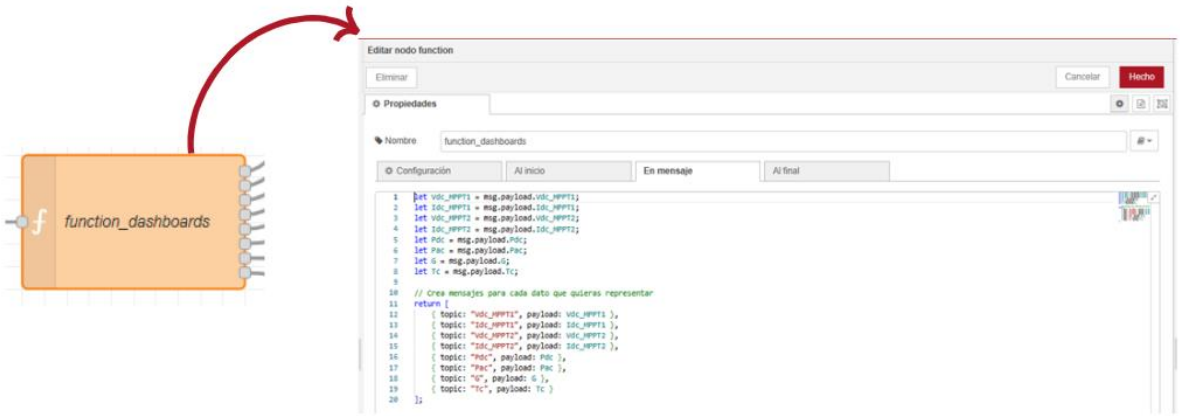

Figura 41: Nodo function_dashboards

- Function_InfluxDB: Este nodo se encarga de preparar los datos para su almacenamiento en InfluxDB. Primero, verifica si msg.payload es una cadena JSON y la convierte en un objeto si es necesario. Luego, extrae y convierte los valores de los parámetros relevantes a números, asignando un valor de 0 en caso de que algún parámetro esté ausente. También extrae la *Timestamp* y valida su existencia. Finalmente, construye un array con un objeto que contiene los campos y la marca de tiempo en el formato esperado por InfluxDB, y lo asigna a msg.payload para su posterior procesamiento. La figura 42 contiene los detalles del código del nodo:

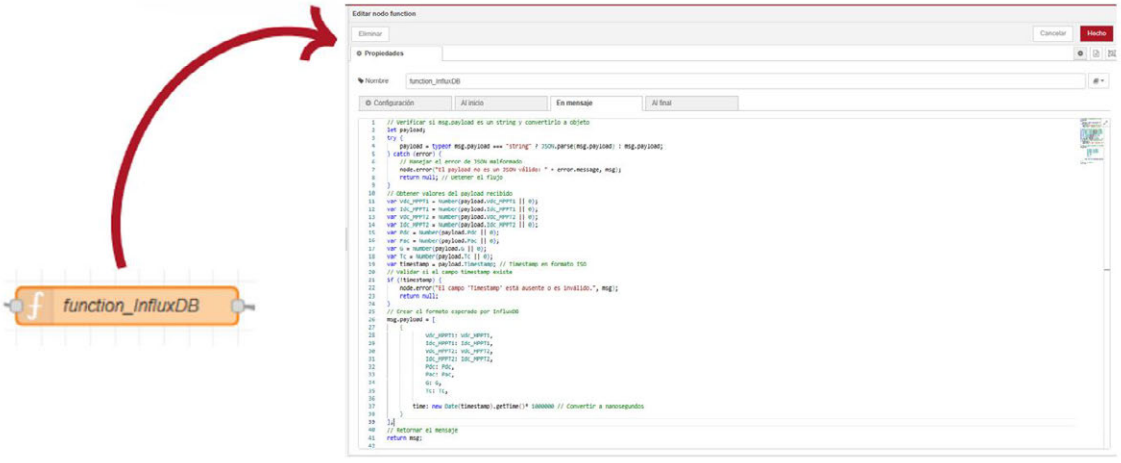

Figura 42: Nodo function_influxDB

- Nodo influxdb out: Se utiliza para escribir datos en una base de datos InfluxDB. En este caso, el nodo recibe un msg.payload de la función anterior que contiene los campos y valores que se desean almacenar junto con la marca de tiempo en nanosegundos.El nodo se configura para escribir datos en una base de datos InfluxDB 2.0 alojada localmente en *localhost* como a través del puerto 8086. Además, se definen otros

parametros relacionados con la configuracion de la base de datos como el *bucket* donde se almacenarán los datos, el *token* de autenticación necesario para acceder a la base de datos, la organización correspondiente y el nombre de la medición donde se registrarán los datos. Para la configuración de este nodo, remítase a la figura 43:

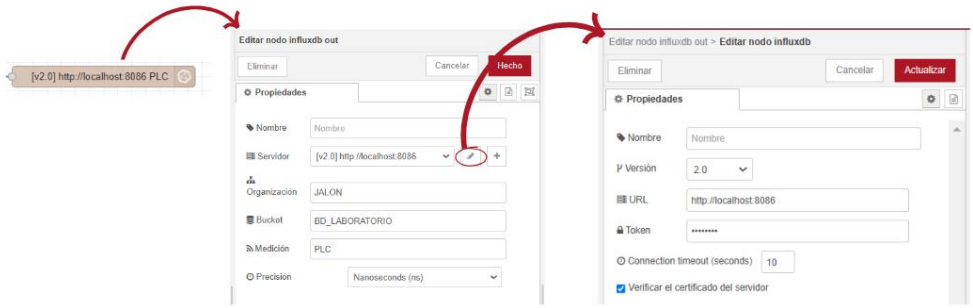

Figura 43: Nodo influxdb out

- Nodos gauge: Estos nodos son una herramienta del paquete node-red-dashboard que permiten visualizar datos en tiempo real de manera gráfica e intuitiva. Para su funcionamiento, estos nodos reciben un mensaje con el valor numérico en msg.payload con la medida a representar. La configuración de este nodo incluye diferentes estilos y tipos de visualización ademas parámetros como el rango mínimo y máximo de valores, la etiqueta, las unidades y los colores de las diferentes secciones del medidor.La configuración correspondiente a este nodo se ilustra en la figura 44:

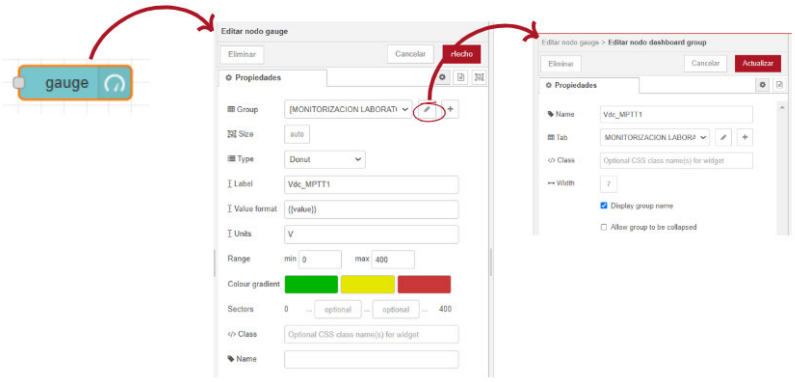

Figura 44: Nodo gauge

- Nodos chart: Estos nodos permiten visualizar datos en tiempo real mediante gráficos de líneas o barras. Su funcionamiento es similar al de los nodos gauge, recibiendo mensajes con valores numéricos en msg.payload y utilizando msg.topic para distinguir entre diferentes series de datos. La configuración del nodo incluye parámetros como el tipo de gráfico, el intervalo de tiempo a mostrar, etiquetas para los ejes y opciones de colores para las distintas series. Además, es posible definir el rango de valores para los ejes y personalizar la apariencia del gráfico según las necesidades del usuario. La configuración empleada en este nodo se visualiza en la figura 45:

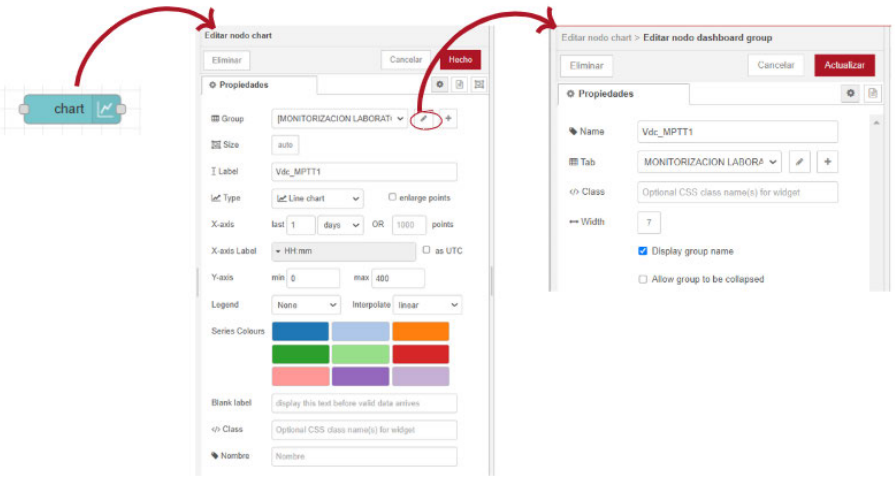

Figura 45: Nodo chart

La combinación de nodos gauge y chart permite crear un panel de control como el de la figura 46 que se alinea con los objetivos de este proyecto. Sin embargo, estos nodos presentan una limitación significativa ya que los datos que muestran son en tiempo real y no se conservan tras un reinicio de Node-RED ya que se almacenan en memoria volátil. Esta falta de persistencia impide mantener un historial de datos, lo que llevó a descartar esta solución como definitiva. Es por este motivo que, para abordar este inconveniente, se emplea una base de datos de series temporales utilizando InfluxDB para el almacenamiento de datos históricos.

Figura 46: Dashboard en Node-RED

4.4.7 Almacenamiento de datos en InfluxDB

Tras la instalación y arranque de InfluxDB, se accede al servicio ingresando en el navegador del ordenador la dirección “localhost:8086”. Esto abre la interfaz de administración de InfluxDB, donde, tras configurar un usuario, se genera un token de acceso que permite que otros servicios como Node-RED y Grafana puedan interactuar con la base de datos de manera segura. Tras esto, se llevó a cabo la creación de un *bucket*, el cual actúa como contenedor donde se almacenarán los datos recopilados por el sistema. Este *bucket* se configuró para que la información permanezca de manera indefinida, garantizando así la disponibilidad de los datos a largo plazo.

La información almacenada en InfluxDB se organiza en forma de series temporales, donde cada dato registrado incluye las siguientes estructuras:

- *Measurements*: Contiene el nombre de la medición, que en este caso siempre será "PLC".
- *Fields*: Pares de clave-valor asociados al tipo de medición, que incluyen tanto el nombre del parámetro medido como su valor correspondiente (por ejemplo, "Voc_MPPT1": 154,4).
- *Timestamps*: Asocian cada dato con una marca de tiempo en UTC, facilitando así el análisis temporal y la realización de consultas históricas.

Toda la información almacenada en la base de datos se puede visualizar desde la opción *Data Explorer* como se puede observar en la figura 47, que facilita la inspección y consulta de las series temporales registradas. Este apartado permite analizar detalles como las marcas de tiempo y los valores de las mediciones. Las consultas se realizan utilizando el lenguaje Flux, que ofrece capacidades avanzadas, como el filtrado por intervalos de tiempo o el tipo de medición. En la siguiente imagen se muestra un ejemplo de datos almacenados y filtrados según el tipo de medida:

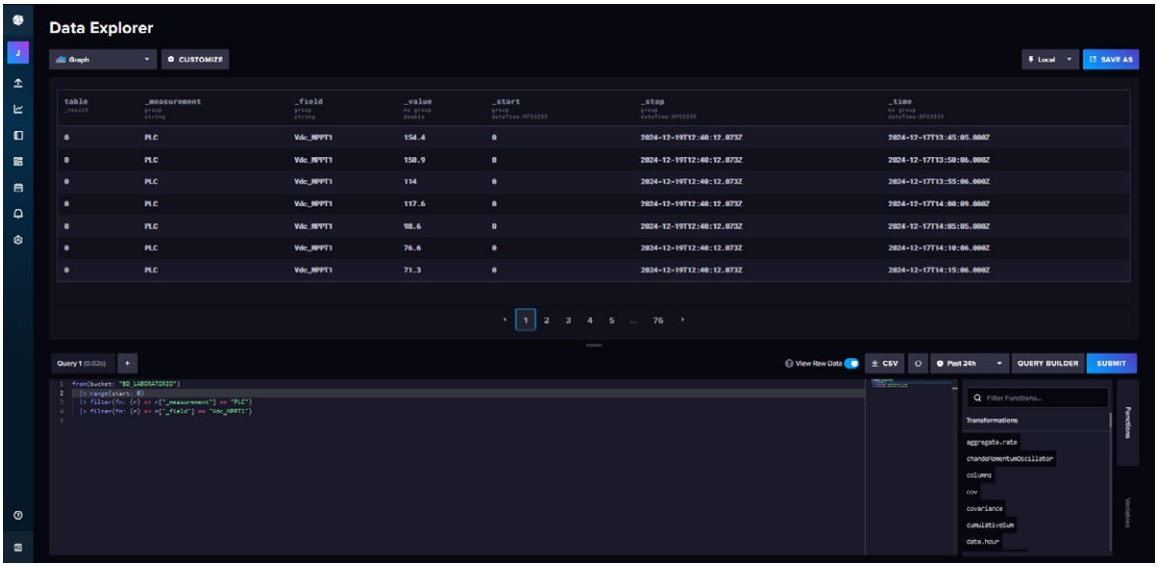

Figura 47: Datos guardados en InfluxDB

Además, InfluxDB permite la creación de paneles de control como los de la figura 48 desde su interfaz gráfica o mediante el lenguaje Flux [55], ofreciendo diversas opciones para personalizar la visualización de datos. La creación de paneles se realiza en la sección *dashboards* [56] donde se selecciona el cubo con las métricas a visualizar, el tipo de gráfico (líneas, barras o mapas de calor entre otros) y se configuran opciones como la agrupación por intervalos de tiempo, filtros específicos o personalización de colores y etiquetas. También es posible ajustar la zona horaria y habilitar la actualización automática del panel en intervalos definidos lo que facilita el monitoreo en tiempo real.

Figura 48: Dashboard en InfluxDB

A diferencia del *dashboard* creado en Node-RED, el de InfluxDB permite trabajar tanto con datos en tiempo real como con datos históricos. Sin embargo, como no es recomendable centralizar las herramientas de almacenamiento de datos y visualización en un mismo sistema ya que cualquier usuario con acceso a esta interfaz podría eliminar el *bucket* haciendo que todo el sistema quedara inservible, se decidió separar las funcionalidades y emplear Grafana como herramienta principal de visualización, manteniendo InfluxDB exclusivamente como sistema de almacenamiento.

4.4.8 Visualización de datos en Grafana

Grafana es la herramienta seleccionada para la visualización de los datos almacenados en InfluxDB, destacando por su versatilidad y su capacidad de integración con dicha base de datos. Para acceder a Grafana, basta con ingresar en el navegador la dirección “localhost:3000”. Una vez dentro, el primer paso es configurar InfluxDB como fuente de datos. Esto se realiza en la sección *Data Sources*, donde se agrega una nueva fuente de datos, se selecciona InfluxDB y se introducen las credenciales necesarias, como la URL del servidor (en este caso, http://localhost:8086), el *bucket* correspondiente y el token de acceso previamente generado en InfluxDB. Al completar esta configuración la cual se ilustra en la figura 49, Grafana obtiene acceso a las métricas almacenadas en InfluxDB, permitiendo utilizarlas para la creación y personalización de paneles de visualización.

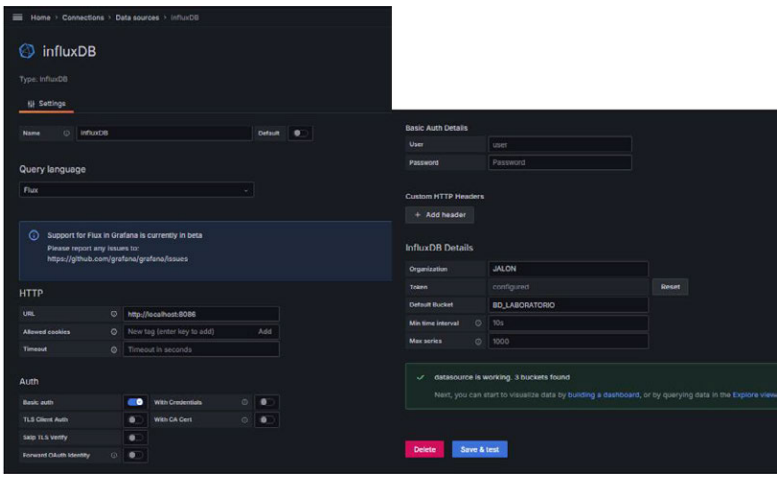

Figura 49: Conexión Grafana con InfluxDB

Grafana permite la creación de paneles y *dashboards* personalizados para visualizar y analizar datos de manera eficiente similarea a los que se pueden observar en la figura 50. Desde la sección *dashboards*, se puede seleccionar la opción "*New Dashboard*" y añadir paneles individuales mediante el botón "*Add new panel*". Estos paneles pueden configurarse de forma intuitiva a través de su interfaz gráfica o empleando el lenguaje Flux. Entre las opciones de visualización disponibles se encuentran gráficos de líneas, gráficos de barras, mapas de calor, indicadores numéricos, tablas y gráficos circulares. Además, permite personalizar los paneles ajustando intervalos de tiempo, colores, estilos y configuraciones visuales para facilitar el análisis de datos. También admite la configuración de alertas automáticas para notificar sobre valores que superen umbrales predefinidos y permite superponer métricas en múltiples ejes en un mismo gráfico para comparar las variables.

Los *dashboards* permiten combinar varios paneles en una única vista para un monitoreo completo. También, existe la opción de mostrar los paneles en modo kiosco y configurarlos para actualizarse automáticamente a intervalos definidos lo cual es una opción ideal para la monitorización en tiempo real. Además, pueden exportarse en formato JSON y compartirse de manera sencilla con otros usuarios,

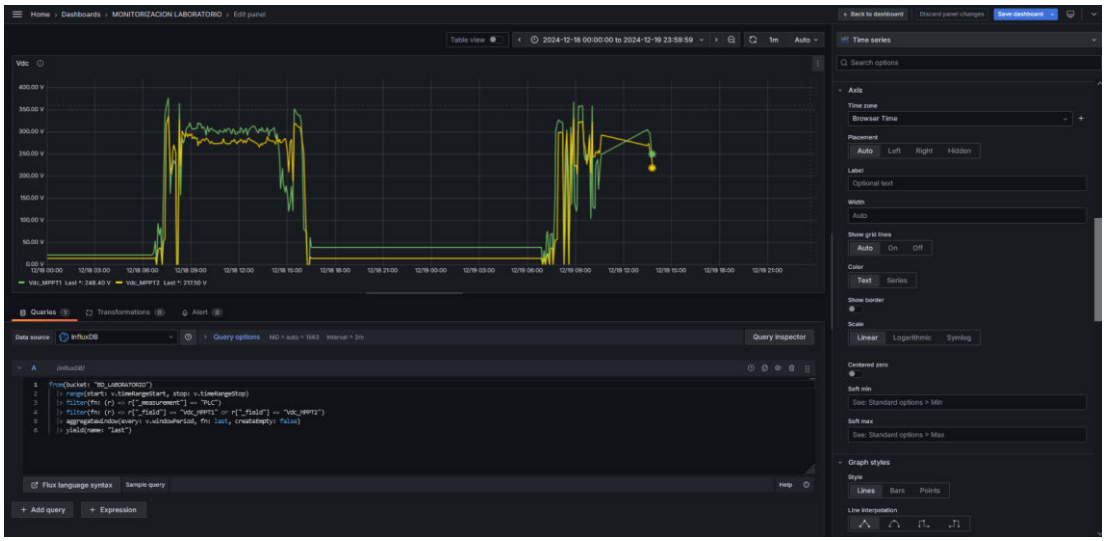

Figura 50: Creación de paneles en Grafana

4.4.9 Consulta remota mediante Tailscale

Como se mencionó anteriormente, Tailscale es una solución VPN que facilita el acceso remoto seguro y sencillo a los servicios implementados en la red local del sistema, como InfluxDB y Grafana. La configuración de Tailscale comienza instalando su cliente en los dispositivos que se desea conectar. Una vez instalado, cada dispositivo inicia sesión en Tailscale utilizando una cuenta vinculada de Google. El sistema asignara automáticamente una dirección IP privada a cada dispositivo permitiendo que sean accesibles y reconocibles entre sí dentro de la red virtual.

En este caso, Tailscale se configuró en el ordenador del laboratorio, donde están alojados Grafana y otros programas, y también en un ordenador personal para habilitar la consulta remota. Tras la configuración, es posible acceder de forma remota a los servicios ingresando en el navegador del ordenador personal la dirección IP privada asignada por Tailscale, seguida del puerto correspondiente. En el ejemplo de la imagen 51, para acceder a Grafana, sería suficiente con introducir en el navegador “http://100.111.143.93:3000”.

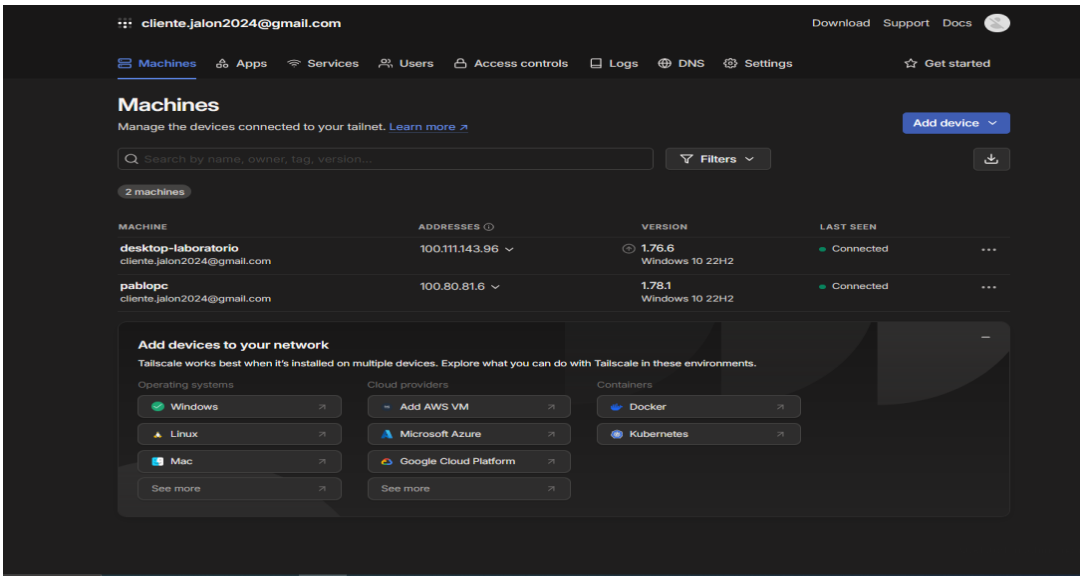

Figura 51: VPN de Tailscale

Además, Tailscale incluye funcionalidades avanzadas, como la gestión de permisos y usuarios, que permiten restringir el acceso a dispositivos o servicios específicos dentro de la red. Esto significa que, si se considera necesario, se puede limitar el acceso a InfluxDB y al resto de programas desde otros dispositivos, mejorando así la seguridad y el control del sistema. Tailscale, por tanto, garantiza un acceso remoto confiable y seguro, simplificando la administración del sistema desde cualquier lugar.

5. Resultados

Este capítulo describe de manera sistemática todas las pruebas a las que fue sometido el sistema para comprobar su correcto funcionamiento junto con los resultados obtenidos de dichas pruebas.

5.1 Extracción de datos desde los dispositivos

Para realizar la extracción de datos desde los dispositivos conectados al sistema, se implementaron dos códigos específicos. El primero, basado en los ejemplos del protocolo Modbus RTU proporcionados por el fabricante del PLC, se utiliza para extraer datos del inversor y contador. El segundo se enfocó en la lectura de las entradas analógicas procedentes de los módulos de referencia para el cálculo de la temperatura de célula e irradiancia.

5.1.1 Lectura de registros del inversor y contador

El propósito principal de esta prueba fue verificar que la conexión y la comunicación entre los dispositivos y el PLC funcionaran correctamente asegurando que los registros leídos correspondieran a los datos específicos que se requerían para el sistema de monitorización. Para ello, el código estaba diseñado para realizar lecturas periódicas de los registros Modbus definidos y enviar los resultados a través del puerto serial. Esta funcionalidad permitió visualizar en tiempo real mediante el monitor serial de Arduino IDE los valores obtenidos, facilitando la detección de posibles errores en la configuración o en la conexión. Durante esta prueba se evaluaron los siguientes aspectos clave:

- Conexión física y comunicación: Confirmar que los dispositivos respondieran a las solicitudes del PLC, verificando la correcta configuración de parámetros como la velocidad de transmisión (*baudrate*), la paridad y las direcciones de los dispositivos esclavos en el bus Modbus.
- Validez de los registros: Asegurar que los registros leídos correspondieran a los parámetros deseados, como tensión, corriente, potencia y otros datos relevantes tanto del inversor como del contador.
- Respuesta de los dispositivos: Validar que los dispositivos devolvieran respuestas coherentes y sin errores de comunicación, indicando un correcto intercambio de información.

El código fue diseñado para ser adaptable y se realizaron versiones específicas para cada uno de los tres modelos de inversores empleados en el proyecto ajustando en cada caso los registros y configuraciones Modbus. Adicionalmente, durante esta prueba se monitorizó el comportamiento del sistema bajo diferentes condiciones para detectar posibles problemas como errores de comunicación desconectando los cables para simular fallos en la transmisión

Los resultados obtenidos confirmaron que los dispositivos pudieron establecer una comunicación efectiva con el PLC y que los registros leídos correspondían correctamente a los parámetros esperados. Sin embargo, durante las pruebas se identificó un problema ocasional

que provocaba que el sistema se bloqueara y dejara de realizar solicitudes a los dispositivos. Este problema se debía a que el PLC quedaba esperando indefinidamente la respuesta de una petición perdida, lo que detenía por completo el funcionamiento del programa. Para resolver esta situación, se implementó la función handleRetry() previamente descrita en este documento. Esta función permite que el PLC reenvíe la solicitud en caso de no recibir respuesta dentro del tiempo establecido. Si la falta de respuesta persiste tras varios intentos, el sistema pasa automáticamente a la siguiente solicitud evitando el bloqueo del programa. Los resultados de esta prueba se ilustran a continuación en la figura 52.

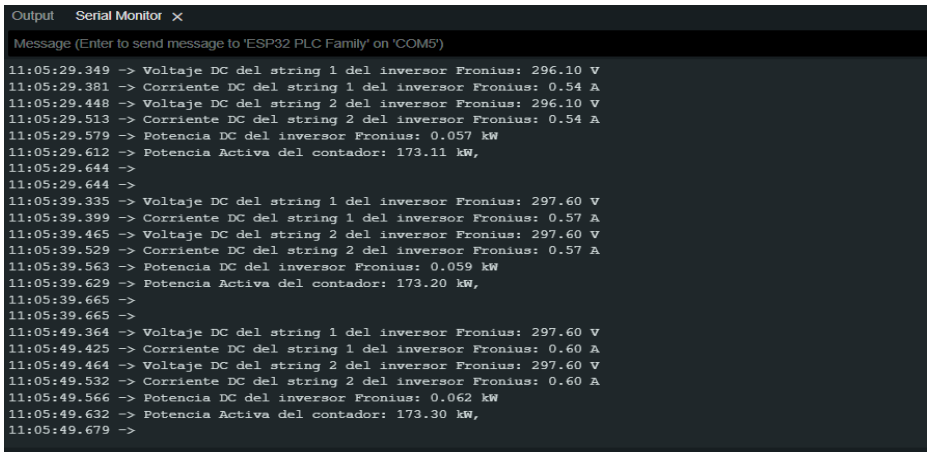

Figura 52: Resultado de la lectura de registros del inversor y contador

5.1.2 Lectura de las entradas analógicas de los módulos de referencia

El propósito principal de esta prueba, al igual que en la anterior, fue verificar que la conexión entre los módulos de referencia y el PLC fuera correcta. Adicionalmente, se buscó comprobar que el PLC fuera capaz de medir con precisión los valores de tensión suministrados a través de sus entradas analógicas. Para ello, inicialmente se introdujeron diferentes valores de tensión al PLC mediante una fuente de alimentación permitiendo asegurar que las señales analógicas eran interpretadas correctamente por el PLC.

Una vez confirmada la precisión en las lecturas, se procedió a desarrollar la función ReadTcandG(), previamente descrita, encargada de calcular la temperatura de célula y la irradiancia a partir de los datos provenientes de los módulos de referencia. Los resultados generados por esta función fueron impresos a través del puerto serial para su verificación como se observa en la figura 53.

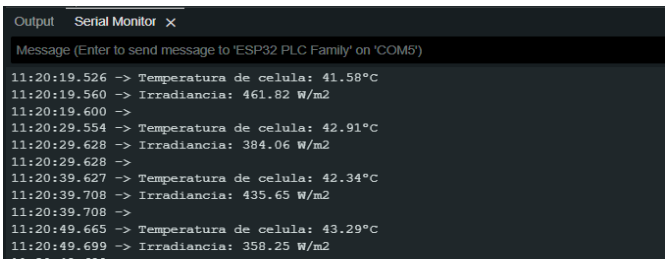

Figura 53: Lectura de las entradas analógicas de los módulos de referencia

5.2 Almacenamiento local en tarjeta SD

El objetivo principal de esta prueba fue verificar que el sistema pudiera almacenar los datos de manera correcta y eficiente en la tarjeta SD, proporcionando un respaldo local en caso de fallos en la transmisión de datos hacia el *broker* MQTT. Además, se evaluó que los datos almacenados tuvieran el formato adecuado para ser introducidos directamente en la base de datos InfluxDB sin necesidad de edición previa.

Dado que InfluxDB permite la importación de datos en formato CSV anotado [38], se decidió utilizar este estándar para el archivo de almacenamiento. Esto facilitó la conectividad entre el sistema de monitorización y la base de datos, asegurando que los datos capturados pudieran ser introducidos de manera inmediata por el usuario en caso de fallos en la transmisión. Este formato incluye filas de anotaciones que describen las propiedades de las columnas, una fila de encabezado que define las etiquetas de las columnas y filas de registros que contienen los datos. La estructura básica del archivo se ilustra en la figura 54 y es la siguiente:

- *#group*: Indica si las columnas forman parte de un grupo de datos, utilizando valores true o false.
- *#datatype*: Define el tipo de dato de cada columna, como *string, long, double* o *dateTime:RFC3339*.
- *#default*: Permite especificar valores predeterminados para las columnas si no se proporcionan datos.
- Encabezado de datos: Contiene las etiquetas de las columnas, como *_time* (marca de tiempo), *_value* (valor medido), *_field* (nombre del parámetro) y *_measurement* (nombre de la medición).
- Filas de datos: Cada fila representa una entrada de datos con su respectiva marca de tiempo y valores medidos.

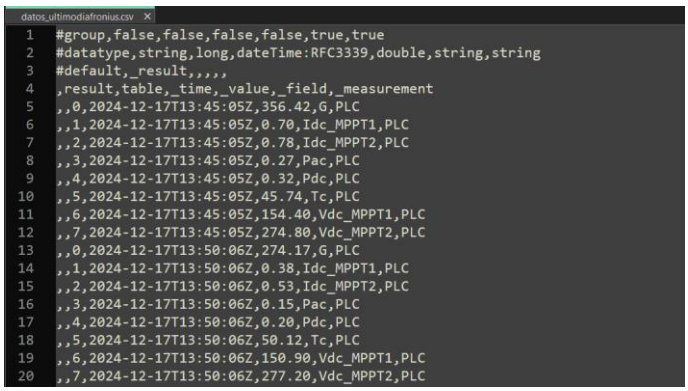

Figura 54: Ejemplo del archivo "datos.txt"

Durante las pruebas, se verificaron aspectos clave como la correcta inicialización del módulo SD, la capacidad del sistema para escribir datos de manera continua y la validez del formato de los registros almacenados. Para garantizar la funcionalidad completa, también se realizaron

pruebas manuales donde los datos almacenados en la tarjeta SD se importaron directamente a la base de datos InfluxDB. Los resultados fueron exitosos, confirmando que el archivo generado era completamente compatible y que no presentaba problemas de formato ni errores al ser cargado. En la siguiente imagen, pueden verse dos medidas guardadas en la tarjeta SD c realizadas con 5 minutos de diferencia con el formato mencionado.

5.3 Publicación de datos en el bróker MQTT

En esta prueba, se publicaron datos desde el PLC hacia el bróker MQTT con el objetivo de verificar que fueran recibidos de manera correcta. Utilizando la librería PubSubClient, se configuró el PLC para enviar los datos en formato JSON al *topic* Home/DatosPLC gestionado por el *broker* Mosquitto.

El formato JSON fue elegido por ser ligero, basado en texto y ampliamente compatible con múltiples lenguajes de programación. Su estructura clara y organizada lo convierte en una opción ideal para la transmisión de datos en entornos IoT. Cada mensaje JSON enviado por el PLC sigue una estructura de pares clave-valor, donde las claves son cadenas de texto y los valores pueden ser numéricos o también cadenas de texto. Dentro de esta estructura, el campo *timestamp* almacena la marca de tiempo en formato ISO 8601, permitiendo organizar los datos cronológicamente. Los demás campos representan valores medidos en el sistema, como temperatura, irradiancia y parámetros eléctricos del inversor y contador. Puede verse un ejemplo en la figura 55.

Para supervisar los datos publicados y confirmar que eran correctamente recibidos por el bróker, se utilizó el puerto serial de Arduino para verificar qué datos se enviaban desde el PLC y el programa MQTT Explorer para visualizar en tiempo real los mensajes publicados en los *topics*. Esto permitió detectar posibles errores en la transmisión y validar la integridad de los datos enviados.

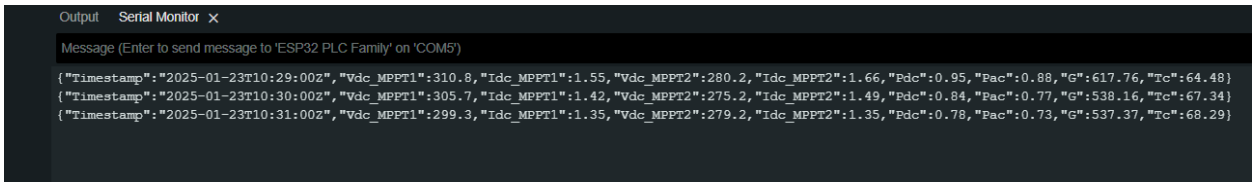

Figura 55: Datos publicados en Mosquitto

Los resultados fueron positivos, ya que el bróker, monitoreado a través de MQTT Explorer, recibió de forma consistente y sin interrupciones todos los datos enviados por el PLC. Esto confirmó la correcta configuración del sistema y validó la funcionalidad básica de la transmisión de datos mediante el protocolo MQTT.

5.4 Procesamiento de datos en Node-RED

En esta prueba se evaluó el correcto funcionamiento del flujo desarrollado en Node-RED para procesar y almacenar los datos, simulando la recepción de información sin necesidad de depender de los datos enviados por el PLC.

Para llevar a cabo esta simulación, se utilizaron los nodos “inject” y “mqtt in” vistos anteriormente para enviar datos en formato JSON que replicaran los mensajes normalmente transmitidos por el PLC al bróker y mediante los nodos “debug” colocados en puntos clave, se mostraron por pantalla la transformación y el manejo de los datos.

Además, se creó un *bucket* de prueba en InfluxDB para almacenar estos datos simulados en Node-RED. Esta configuración permitió no solo verificar que los datos fueran correctamente procesados, sino también confirmar su almacenamiento preciso y estructurado en la base de datos. Tras ejecutar el flujo completo, se consultó directamente en InfluxDB para verificar que los datos se habían registrado con las marcas de tiempo y los valores adecuados, asegurando así el correcto funcionamiento del sistema de almacenamiento.

Los resultados fueron satisfactorios. Los mensajes capturados en los nodos “debug” mostraron que el flujo procesó los datos simulados sin errores, y cada transformación realizada fue acorde a lo esperado tal y como se ve en la figura 56. Asimismo, los datos almacenados en el *bucket* de prueba de InfluxDB reflejaron con precisión los valores enviados desde Node-RED, confirmando que el proceso de almacenamiento fue exitoso y sin pérdida de información.

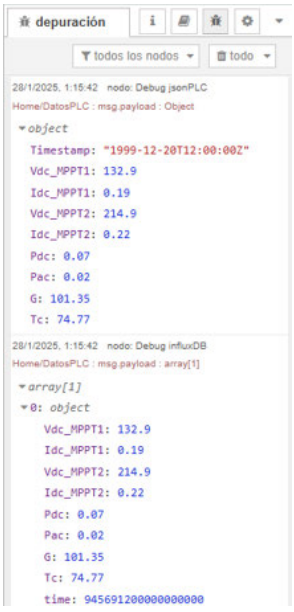

Figura 56: Datos procesados en Node-RED.

5.5 Validación final del sistema

Para la validación final, el sistema se dejó funcionando durante un periodo de tiempo prolongado con el objetivo de observar su comportamiento y confirmar que todas las partes trabajaban correctamente de manera conjunta. Durante este tiempo, no se realizaron modificaciones en ninguno de los programas, lo que permitió determinar si el flujo de información se mantenía estable y sin interrupciones.

Los resultados obtenidos demostraron que el sistema era capaz de operar de forma continua y fiable, cumpliendo con los objetivos planteados en el proyecto y validando su funcionalidad en condiciones reales de operación. Para analizar el desempeño del sistema y facilitar la interpretación de los datos obtenidos, se implementó un *dashboard* en Grafana que proporciona una representación gráfica de las mediciones en tiempo real y su evolución histórica.

La solución final en Grafana consta de dos secciones principales. La primera es la visualización de las últimas mediciones, donde se utilizaron indicadores de tipo *gauge* para mostrar los valores actuales de todas las variables como se puede observar en la figura 57. Cada indicador se actualiza en tiempo real con los datos provenientes de InfluxDB, asegurando que el usuario pueda visualizar instantáneamente el estado del sistema.

Figura 57: Gráficos de las últimas medidas en Grafana

La segunda sección corresponde al historial de datos, donde se implementaron gráficos de líneas para representar la evolución de las variables a lo largo del tiempo. Estos gráficos permiten analizar patrones de comportamiento, detectar anomalías y evaluar la estabilidad del sistema. Se incorporaron múltiples métricas en un mismo gráfico para facilitar la comparación entre las corrientes y tensiones de los diferentes strings, permitiendo evaluar su comportamiento y detectar posibles discrepancias en la generación de energía. Además, el usuario puede seleccionar la hora y fecha de visualización, lo que facilita el análisis de períodos específicos y la comparación de datos en diferentes intervalos de tiempo. Por ejemplo, en la Figura 58 se puede observar el grafico historico de la irradiancia leida durante el dia 30/01/2025.

Figura 58: Ejemplo de grafico histórico de la irradiancia

5.6 Incidencias detectadas y correcciones

Durante el funcionamiento continuo del sistema, se identificaron algunos problemas que podrían afectar la precisión y estabilidad de los datos en periodos prolongados. A continuación, se describen en detalle las incidencias detectadas y las posibles soluciones para mitigar su impacto:

- Valores erróneos en los registros del inversor Fronius Primo y contador

El inversor Fronius Primo presentó varias incidencias durante las pruebas. El primer problema detectado fue una discrepancia en las direcciones de los registros Modbus proporcionadas en el *Map Register* del fabricante. Se descubrió que las direcciones indicadas requerían una corrección, ya que se debía restar 1 a cada valor especificado para obtener la dirección real de los registros. Por ejemplo, para leer la corriente que le llega al MPPT1, el *Map Register* indica la dirección 40283, sin embargo, la dirección correcta es 40282. Este error fue identificado y corregido en el código poniendo el valor de dirección correspondiente.

Otro fallo relacionado con este inversor y con el contador ocurre en los momentos del día con baja irradiancia solar, cuando los dispositivos están en el proceso de encendido o apagado. Durante estos periodos, los valores leídos de los registros pueden ser inconsistentes y tomar el valor 65535 (0xFFFF en hexadecimal), lo que genera datos erróneos en el sistema como se puede observar en la figura 59. Para solucionar esto, se implementó una corrección en el código para que, cuando los valores leídos de los registros del inversor sean 0xFFFF, se reemplacen por 0, evitando así la representación de valores incorrectos en el monitoreo gráfico y asegurando que las mediciones reflejen datos realistas.

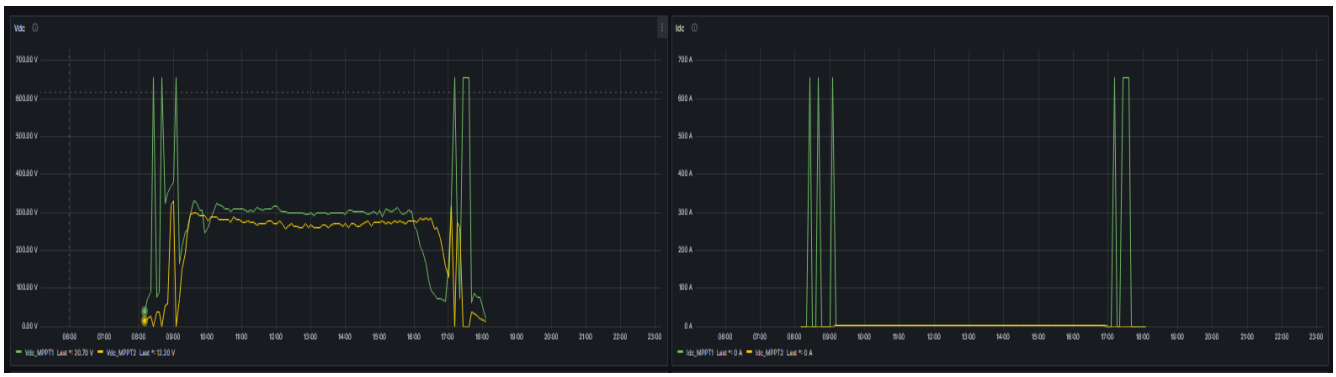

Figura 59: Valores erróneos de los registros del inversor Fronius Primo

- Desviaciones en la hora del RTC

Otro problema detectado fue la desviación progresiva en la hora del RTC cuando el sistema opera sin conexión a Internet. En este estado, el PLC no puede sincronizar su reloj con un servidor NTP, lo que provoca una acumulación de errores en la medición del tiempo. Según el fabricante del PLC, esta desviación se estima entre 5 y 7 segundos por día. Sin embargo, en las pruebas realizadas en el laboratorio, se observó una desviación de aproximadamente 7 minutos en un mes, lo que equivale a 14 segundos de desviación diaria. El origen de este problema puede deberse a la deriva natural del oscilador del RTC, influenciada por factores como la variación de temperatura a la que está sometido el PLC y su hardware interno.

El RTC es fundamental en este sistema, ya que las marcas de tiempo asociadas a cada medición son esenciales para el análisis de datos en InfluxDB y su representación en Grafana. Una desviación en la hora puede generar inconsistencias en las gráficas y dificultar la correlación de eventos en el sistema.

La única forma de mitigar este problema es proporcionar una conexión a Internet al PLC, permitiendo que cada 5 minutos, cuando se realicen las mediciones, el RTC también se actualice mediante un servidor NTP. Esto garantizará la precisión de las marcas de tiempo y evitará desviaciones acumulativas por lo que, aunque el sistema pueda funcionar sin conexión a internet, se recomienda encarecidamente que sí que tenga esta capacidad.

- Valores erróneos en la temperatura de célula e irradiancia por la noche

El último problema identificado está relacionado con los cálculos de la temperatura de célula y la irradiancia durante la noche. En ausencia de radiación solar, los valores de V_{oc} e I_{sc} son prácticamente 0, lo que provoca que la T_c calculada tome valores extremadamente altos (alrededor de 350°C). Este error también afecta al cálculo de la G ya que esta depende de la Tc y de la Isc sin embargo los valores de esta se mantienen dentro de un rango más razonable (en torno a 100 W/m²).

5.7 Análisis de resultados

Una vez que se aseguró de que el sistema funcionaba correctamente, se dejó funcionando en la instalación del laboratorio durante un mes consecutivo (desde el 24 de enero de 2025 hasta el 24 de febrero de 2025). La instalación FV a la que se conectó el sistema se compone de 35 módulos ESPMC050 de 50 Wp de la marca ATERSA, distribuidos en dos ramas de 18 y 17 módulos respectivamente. Esto da lugar a una instalación con una potencia total de 1750 Wp. El sistema se conectó al inversor Fronius Primo del que ya se habló en apartados anteriores. Adicionalmente, se añadieron dos módulos idénticos como módulos de referencia, que fueron empleados para medir los parámetros de tensión en circuito abierto (V_{oc}) y corriente de cortocircuito (I_{sc}).

Durante este periodo, el sistema recopiló y almacenó datos en una tarjeta SD, ocupando un total de 3 MB de espacio. Este bajo consumo de almacenamiento permite estimar que, con una tarjeta SD de 8 GB, el sistema podría seguir almacenando datos durante un periodo extremadamente largo. Las figuras numeradas del 60 al 65 presentan los gráficos correspondientes a los resultados de todas las mediciones realizadas durante el mes completo de operación.

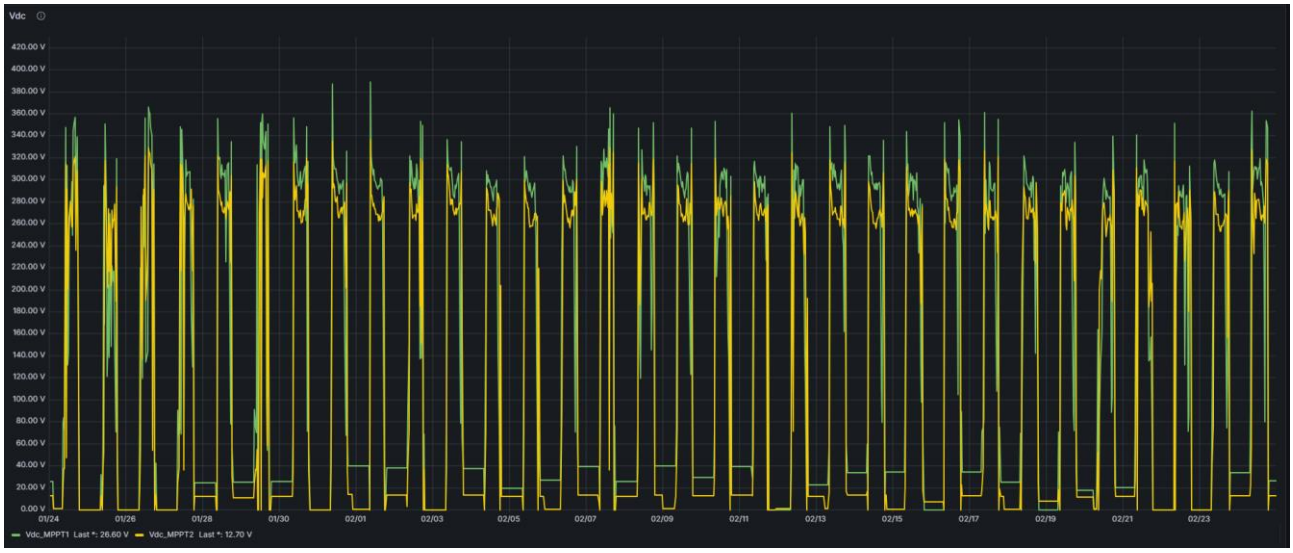

Figura 60: Resultados Vdc_MPPT1 y Vdc_MPPT2 tras un mes de funcionamiento

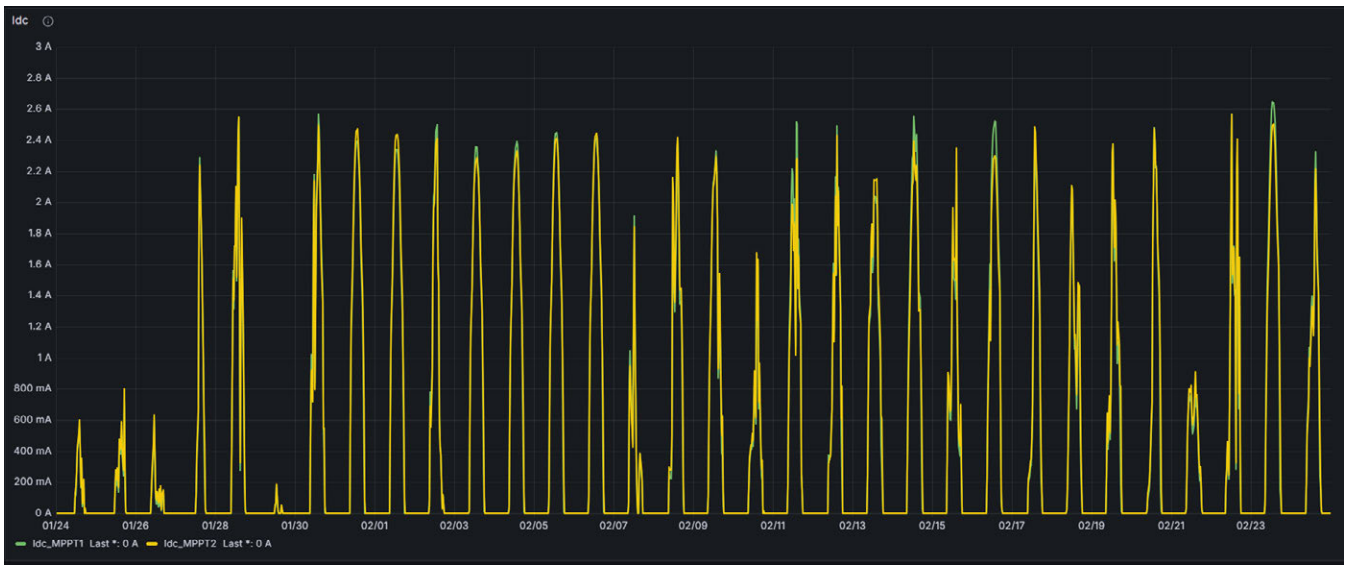

Figura 61: Resultados Idc_MPPT1 e Idc_MPPT2 tras un mes de funcionamiento

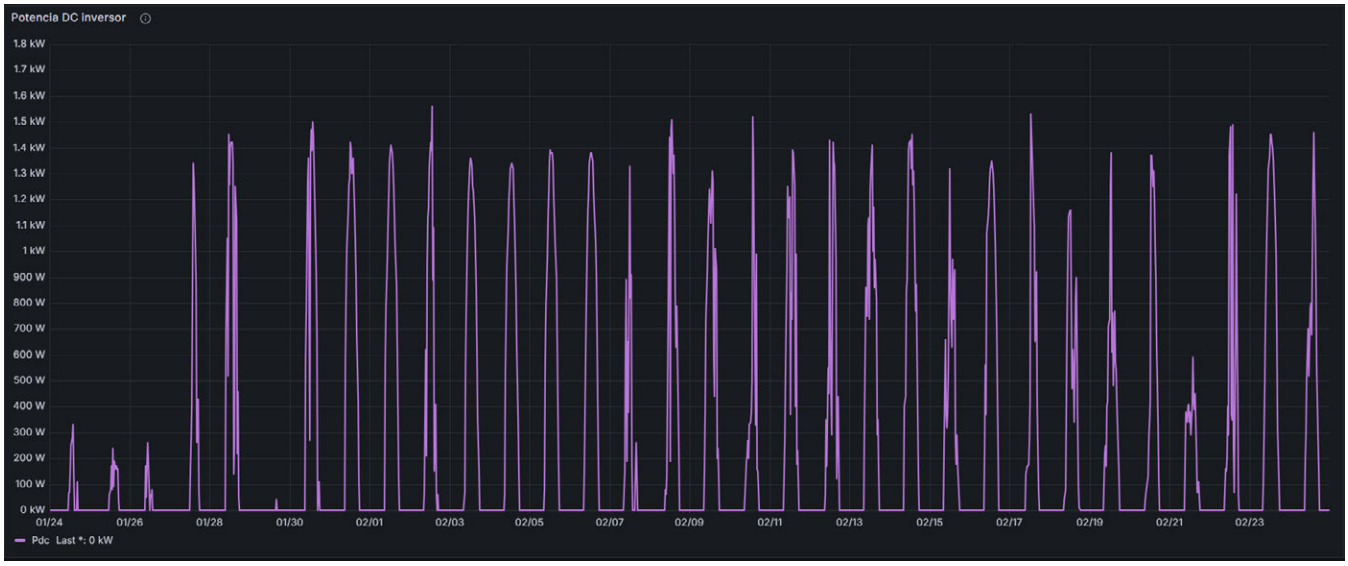

Figura 62: Resultados Pdc tras un mes de funcionamiento

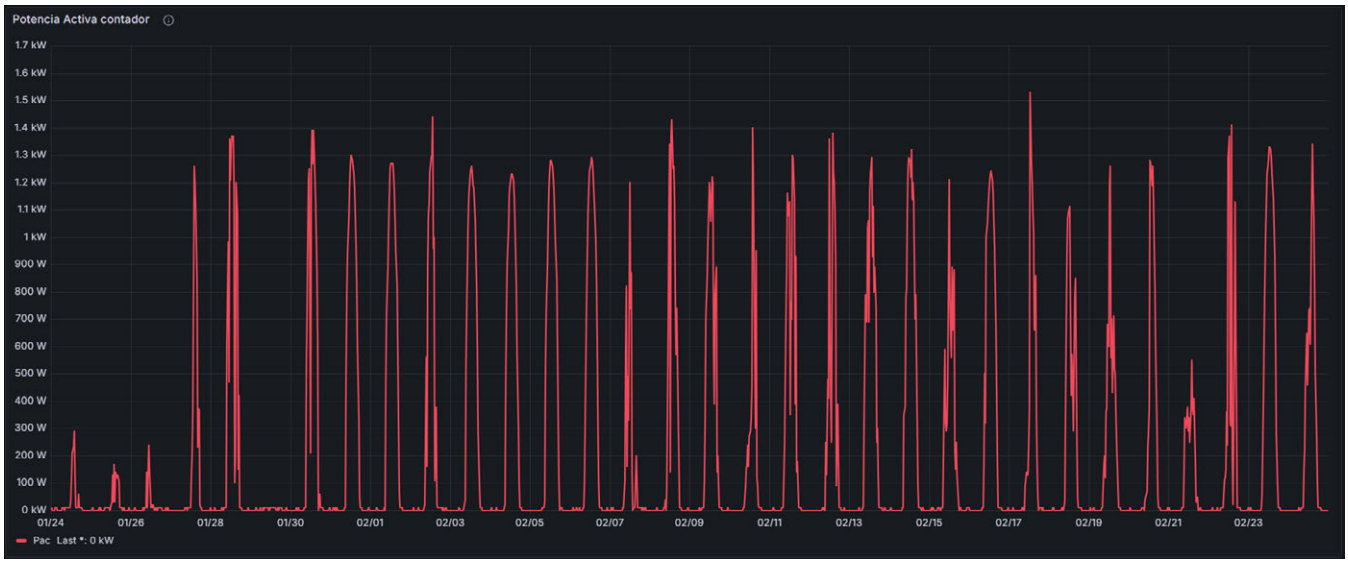

Figura 63: Resultados Pac tras un mes de funcionamiento

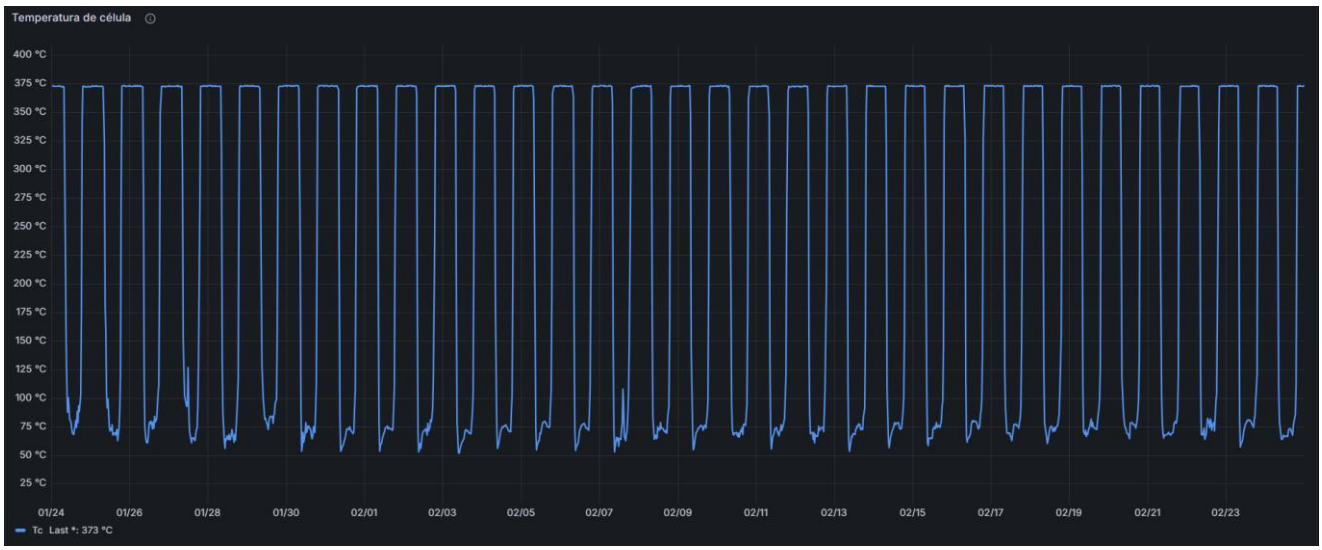

Figura 64: Resultados Tc tras un mes de funcionamiento

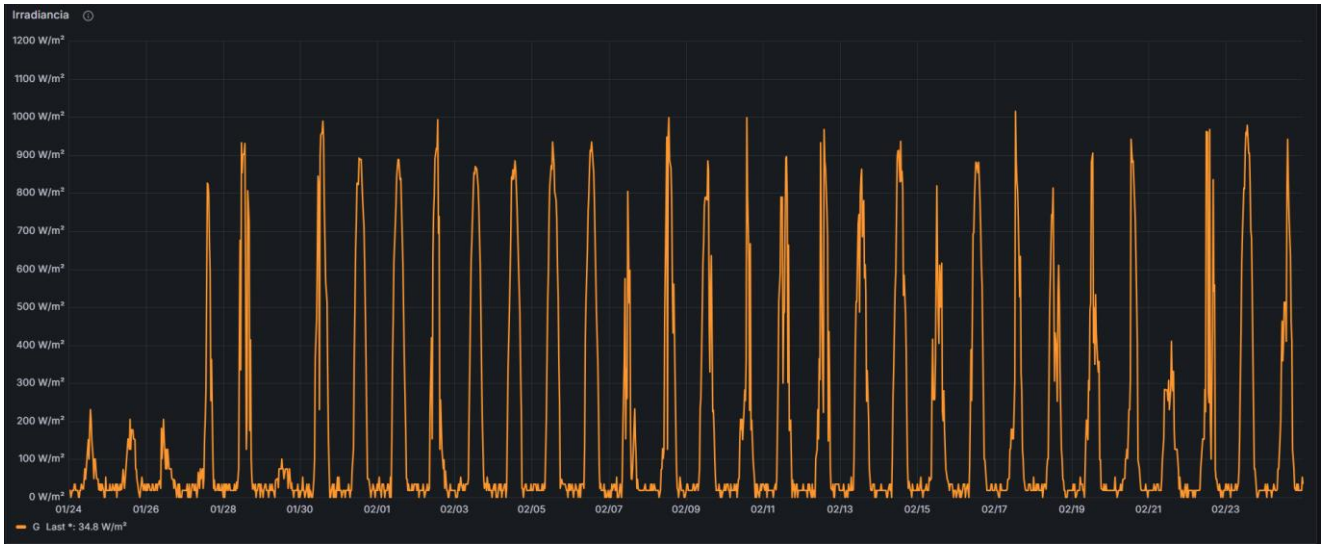

Figura 65: Resultados G tras un mes de funcionamiento

Entre los valores más relevantes registrados se destacan:

- Irradiancia (G): Registró un pico de 1260,44 W/m² el 12/02/2025 a las 13:00:00, que es un día con nubes rápidas.
- Tensión en MPPT1 y MPPT2 (Vdc_MPPT1 y Vdc_MPPT2): La Vdc_MPPT1 alcanzó un máximo de 389,3 V el 30/01/2025 a las 09:00:00 y la Vdc_MPPT2 Llegó a 340 V el 09/02/2025 a las 08:45:00.
- Corriente en MPPT1 y MPPT2 (Idc_MPPT1 e Idc_MPPT2): Se registró un pico de 3,26A y 3,03A respectivamente el 12/02/2025 a las 13:00:00, que, como es de esperar, coincide con el punto de máxima irradiancia.
- Potencia de corriente continua (P_{DC}): Alcanzó un máximo de 1,92 kW el 12/02/2025 a las 13:00:00.
- Potencia de corriente alterna (P_{AC}): Presentó un máximo de 1,85 kW el 12/02/2025 a las 13:00:00.

- Temperatura de célula (T_c): El valor mínimo registrado fue 45,82°C el 30/01/2025 a las 10:25:00. Debido a un error de medida, los valores obtenidos presentan un error aproximado de 15°C a 20°C por encima de la temperatura real.

La irradiancia y la corriente en los MPPTs presentan una fuerte correlación, ya que los picos de corriente coinciden con los momentos de máxima irradiancia, lo que indica un mayor flujo de energía hacia el inversor. De manera similar, la potencia generada, tanto en corriente continua como en alterna, sigue esta misma tendencia, alcanzando sus valores más altos en los momentos de máxima irradiancia. En cuanto a la tensión en los MPPTs, se observa que alcanza sus valores máximos cuando el inversor inicia o detiene la generación de energía, aproximadamente a las 08:30AM y las 19:00PM. Durante estos instantes, se producen picos de tensión mientras la corriente comienza a aumentar o disminuir.

La dependencia de prácticamente todas las medidas con la irradiancia es clara. Además, esto también puede observarse en las gráficas obtenidas en Grafana, donde la mayoría de las curvas siguen un comportamiento similar al de la irradiancia. En la imagen 66 mostrada a continuación, se pueden comparar las variaciones de irradiancia en tres condiciones climáticas distintas: despejado (16 de febrero), nublado (10 de febrero) y lluvioso (24 de enero). Se observa que, en un día despejado, la irradiancia sigue un patrón suave y continuo, con una forma de campana de Gauss y alcanzando valores máximos cercanos a 900 W/m² en torno al mediodía, lo que indica una exposición solar directa sin obstrucciones. En contraste, en un día nublado, la irradiancia presenta fluctuaciones significativas debido al paso intermitente de nubes, lo que provoca variaciones bruscas y picos irregulares a lo largo del día. Entre esos picos es importante mencionar los producidos en torno a las 14:00 horas ya que superan en tamaño incluso a la curva del día soleado. Esto puede ser debido a que, en ocasiones, las propias nubes crean un efecto lupa potenciando la incidencia de los rayos solares sobre los módulos FV. Finalmente, en un día lluvioso, la irradiancia es considerablemente más baja y estable, sin variaciones demasiado abruptas y con un máximo significativamente menor, reflejando la limitada disponibilidad de radiación solar causada por las nubes y la lluvia. Estos resultados muestran cómo las condiciones meteorológicas afectan directamente a la captación de energía solar y, por ende, al rendimiento del sistema FV.

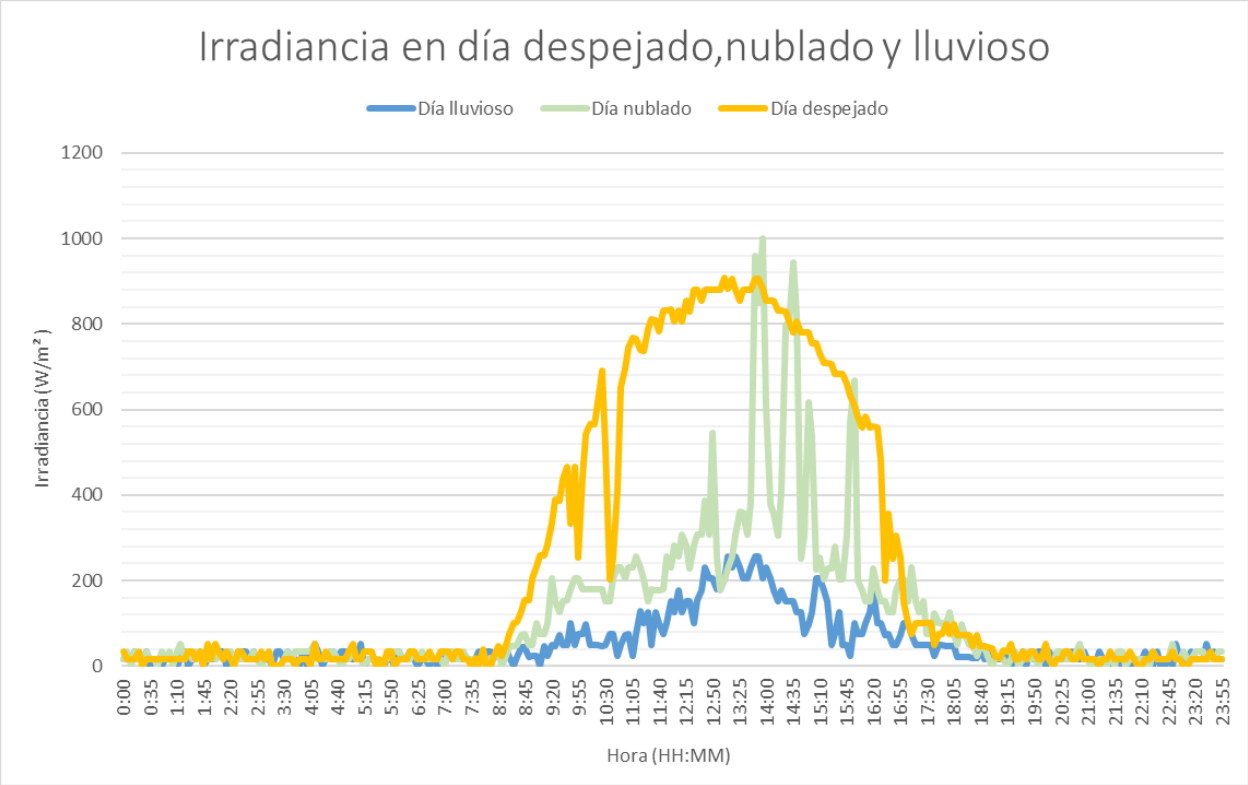

Figura 66: Irradiancia en día despejado, nublado y lluvioso

Mediante la curva de irradiancia, es posible calcular la irradiancia media y la irradiación total diaria. Para el cálculo de la irradiancia media, la cual se mide en W, se empleará la fórmula (6):

$$G_{media,diaria} = \frac{\sum_{i=1}^{n} G_i}{n} \qquad W/m^2 \qquad (6)$$

Donde:

- G_i: Irradiancia instantánea en el instante i.
- n: Número total de mediciones. Como durante la noche no tiene sentido contemplar dichos valores ya que son 0 o muy próximos a 0, el número de mediciones que se contemplarán serán únicamente las tomadas entre las 8:30AM y las 19:00PM que es cuando comienza o se detiene la generación de energía. Esto hace que n sea constante y de valor 126 (12 muestras a la hora x 10,5 horas = 126).

Para el cálculo de la irradiación total diaria (la cual se mide en Wh/m²), será necesario integrar la curva de irradiancia a lo largo del tiempo durante el día. Sin embargo, como los valores de irradiancia que lee el sistema son discretos y a intervalos regulares (cada 5 minutos) podemos aproximar la integral usando la suma de Riemann [57], empleando la expresión (7):

$$H_{total,diaria} \simeq \sum_{i=1}^{n} G[x_i] * \Delta x \qquad Wh/m^2 \qquad (7)$$

Donde:

- $G[x_i]$: Es el valor de la irradiancia en el intervalo i.
- Δx: Es la duración de cada intervalo (5 minutos = 5/60 horas).
- n: es el número total de intervalos. Al igual que con la formula anterior, el número de mediciones que se contemplarán serán únicamente las tomadas entre las 8:30 y las 19:00.

Empleando estas dos fórmulas y sumando todas las irradiaciones totales de cada día para obtener la irradiación total mensual se obtienen los resultados que se reflejan en la tabla 2:

Tabla 2: Irradiación total y media diaria

FECHA	Irradición total diaria (Wh/m²)	Irradiancia media diaria (W)
24/01/2025	1050,70	100,07
25/01/2025	1296,32	123,46
26/01/2025	1027,35	97,84
27/01/2025	2794,23	266,12
28/01/2025	4488,39	427,47
29/01/2025	662,00	63,05
30/01/2025	5120,27	487,65
31/01/2025	5855,23	557,64
01/02/2025	5751,14	547,73
02/02/2025	4151,99	395,43
03/02/2025	5704,76	543,31
04/02/2025	5705,01	543,33
05/02/2025	5904,80	562,36
06/02/2025	5995,09	570,96
07/02/2025	2149,25	204,69
08/02/2025	4704,51	448,05
09/02/2025	4995,08	475,72
10/02/2025	2563,11	244,11
11/02/2025	5000,96	476,28
12/02/2025	4252,73	405,02
13/02/2025	5047,88	480,75
14/02/2025	5750,86	547,70
15/02/2025	3684,66	350,92
16/02/2025	5558,24	529,36
17/02/2025	3857,14	367,35
18/02/2025	4021,01	382,95
19/02/2025	4196,90	399,71
20/02/2025	4112,28	391,65
21/02/2025	2320,74	221,02
22/02/2025	4271,66	406,82
23/02/2025	6402,79	609,79
24/02/2025	4205,73	400,55
	Irradiación total mensual (kWh/m²)	132,60

La irradiación total diaria varía significativamente, desde un mínimo de 662,00 Wh/m² el 29/01/2025 hasta un máximo de 6402,79 Wh/m² el 23/02/2025, con una irradiancia media diaria que oscila entre 63,05 W y 609,79 W en los mismos días. Se observa una tendencia de mayor irradiancia en la segunda mitad del período, especialmente a partir del 30/01/2025, con varios días superando los 5000 Wh/m² debido a que la duración del día va en aumento. En general, los valores de irradiancia obtenidos son bastante aceptables considerando el periodo de medición, lo que se refleja en una irradiación total mensual de 132,6 kWh/m².

De forma análoga a como se calculó la irradiancia media y total diaria, es posible determinar tanto la energía total diaria como la potencia media diaria que llega al inversor, así como la energía total diaria y la potencia media diaria que el inversor inyecta a la red. Esto se logra utilizando las ecuaciones (5) y (6), reemplazando la irradiancia instantánea por la potencia

instantánea en corriente continua o alterna según corresponda. Los resultados de estas operaciones se reflejan la tabla 3:

Tabla 3: Energía total y media diaria recibida e inyectada

FECHA	Energía DC total diaria (Wh)	Potencia DC media diaria (W)
24/01/2025	819,17	78,02
25/01/2025	1101,67	104,09
26/01/2025	678,33	64,09
27/01/2025	3683,33	348,03
28/01/2025	6523,33	616,38
29/01/2025	81,67	7,72
30/01/2025	7446,67	703,62
31/01/2025	8639,17	816,30
01/02/2025	8468,33	800,16
02/02/2025	5759,17	544,17
03/02/2025	8321,67	786,30
04/02/2025	8225,00	777,17
05/02/2025	8525,00	805,51
06/02/2025	8710,00	822,99
07/02/2025	2720,83	257,09
08/02/2025	6585,83	622,28
09/02/2025	7088,33	669,76
10/02/2025	3247,50	306,85
11/02/2025	7291,67	688,98
12/02/2025	5845,83	552,36
13/02/2025	7263,33	686,30
14/02/2025	8364,17	790,31
15/02/2025	5193,33	490,71
16/02/2025	7987,50	754,72
17/02/2025	5135,00	485,20
18/02/2025	5526,67	522,20
19/02/2025	5807,50	548,74
20/02/2025	5380,00	508,35
21/02/2025	2885,00	272,60
22/02/2025	5750,00	543,31
23/02/2025	9352,50	882,36
24/02/2025	5875,00	555,12
Energía DC total mensual (kWh)		184,28

FECHA	Energía AC total diaria (Wh)	Potencia AC media diaria (W)
24/01/2025	669,17	63,73
25/01/2025	840,83	79,45
26/01/2025	515,00	48,66
27/01/2025	3450,00	325,98
28/01/2025	6060,83	572,68
29/01/2025	118,33	11,18
30/01/2025	6933,33	655,12
31/01/2025	8009,17	756,77
01/02/2025	7824,17	739,29
02/02/2025	5346,67	505,20
03/02/2025	7795,00	736,54
04/02/2025	7698,33	727,40
05/02/2025	7942,50	750,47
06/02/2025	8094,17	764,80
07/02/2025	2411,67	227,87
08/02/2025	6029,17	569,69
09/02/2025	6594,17	623,07
10/02/2025	2852,50	269,53
11/02/2025	6784,17	641,02
12/02/2025	5412,50	511,42
13/02/2025	6707,50	633,78
14/02/2025	7672,50	724,96
15/02/2025	4659,17	440,24
16/02/2025	7376,67	697,01
17/02/2025	4667,50	441,02
18/02/2025	5073,33	479,37
19/02/2025	5257,50	496,77
20/02/2025	4962,50	468,90
21/02/2025	2475,83	233,94
22/02/2025	5279,17	498,82
23/02/2025	8601,67	812,76
24/02/2025	5341,67	504,72
Energía AC total mensual (kWh)		169,46

Los datos reflejados en la tabla 3, siguen un patrón muy similar a los representados en la tabla 2. Se observa que la energía total diaria en corriente continua varía desde un mínimo de 81,67 Wh el 29/01/2025 hasta un máximo de 9352,50 Wh el 23/02/2025, mientras que la potencia media diaria en corriente continua oscila entre 7,72 W y 882,36 W en los mismos días. En corriente alterna, la energía total diaria presenta un rango desde 118,33 Wh hasta 8601,67 Wh, con una potencia media diaria que fluctúa entre 11,18 W y 812,76 W con los valores máximos y mínimos en los mismos días que los de la potencia continua. Con ambas energías totales mensuales es posible calcular el rendimiento o eficiencia del inversor en % empleando la formula (8):

$$\eta = \frac{E_{AC}}{E_{DC}} * 100 \qquad \% \qquad (8)$$

Donde:

- E_{AC}: Es la energía AC total mensual (la que se inyecta a la red).
- E_{DC}: Es la energía DC total mensual (la que recibe el inversor de los módulos FV).

El rendimiento que se obtiene tras hacer dicha operación es de 91,95%, un valor ligeramente inferior al especificado en la hoja de datos del fabricante, que indica un rendimiento máximo del 98% y un rendimiento europeo de 96,1%. Esta diferencia puede deberse a que el modelo de inversor empleado tiene una potencia nominal CA de 3kW mientras que la potencia de la

instalación del laboratorio es de 1750W. En consecuencia, el inversor estará operando al 58% de su capacidad (1750 W / 3000 W) por lo que, si se desea mejorar la eficiencia de la instalación, sería recomendable emplear otro inversor con menor potencia nominal o aumentar la potencia instalada.

Finalmente, se analizará la relación entre la irradiación total diaria y la energía total diaria. Para ello, en la imagen 67 se presenta un gráfico con el ajuste de regresión lineal y el coeficiente de determinación (R^2) entre las variables mencionadas:

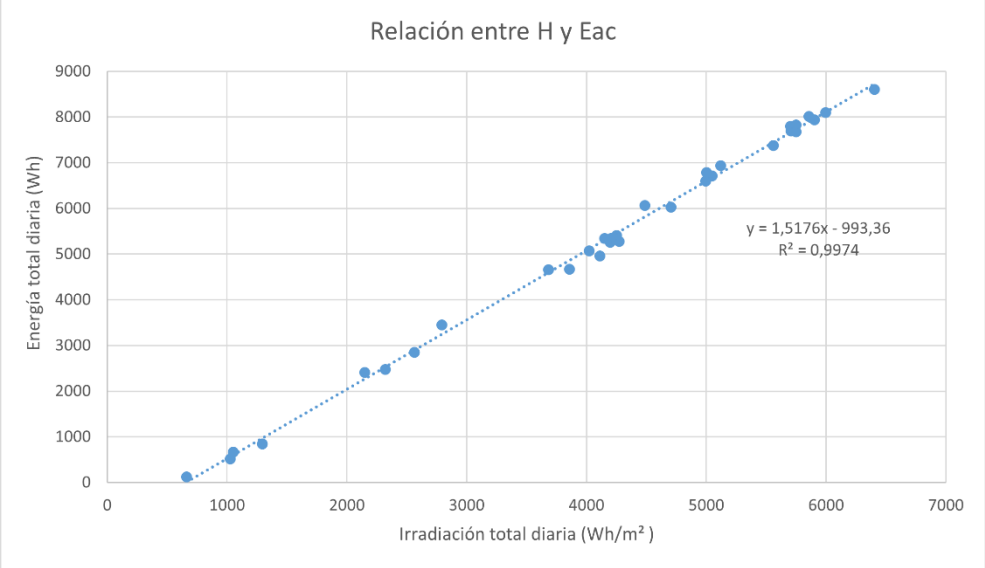

Figura 67: Relación entre la irradiación total diaria y la energía total diaria

El análisis de regresión lineal entre la irradiancia total diaria y la energía en CA total diaria muestra una relación fuertemente lineal, con un coeficiente de determinación $R^2 = 0{,}9974$ lo que sugiere una correlación casi perfecta entre estas dos variables y por lo tanto con el resto de las variables por lo que es posible destacar el papel de la irradiancia como factor determinante en la producción de energía solar.

6. Presupuesto

En este apartado se presenta el desglose del presupuesto total del proyecto, estructurado en dos categorías principales: costes asociados a los materiales utilizados y costes correspondientes a la mano de obra requerida.

6.1 Costes materiales

Los costes materiales incluyen todos los componentes utilizados para la implementación del sistema. No se han considerado los costos de la instalación FV ya que esta es propiedad de la universidad ni los costos del contador y los inversores puesto que fueron prestados para el desarrollo del proyecto y deberán ser devueltos. Del mismo modo, los costos asociados al software utilizado son nulos, dado que se han empleado alternativas de código abierto y gratuitas.

Tabla 4: Costes materiales

Material	Cantidad	Coste unitario (€)	Coste total (€)
PLC ESP32 PLC 21 +	1	225	225,00 €
Tarjeta Micro SD 8GB	1	10	10,00 €
Ordenador con Windows 10	1	300	300,00 €
Router WiFi	1	50	50,00 €
Tarjeta SIM (alquiler mensual x 4 meses)	4	10	40,00 €
Materiales adicionales (cables, conectores, etc.)	-	30	30,00 €
		Total costes materiales:	655,00 €

6.2 Costes por mano de obra

Para estimar los costes por mano de obra, se consideró el perfil de un ingeniero junior en telecomunicaciones con menos de un año de experiencia, cuyo costo promedio por hora se estima en 13€[58]. Se calculó un total de 340 horas necesarias para el diseño, desarrollo, pruebas y documentación del sistema, como se detalla a continuación:

Tabla 5: Costes por mano de obra

Actividad	Horas estimadas	Costo por hora (€)	Costo total (€)
Trabajo total (ingeniero junior)	340	13	4.420,00 €

6.3 Presupuesto total

El presupuesto total del proyecto asciende a 5.075,00 €, resultado de la suma de los costes asociados tanto a los materiales como a la mano de obra necesarios para su ejecución.

7. Impacto del proyecto

El impacto del sistema de monitorización aplicado al contexto de CE rurales, como el proyecto JALON, se centra en la mejora de la eficiencia energética y el fortalecimiento de estas comunidades. Más allá de abordar desafíos técnicos específicos, este sistema busca aportar soluciones sostenibles que transformen la forma en que estas comunidades gestionan y consumen energía, promoviendo una mayor autonomía y resiliencia local. En este capítulo se explorará el alcance de sus efectos en ámbitos sociales, ambientales, económicos y tecnológicos, así como su posible aportación a los ODS (Objetivos de Desarrollo Sostenible) [59].

7.1 Impacto social

El sistema de monitorización implementado contribuye a la inclusión social y al empoderamiento de los ciudadanos dentro de las CE rurales. Al proporcionar datos accesibles y en tiempo real sobre la producción y el consumo de energía, los miembros de estas comunidades pueden tomar decisiones informadas sobre su uso energético, fomentando un mayor sentido de participación y responsabilidad.

Además, al facilitar el acceso a herramientas tecnológicas avanzadas en zonas rurales, el sistema contribuye a reducir la brecha digital entre estas áreas y las zonas urbanas más desarrolladas. La disponibilidad de datos precisos permite a los ciudadanos optimizar su consumo y comprender mejor los beneficios del autoconsumo, reforzando la cohesión comunitaria en torno a la transición energética.

En este sentido, el sistema de monitorización respalda los objetivos del ODS 11 (Ciudades y comunidades sostenibles), promoviendo modelos energéticos accesibles y sostenibles, y el ODS 17 (Alianzas para lograr los objetivos) fomentando la colaboración entre ciudadanos, gobiernos y empresas locales en la gestión de la energía.

7.2 Impacto ambiental

La integración del sistema de monitoreo en CE favorece la reducción del impacto ambiental al optimizar el uso de la energía renovable disponible, minimizando pérdidas y maximizando la eficiencia de los recursos.

El acceso a datos en tiempo real sobre la producción y consumo permite a los usuarios adaptar su comportamiento para maximizar el aprovechamiento del sistema de autoconsumo, reduciendo así la compra de energía del mercado. En el caso de España, donde en 2024 un 44.2% de la electricidad generada provino de fuentes no renovables[60], esta optimización contribuye directamente a disminuir la dependencia de tecnologías con mayor impacto ambiental, reduciendo las emisiones de carbono y fomentando un uso más eficiente de la energía renovable disponible.

En línea con los objetivos del ODS 7 (Energía asequible y no contaminante) y el ODS 13 (Acción por el clima), el sistema de monitorización apoya una gestión energética más sostenible, contribuyendo a la lucha contra el cambio climático y promoviendo el uso responsable de los recursos energéticos.

7.3 Impacto económico

En el ámbito económico, el sistema de monitorización en tiempo real facilita una gestión eficiente del autoconsumo, con reducción de costes energéticos dentro de las comunidades. Su implementación permite una mejor planificación y mantenimiento del sistema FV, al detectar anomalías en la producción o el consumo, lo que evita fallos prolongados y reduce la necesidad de reparaciones costosas. Al mismo tiempo, el mantenimiento continuo y optimizado fomenta la demanda de servicios técnicos especializados, generando oportunidades de empleo local y fortaleciendo la economía de la comunidad.

Desde el punto de vista del usuario, la posibilidad de adaptación del consumo a la generación FV no solo se traduce en un ahorro económico directo, sino que también refuerza la autonomía energética de los participantes, reduciendo su exposición a la volatilidad de los precios de la electricidad y favoreciendo un modelo de consumo más eficiente y sostenible.

En este sentido, el sistema de monitorización contribuye al ODS 12 (Producción y consumo responsables), garantizando una gestión eficiente de los recursos energéticos y minimizando los desperdicios. Además, la creación de empleo local tanto para la instalación como para el mantenimiento de los sistemas FV, fortalece la economía local vinculada al ODS 8 (Trabajo decente y crecimiento económico), promoviendo economías locales justas y sostenibles, al tiempo que fortalece la resiliencia económica de las comunidades involucradas.

7.4 Impacto tecnológico

El sistema de monitorización desarrollado es una herramienta adaptable y replicable en distintas CE rurales, facilitando la innovación en la gestión de energía distribuida. Su enfoque basado en tecnologías de código abierto permite su escalabilidad y personalización según las necesidades específicas de cada comunidad, fomentando el desarrollo tecnológico en el sector energético.

La implementación de este sistema promueve una infraestructura energética más moderna y digitalizada, alineándose con el ODS 9 (Industria, innovación e infraestructura). Su capacidad para integrar diferentes fuentes de generación y analizar datos en tiempo real permite mejorar la eficiencia operativa de las comunidades energéticas y servir como referencia para futuros desarrollos en la gestión de energía renovable.

8. Conclusiones y trabajos futuros

En este apartado se presentan las principales conclusiones extraídas del desarrollo del proyecto, así como una serie de propuestas para futuros trabajos que podrían expandir o mejorar los resultados obtenidos.

8.1 Conclusiones

A lo largo del proyecto se logró diseñar e implementar un sistema de monitorización orientado a las necesidades específicas de CE rurales con pequeñas instalaciones FV. Este trabajo respondió a la carencia de herramientas accesibles y personalizadas para estas comunidades, que dependían de soluciones diseñadas para grandes plantas FV, inadecuadas por sus altos costos y complejidad.

Durante el proyecto, se llevaron a cabo las etapas clave de adquisición y transmisión de datos, almacenamiento y visualización, implementando soluciones hardware y software para garantizar la viabilidad técnica del sistema. El sistema se probó en un entorno controlado de laboratorio, logrando la correcta adquisición de datos de inversores, contadores y módulos de referencia. Además, se logró transmitir datos utilizando el bróker Mosquitto, integrar exitosamente el almacenamiento en InfluxDB y finalmente visualizar la información en tiempo real mediante dashboards en Grafana. La solución demostró ser robusta y adaptable a diferentes configuraciones de instalaciones fotovoltaicas.

El sistema demostró ser robusto y eficiente. No obstante, se identificaron limitaciones importantes, como la dependencia de una conexión a internet estable para el óptimo rendimiento de las funcionalidades avanzadas o la necesidad de emplear un ordenador en el que estén alojadas todas las aplicaciones. Esto representa un aspecto clave a abordar en futuras mejoras.

8.2 Trabajos futuros

Aunque el proyecto ha cumplido con los objetivos propuestos, existen varias áreas de mejora y ampliación:

- Monitorización de otros parámetros: Ampliar el alcance del sistema para incluir el monitoreo de variables adicionales como registros de errores de los dispositivos, rendimiento y energía generada en cada *string* fotovoltaico. Esto permitiría un análisis más completo y la detección temprana de anomalías.

- Contenerización de aplicaciones: Migrar los componentes del sistema a contenedores mediante herramientas como Docker para simplificar su despliegue, mantenimiento y escalabilidad. Por ejemplo, contenerizar el bróker Mosquitto, InfluxDB y Grafana permitiría implementar el sistema en diferentes entornos con mayor facilidad. Además, la utilización de servicios en la nube, como Grafana Cloud o InfluxDB Cloud, podría eliminar la necesidad de hardware local dedicado, reduciendo los costos

operativos y facilitando el acceso desde ubicaciones remotas. Sin embargo, esta decisión implica considerar aspectos como la seguridad, la privacidad de los datos y los costos recurrentes asociados a los servicios en la nube.

- App para el móvil: Desarrollar una aplicación móvil para permitir a los usuarios acceder a los datos del sistema desde sus dispositivos en cualquier momento y lugar. La aplicación podría incluir funciones como alertas en tiempo real, visualización de datos históricos y recomendaciones para mejorar el autoconsumo energético. Esta solución aumentaría la accesibilidad y facilitaría la gestión de la producción y consumo energético para los miembros de la comunidad.

A modo de conclusión, la implementación del sistema en un entorno real representa un paso fundamental para validar la funcionalidad del mismo. Una prueba piloto en alguna de las instalaciones fotovoltaicas ya operativas de la comarca de Calatayud permitiría obtener datos precisos sobre su desempeño y recopilar retroalimentación directa de los usuarios. Esto serviría para realizar ajustes y optimizaciones basados en necesidades reales y expectativas de los usuarios finales. También sería una oportunidad para detectar posibles mejoras y ampliar el alcance funcional del sistema en términos de nuevas funcionalidades y compatibilidades.

9. Referencias

[1] Jorge Díaz Lanchas y Alejandro Labanda, “La transición energética hacia 2030, ¿cuáles son las oportunidades económicas para los territorios rurales? - Funcas.” Accessed: Aug. 21, 2024. [Online]. Available: https://www.funcas.es/articulos/la-transicion-energetica-hacia-2030-cuales-son-las-oportunidades-economicas-para-los-territorios-rurales/

[2] “Sobre – Jalon-CE.” Accessed: Aug. 21, 2024. [Online]. Available: https://jalon-ce.eu/es/sobre-el-proyecto-jalon/

[3] S. O. M. Boulanger, M. Massari, D. Longo, B. Turillazzi, and C. A. Nucci, “Designing collaborative energy communities: A european overview,” *Energies (Basel)*, vol. 14, no. 24, Dec. 2021, doi: 10.3390/en14248226.

[4] “¿Qué es el autoconsumo? | Idae.” Accessed: Jan. 06, 2025. [Online]. Available: https://www.idae.es/tecnologias/energias-renovables/oficina-de-autoconsumo/que-es-el-autoconsumo

[5] J. Roberts, D. Frieden, J. Research,) Stanislas D’herbemont, and R. Eu), “Energy Community Definitions,” May 2019. Accessed: Dec. 21, 2024. [Online]. Available: https://main.compile-project.eu/wp-content/uploads/Explanatory-note-on-energy-community-definitions.pdf

[6] A. C. Lazaroiu, M. Roscia, G. C. Lazaroiu, and P. Siano, “Review of Energy Communities: Definitions, Regulations, Topologies, and Technologies,” *Smart Cities 2025, Vol. 8, Page 8*, vol. 8, no. 1, p. 8, Jan. 2025, doi: 10.3390/SMARTCITIES8010008.

[7] D. de São José, P. Faria, and Z. Vale, “Smart energy community: A systematic review with metanalysis,” *Energy Strategy Reviews*, vol. 36, Jul. 2021, doi: 10.1016/j.esr.2021.100678.

[8] S. Soeiro and M. Ferreira Dias, “Renewable energy community and the European energy market: main motivations,” *Heliyon*, vol. 6, no. 7, Jul. 2020, doi: 10.1016/j.heliyon.2020.e04511.

[9] A. Caramizaru, “Energy communities: an overview of energy and social innovation,” 2020, doi: 10.2760/180576.

[10] “Notre histoire - Enercoop.” Accessed: Dec. 29, 2024. [Online]. Available: https://www.enercoop.fr/notre-projet/notre-histoire

[11] “100 % Ökostrom – Klimaschutz mit Rebellenkraft | EWS Schönau.” Accessed: Dec. 29, 2024. [Online]. Available: https://www.ews-schoenau.de/

[12] “Eigg Electric - The Isle of Eigg.” Accessed: Dec. 29, 2024. [Online]. Available: http://isleofeigg.org/eigg-electric/

[13] "Energy agencies and Renewable Energy Communities. A new path for energy decentralization - Energy Cities." Accessed: Jun. 11, 2025. [Online]. Available: https://energy-cities.eu/event/energy-agencies-and-renewable-energy-communities-a-new-path-for-energy-decentralization/

[14] I. Capellán-Pérez, Á. Campos-Celador, and J. Terés-Zubiaga, "Renewable Energy Cooperatives as an instrument towards the energy transition in Spain," *Energy Policy*, vol. 123, pp. 215–229, Dec. 2018, doi: 10.1016/J.ENPOL.2018.08.064.

[15] ECODES, "Informe de Indicadores 2023: Observatorio de Comunidades Energéticas Energía Común." Accessed: Jan. 02, 2025. [Online]. Available: https://ecodes.org/images/que-hacemos/03.Energia_y_personas/Energía_común/240611_Informe_2023_Energía_Común.pdf

[16] T. y C. Ministerio de Industria, "BOE-A-2011-19242 Real Decreto 1699/2011, de 18 de noviembre, por el que se regula la conexión a red de instalaciones de producción de energía eléctrica de pequeña potencia." Accessed: Jan. 02, 2025. [Online]. Available: https://www.boe.es/buscar/doc.php?id=BOE-A-2011-19242

[17] E. y T. Ministerio de Industria, "BOE-A-2015-10927 Real Decreto 900/2015, de 9 de octubre, por el que se regulan las condiciones administrativas, técnicas y económicas de las modalidades de suministro de energía eléctrica con autoconsumo y de producción con autoconsumo." Accessed: Jan. 02, 2025. [Online]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-2015-10927

[18] Jefatura del Estado, "BOE-A-2018-13593 Real Decreto-ley 15/2018, de 5 de octubre, de medidas urgentes para la transición energética y la protección de los consumidores." Accessed: Jan. 02, 2025. [Online]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-2018-13593

[19] Ministerio para la Transición Ecológica, "BOE-A-2019-5089 Real Decreto 244/2019, de 5 de abril, por el que se regulan las condiciones administrativas, técnicas y económicas del autoconsumo de energía eléctrica." Accessed: Jan. 02, 2025. [Online]. Available: https://www.boe.es/buscar/doc.php?id=BOE-A-2019-5089

[20] "Som Energia | La Cooperativa d'Energia Verda." Accessed: Jan. 05, 2025. [Online]. Available: https://www.somenergia.coop/

[21] "Electricidad 100% renovable y ética - Energética Coop." Accessed: Jan. 05, 2025. [Online]. Available: https://www.energetica.coop/

[22] "AURORA H2020." Accessed: Jan. 05, 2025. [Online]. Available: https://www.aurora-h2020.eu/um-main/

[23] "CERCA Energía | Comunidad de Energías Renovables de la Comarca de Calatayud." Accessed: Jun. 11, 2025. [Online]. Available: https://cercaenergia.com/

[24] "Directive - 2018/2001 - EN - EUR-Lex." Accessed: Jan. 06, 2025. [Online]. Available: https://eur-lex.europa.eu/eli/dir/2018/2001/oj/eng

[25] "Hoja de Ruta del Autoconsumo | Idae." Accessed: Jan. 06, 2025. [Online]. Available: https://www.idae.es/tecnologias/energias-renovables/oficina-de-autoconsumo/hoja-de-ruta-del-autoconsumo

[26] N. Forero, J. Hernández, and G. Gordillo, "Development of a monitoring system for a PV solar plant," *Energy Convers Manag*, vol. 47, no. 15–16, pp. 2329–2336, Sep. 2006, doi: 10.1016/J.ENCONMAN.2005.11.012.

[27] L. Hui, W. Gui-Rong, W. Jian-Ping, and D. Peiyong, "Monitoring platform of energy management system for smart community," *Proceedings of the 29th Chinese Control and Decision Conference, CCDC 2017*, pp. 1832–1836, Jul. 2017, doi: 10.1109/CCDC.2017.7978814.

[28] S. R. Madeti and S. N. Singh, "Monitoring system for photovoltaic plants: A review," *Renewable and Sustainable Energy Reviews*, vol. 67, pp. 1180–1207, Jan. 2017, doi: 10.1016/J.RSER.2016.09.088.

[29] "ESP32 PLC Ethernet 21 I/Os analógico / digital." Accessed: Jan. 14, 2025. [Online]. Available: https://www.industrialshields.com/es_ES/shop/esp32-plc-21-2801#attr=2238,165,2304,5848,5849,5850,2305,5851,3757,6641,6092,6109

[30] "Especificaciones del Smart Energy Controller_Especificaciones del inversor solar | Huawei FusionSolar." Accessed: Jan. 14, 2025. [Online]. Available: https://solar.huawei.com/es/professionals/all-products/SUN2000-3-4-5-6KTL-L1/specs

[31] "Fronius Primo 3.0-1." Accessed: Jan. 14, 2025. [Online]. Available: https://www.fronius.com/es-es/spain/energia-solar/instaladores-y-socios/datos-tecnicos/todos-los-productos/inversor/fronius-primo/fronius-primo-3-0-1

[32] "Inversor de conexión a red inteligente 0,7-8 kW ." Accessed: Jan. 14, 2025. [Online]. Available: https://sp.saj-electric.com/es-es/r5-series

[33] "CONTAX D-6041-BUS." Accessed: Jan. 14, 2025. [Online]. Available: https://orbis.es/productos/medida-y-gestion-de-energia-agua-y-gas/contadores-industriales-multifuncion/monofasicos-contadores-industriales-multifuncion/contax-d-6041-bus/

[34] "Placa Solar policristalina 50W 12V ESPMC-50." Accessed: Jan. 14, 2025. [Online]. Available: https://atersa.shop/producto/panel-solar-espmc-50w/?srsltid=AfmBOooHvF2udf2cGZ75uVXPwUy0eKVr1qNSZRHVqslOZywPPQtERw10

[35] J. Polo, W. G. Fernandez-Neira, and M. C. Alonso-García, "On the use of reference modules as irradiance sensor for monitoring and modelling rooftop PV systems," *Renew Energy*, vol. 106, pp. 186–191, Jun. 2017, doi: 10.1016/J.RENENE.2017.01.026.

[36] "Modbus RTU: El protocolo industrial de comunicación por excelencia - Ditel Diseños y Tecnología S.A." Accessed: Jan. 15, 2025. [Online]. Available: https://www.ditel.es/modbus-rtu-el-protocolo-industrial-de-comunicacion-por-excelencia/

[37] "The Modbus Organization." Accessed: Jan. 15, 2025. [Online]. Available: https://modbus.org/

[38] "Modbus Organization, 'Modbus application protocol specification V1.1b3,.'" Accessed: Jan. 15, 2025. [Online]. Available: https://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[39] "¿Qué es MQTT? Su importancia como protocolo IoT." Accessed: Jan. 16, 2025. [Online]. Available: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

[40] "Explicación detallada del protocolo MQTT_internet de las cosas industrial_Casos de aplicación_Blog_Chengdu Ebyte Electronic Technology Co.,Ltd-ES." Accessed: Jan. 16, 2025. [Online]. Available: https://www.es-ebyte.com/news/488#a3.

[41] "MQTT - Guía con Toda la información que debes saber de este sistema." Accessed: Jan. 16, 2025. [Online]. Available: https://descubrearduino.com/mqtt-que-es-como-se-puede-usar-y-como-funciona/

[42] "Comunicando Flutter com NodeMCU usando protocolo MQTT | by Ricardo Ogliari | Medium." Accessed: Jan. 16, 2025. [Online]. Available: https://ricardoogliari.medium.com/comunicando-flutter-com-nodemcu-usando-protocolo-mqtt-8be193a9b4bc

[43] "MQTT vs HTTP: ¿qué protocolo es mejor para IoT? – BorrowBits." Accessed: Jan. 16, 2025. [Online]. Available: https://borrowbits.com/2020/04/mqtt-vs-http-que-protocolo-es-mejor-para-iot/

[44] "Software | Arduino." Accessed: Jan. 16, 2025. [Online]. Available: https://www.arduino.cc/en/software/

[45] "Eclipse Mosquitto." Accessed: Jan. 17, 2025. [Online]. Available: https://mosquitto.org/

[46] "home - Node-RED." Accessed: Jan. 17, 2025. [Online]. Available: https://nodered.org/

[47] "InfluxDB OSS v2 Documentation." Accessed: Apr. 18, 2025. [Online]. Available: https://docs.influxdata.com/influxdb/v2/

[48] “Grafana OSS and Enterprise | Grafana documentation.” Accessed: Jan. 18, 2025. [Online]. Available: https://grafana.com/docs/grafana/latest/

[49] “Tailscale · Best VPN Service for Secure Networks.” Accessed: Jan. 21, 2025. [Online]. Available: https://tailscale.com/

[50] “Variaciones de la tensión, intensidad y potencia con la irradiancia y la temperatura. - Formación para la Industria 4.0.” Accessed: Jan. 21, 2025. [Online]. Available: https://automatismoindustrial.com/curso-energia-solar-fotovoltaica/componentes-energia-solar-fotovoltaica/variaciones-de-la-tension-intensidad-y-potencia-con-la-irradiancia-y-la-temperatura/

[51] “RTC - ArduWiki.” Accessed: Nov. 19, 2024. [Online]. Available: https://arduwiki.perut.org/index.php/RTC

[52] “Conceptos básicos: tarjeta SD en un PLC industrial.” Accessed: Nov. 21, 2024. [Online]. Available: https://www.industrialshields.com/es_ES/blog/blog-industrial-open-source-1/conceptos-basicos-tarjeta-sd-en-un-plc-industrial-167

[53] “GitHub - knolleary/pubsubclient: A client library for the Arduino Ethernet Shield that provides support for MQTT.” Accessed: Nov. 15, 2024. [Online]. Available: https://github.com/knolleary/pubsubclient

[54] “Library - Node-RED.” Accessed: Dec. 01, 2025. [Online]. Available: https://flows.nodered.org/

[55] “Annotated CSV | InfluxDB Cloud (TSM) Documentation.” Accessed: Dec. 18, 2025. [Online]. Available: https://docs.influxdata.com/influxdb/cloud/reference/syntax/annotated-csv/

[56] “Create an InfluxDB dashboard | InfluxDB OSS v2 Documentation.” Accessed: Dec. 18, 2025. [Online]. Available: https://docs.influxdata.com/influxdb/v2/visualize-data/dashboards/create-dashboard/

[57] “Suma de Riemann - Calculointegral.com.” Accessed: Jan. 26, 2025. [Online]. Available: https://calculointegralweb.com/suma-de-riemann/

[58] “¿Cuánto Cobra un Ingeniero de Telecomunicaciones? (Sueldo 2025) | Jobted.es.” Accessed: Jan. 30, 2025. [Online]. Available: https://www.jobted.es/salario/ingeniero-telecomunicaciones

[59] “Objetivos y metas de desarrollo sostenible - Desarrollo Sostenible.” Accessed: Feb. 01, 2025. [Online]. Available: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

[60] “La demanda de energía eléctrica en España aumenta un 1,5% en diciembre | Red Eléctrica.” Accessed: Feb. 01, 2025. [Online]. Available: https://www.ree.es/es/sala-de-prensa/actualidad/nota-de-prensa/2025/01/la-demanda-de-energia-electrica-en-espana-aumenta-un-uno-cinco-por-ciento-en-diciembre

Anexo: Manual de usuario

Este manual de usuario detalla las configuraciones adicionales necesarias para garantizar el correcto funcionamiento del sistema. A continuación, se presentan las instrucciones y pasos clave para su correcta puesta en marcha.

A.1 Conexiones

A.1.1 Conexiones PLC

Para que el sistema funcione correctamente, es fundamental realizar las conexiones adecuadas en el PLC. A continuación, se describen los pasos a seguir:

1. Alimentación del PLC

El PLC se alimentará mediante una fuente de alimentación conmutada, la cual proporciona una tensión de 24V para su correcto funcionamiento. Para ello, el PLC dispone de un conector especial ubicado en la parte inferior derecha, diseñado específicamente para esta funcionalidad.

2. RS-485 (Modbus RTU)

Para establecer la comunicación mediante RS-485, es necesario habilitar esta función en el PLC. Para ello, se debe localizar los interruptores situados en la parte inferior izquierda del PLC y situar el switch 3 en la posición OFF (ver figura 68). Posteriormente, se deben conectar los cables A+ y B- (también denominados D+ y D- respectivamente) de la zona 0 del PLC.

El sistema permite la conexión de hasta 32 dispositivos mediante el protocolo Modbus RTU, con una capacidad de transmisión de datos de hasta 1200 metros. Además, el PLC dispone de una línea GND para la comunicación RS-485, la cual puede ayudar a reducir interferencias. Se realizaron pruebas comparando la comunicación con y sin esta línea conectada, sin encontrar diferencias significativas en el rendimiento. Otra de las opciones garantizar una comunicación estable y minimizar posibles interferencias, se recomienda el uso de resistencias de terminación en los extremos del bus de comunicación.

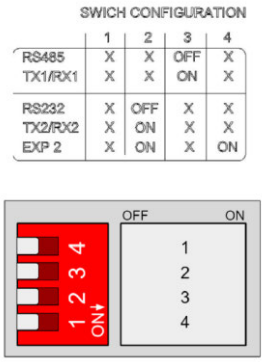

Figura 68: Configuración switches PLC

3. Entradas Analógicas

Las entradas analógicas empleadas se sitúan en la zona A de conexiones del PLC. En concreto, el modelo utilizado dispone de hasta 6 entradas analógicas, por lo que se pueden seleccionar las que se desee. En este caso, se han utilizado las entradas IO_12 e IO_11.

Para las líneas de GND de dichas entradas, se puede emplear cualquiera de las disponibles en el PLC, ya que todas las conexiones GND son comunes dentro del sistema. El esquema general de conexiones puede verse en la figura 69:

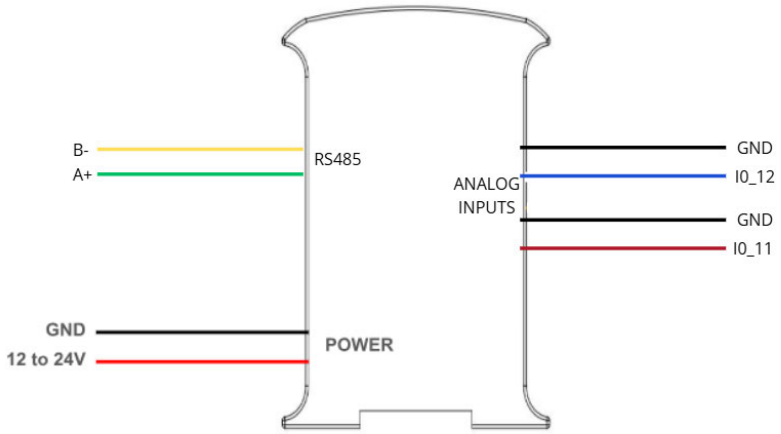

Figura 69: Esquema general de conexiones del PLC

4. Tarjeta microSD

La ranura para la tarjeta microSD se encuentra en la parte superior del PLC. Para acceder a ella, es necesario remover la tapa y con la ayuda de unas pinzas, colocar cuidadosamente la tarjeta en la ranura correspondiente.

5. Conexión ordenador

Para conectar el PLC con el ordenador y programarlo se requiere un cable micro-USB tipo B. Este cable debe conectarse al ordenador mediante el puerto USB estándar y al PLC a través del puerto micro-USB tipo B, el cual se encuentra ubicado en el lado derecho del PLC

A.1.1 Conexión RS485 del inversor SUN2000

Para conectar el inversor SUN2000 al bus Modbus, es necesario instalar el adaptador de los cables de señal del inversor, ubicado en la parte inferior del equipo. Este adaptador cuenta con un conector de pines, de los cuales solo son relevantes los pines 1 y 2 para la comunicación Modbus RTU, correspondiendo el pin 1 con el cable B- y el 2 con A+. En la figura 70 puede verse el conexionado:

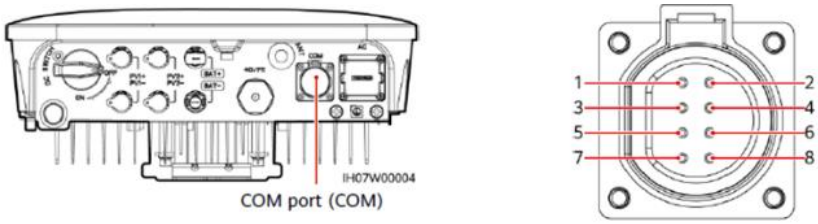

Figura 70: Conexión RS485 del inversor SUN2000

Los parámetros de configuración de la conexión, como la dirección Modbus, paridad, bit de stop y baudrate, pueden ser configurados a través de la aplicación móvil FusionSolar.

A.1.2 Conexión RS485 del inversor FRONIUS PRIMO

Para conectar el inversor Fronius Primo al bus Modbus, es necesario remover la tapa inferior del inversor y localizar el conector de tornillos, donde se deben instalar los cables según el esquema mostrado en la imagen 71:

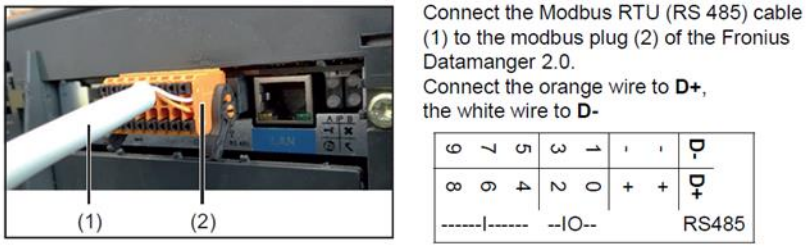

Figura 71: Conexión RS485 del inversor Fronius Primo

La dirección Modbus puede configurarse mediante la interfaz de usuario y pantalla del inversor, mientras que el resto de los parámetros de comunicación pueden ser ajustados a través de la aplicación móvil Solar Start.

A.1.3 Conexión RS485 del inversor SAJ R5

El inversor SAJ R5 emplea un conector RJ45 para su comunicación RS-485, ubicado en la parte inferior del dispositivo. Según el *datasheet*, los cables de interés deberían ser el 7 y el 8, que corresponden con A+ y B-. Sin embargo, tras realizar numerosas pruebas con esta configuración sin obtener resultados, se consultó directamente con el fabricante. Tras esta consulta, se confirmó que los cables correctos para la comunicación son, independientemente del tipo de RJ45, el cable marrón para A+ y el cable marrón a líneas para B-. En la figura 72 se ilustra el conexionado:

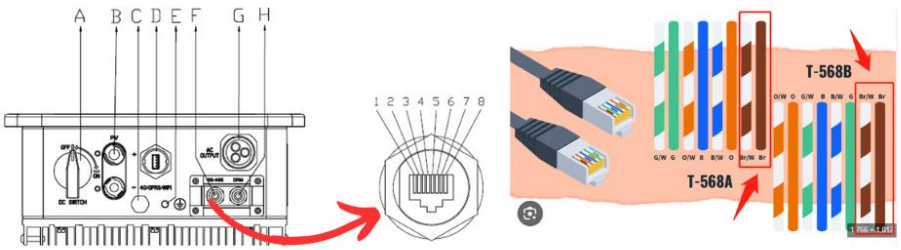

Figura 72: Conexión RS485 del inversor SAJ R5

Todos los parámetros de comunicación pueden configurarse a través de la aplicación móvil eSolarO&M.

A.1.4 Conexión RS485 del contador

El contador CONTAX D-6041-BUS utiliza un conector de tornillos para la comunicación mediante Modbus RTU. Este conector se encuentra ubicado en la parte superior del contador, y los cables deben conectarse siguiendo el esquema mostrado en la imagen 73.

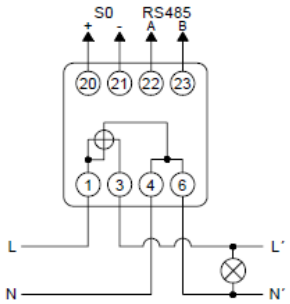

Figura 73: Conexión RS485 del contador CONTAX D-6041-BUS

A.2 Instalación y configuración de los programas.

A.2.1 Mosquitto

Para garantizar el correcto funcionamiento del bróker Mosquitto, es necesario instalarlo y configurarlo adecuadamente en el ordenador que actuará como servidor localhost. La instalación se realiza descargando el software desde la página oficial de Mosquitto, ejecutando el instalador y asegurando que se seleccione la opción para instalar los servicios. Una vez finalizada la instalación, se puede verificar su funcionamiento abriendo CMD y ejecutando mosquitto -v, lo que confirmará que el bróker está en ejecución en el puerto 1883. Para que Mosquitto se inicie automáticamente cada vez que el ordenador se encienda, se ha configurado su inicio retrasado a través del administrador de servicios de Windows como se ve en la figura 74. Esta opción permite que el bróker arranque automáticamente después de que el sistema haya cargado los servicios esenciales, evitando posibles conflictos en el inicio.

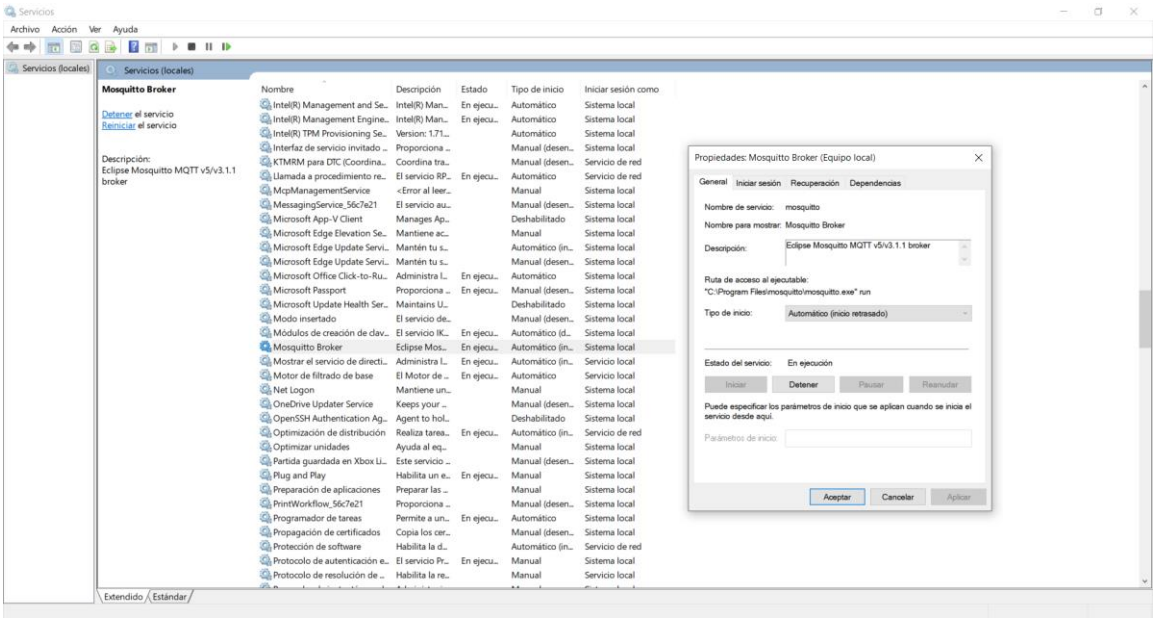

Figura 74: Inicio automático Mosquitto

Para que Mosquitto funcione correctamente en localhost, es necesario editar el archivo de configuración *mosquitto.conf*, ubicado en la carpeta de instalación del programa *C:\Program Files\Mosquitto*. Se debe abrir con un editor de texto y añadir las líneas resaltadas de la imagen 75, las cuales son necesarias para permitir conexiones sin autenticación en el equipo local.

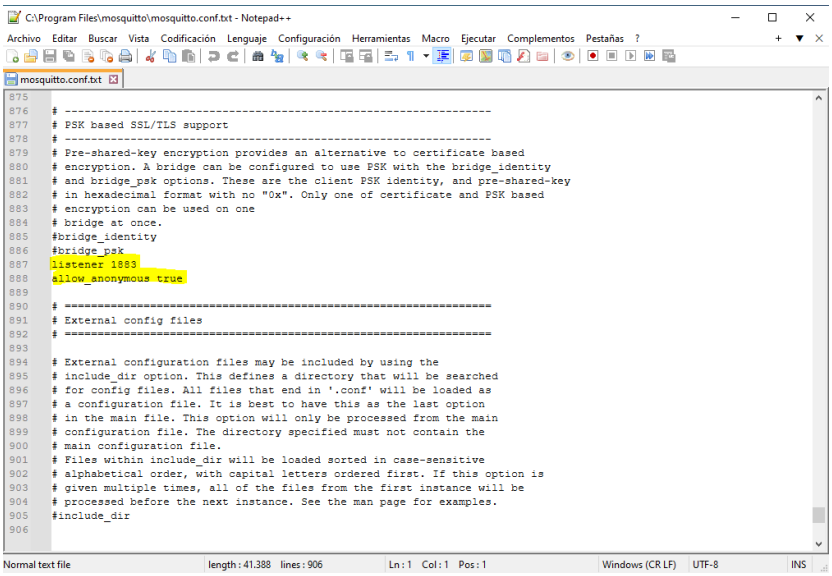

Figura 75: Archivo "mosquitto.conf"

Otra configuración importante es asegurarse de que el Firewall de Windows no bloquee las conexiones entrantes hacia Mosquitto. Para ello, se debe crear una regla de entrada en el Firewall de Windows Defender que permita el tráfico en el puerto 1883 tal y como se ve en la figura 76. Esto se realiza abriendo Windows Defender Firewall con seguridad avanzada, y en la sección de reglas de entrada, se debe agregar y configurar la siguiente regla para permitir que otros dispositivos en la red puedan comunicarse con el bróker Mosquitto sin restricciones.

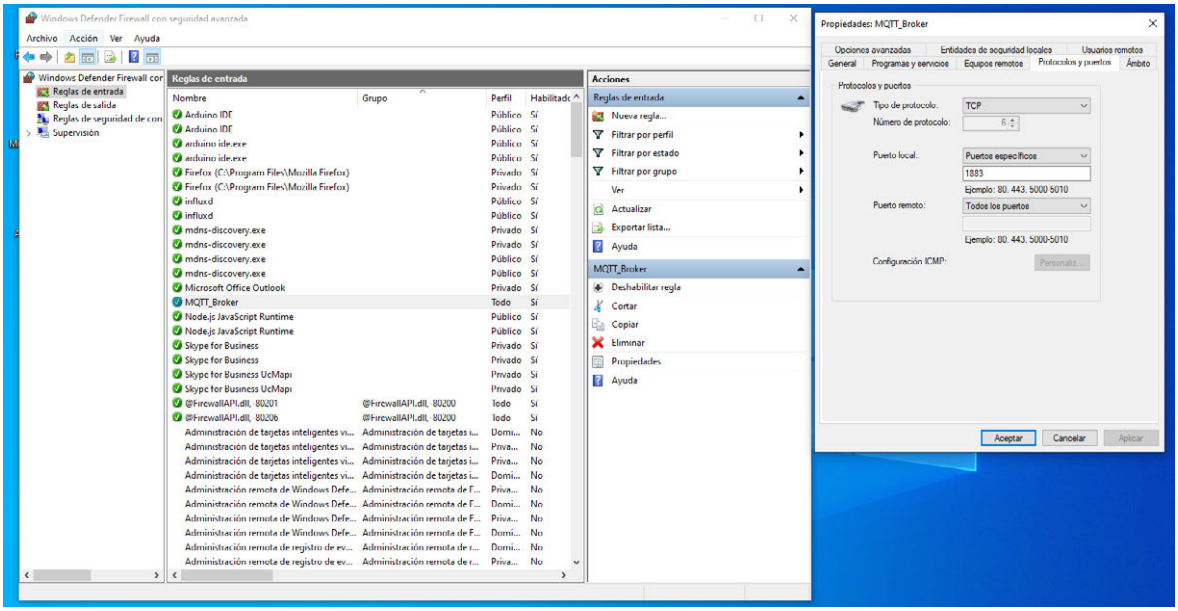

Figura 76: Configuración regla de entrada Mosquitto

Con estas configuraciones adicionales, el bróker Mosquitto está accesible desde otros dispositivos dentro de la misma red local, permitiendo la comunicación MQTT entre el PLC y otros sistemas. Finalmente, para comprobar que Mosquitto está funcionando correctamente, es posible realizar publicaciones y suscripciones desde dos instancias diferentes de CMD ejecutando los comandos de la imagen 77:

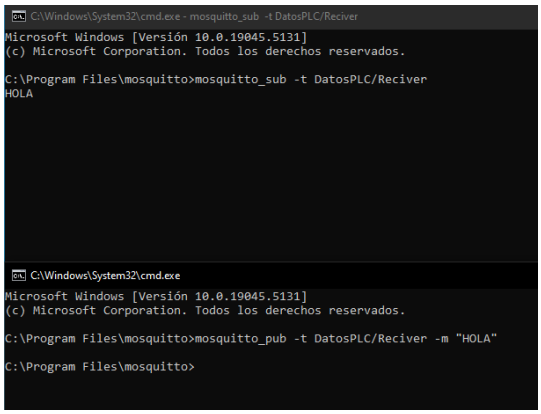

Figura 77: Publicación/Suscripción en Mosquito usando el CMD

Si el mensaje aparece en la primera ventana, se confirma que Mosquitto está funcionando correctamente y aceptando conexiones por lo que el servicio queda correctamente instalado y operativo.

A.2.2 Node-RED

Para instalar Node-RED, es imprescindible contar previamente con Node.js, ya que esta plataforma se ejecuta sobre dicho entorno. La instalación de Node.js puede realizarse descargándolo desde su sitio web oficial y siguiendo las instrucciones proporcionadas. Una vez instalado, se procede a la instalación de Node-RED mediante Node Package Manager (npm), ejecutando en la línea de comandos con privilegios de administrador el siguiente

comando: *npm install -g --unsafe-perm node-red*. Finalizado este proceso, Node-RED puede iniciarse en cualquier momento ejecutando el comando *node-red*, tras lo cual la consola mostrará información relevante y confirmará que el servicio se ha iniciado correctamente. De forma predeterminada, Node-RED opera en el puerto 1880, permitiendo el acceso a su interfaz web a través de un navegador mediante la dirección http://localhost:1880. La imagen 78 muestra el proceso de instalación e inicialización de Node-red:

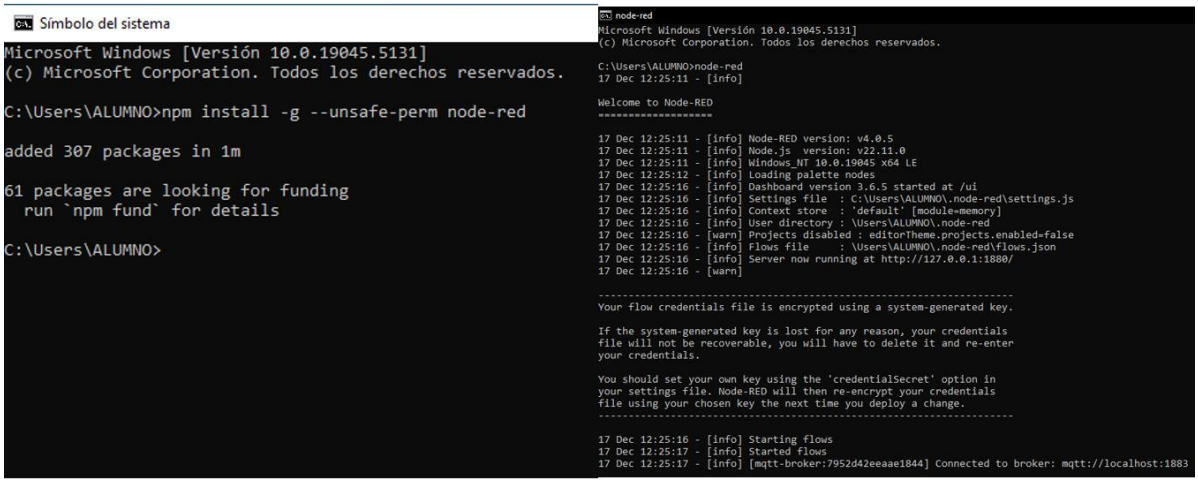

Figura 78: Instalación e inicialización Node-RED

A.2.3 InfluxDB

La instalación de InfluxDB v2 puede realizarse a través de su sitio web oficial, donde se proporciona un comando para ejecutar en PowerShell de Windows. No obstante, este método puede presentar inconvenientes, por lo que se recomienda copiar la dirección del instalador, pegarla en un navegador web y proceder con la descarga manualmente.

Una vez instalado, el servicio de InfluxDB puede iniciarse en cualquier momento accediendo a la carpeta donde se encuentra instalado el programa (generalmente C:\Program Files\InfluxData) y desde allí abrir el CMD y ejecutar el comando *influxdb.exe*.

Una vez iniciado InfluxDB, se puede acceder a su interfaz web a través de un navegador ingresando la dirección http://localhost:8086. En esta interfaz, se debe realizar la configuración inicial, que incluye la creación de un usuario administrador con unas credenciales que nos permitirán futuros accesos, una organización y un bucket como se ve en la imagen 79.

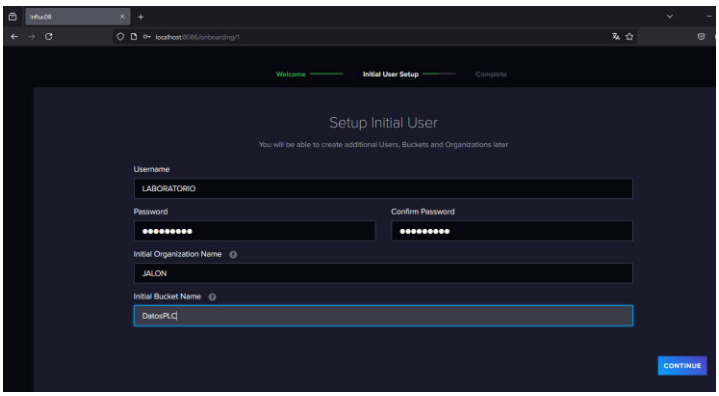

Figura 79: Configuración inicial InfluxDB

A.2.4 Grafana

La instalación de Grafana en Windows se puede realizar descargando el instalador desde su sitio web oficial y siguiendo las instrucciones proporcionadas. Una vez descargado, se ejecuta el instalador y se completan los pasos indicados en el asistente de instalación.

Tras finalizar el proceso, es recomendable configurar su inicio automático de manera similar a Mosquitto, asegurando que el servicio de Grafana se inicie junto con el sistema operativo. Asimismo, se debe establecer una regla de entrada en el firewall, tal como se hizo con Mosquitto, permitiendo el acceso a través del puerto por defecto (3000) para evitar restricciones de conexión.

Una vez instalado y configurado, Grafana puede iniciarse y accederse desde un navegador ingresando la dirección http://localhost:3000. Al acceder por primera vez, se solicitará autenticación con las credenciales predeterminadas (usuario: admin, contraseña: admin) y dará la posibilidad de cambiar dichas credenciales.