UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Monitorización y optimización de una planta solar híbrida de bombeo de agua y fotovoltaica

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Ros Bisbal, Jose

Tutor/a: Valderas Aranda, Pedro José

Cotutor/a externo: Garrido Sarasol, José Ángel

CURSO ACADÉMICO: 2023/2024

Resum

El present Treball de Fi de Grau(TFG) aborda el disseny, desenvolupament i implementació d'una planta solar híbrida fotovoltaica i de bombament d'aigua en un llogaret africà(Zimbàbue). Aquest projecte té com a objectiu principal proporcionar una solució sostenible per al subministrament d'aigua potable i energia elèctrica en comunitats rurals desfavorides mitjançant la integració d'energies renovables. Al llarg del treball, s'ha realitzat una selecció acurada dels subsistemes, del sistema d'emmagatzematge d'energia i del programari de monitoratge, tot això orientat a maximitzar l'eficiència i minimitzar els costos.

Inicialment, es va desenrotllar una planta de prova en un entorn controlat, la qual cosa va permetre identificar i solucionar possibles fallades abans de la instal·lació definitiva. La posada en marxa en el llogaret africà va ser un èxit, garantint un subministrament constant d'aigua i millorant significativament la qualitat de vida dels habitants. A més, el projecte ha demostrat ser reproduïble i escalable, la qual cosa ha motivat la planificació de noves instal·lacions en altres comunitats de la regió.

Les conclusions del projecte destaquen la importància d'un enfocament holístic que integra aspectes tècnics, socials i ambientals, i se subratllen les oportunitats per a futures millores i innovacions. Aquest TFG no sols proporciona una solució tècnica eficaç, sinó que també contribueix al desenvolupament sostenible i a l'apoderament de comunitats rurals a Àfrica.

Paraules clau: Monitoratge, Energia solar, Bombament d'aigua, HTTP, RS232, RS485, MODBUS TCP/RTU, Node.js, Subsistema, Planta Híbrida

Resumen

El presente Trabajo de Fin de Grado(TFG) aborda el diseño, desarrollo e implementación de una planta solar híbrida fotovoltaica y de bombeo de agua en una aldea africana(Zimbabue). Este proyecto tiene como objetivo principal proporcionar una solución sostenible para el suministro de agua potable y energía eléctrica en comunidades rurales desfavorecidas mediante la integración de energías renovables. A lo largo del trabajo, se ha realizado una selección cuidadosa de los subsistemas, del sistema de almacenamiento de energía y del software de monitorización, todo ello orientado a maximizar la eficiencia y minimizar los costos.

Inicialmente, se desarrolló una planta demo en un entorno controlado, lo que permitió identificar y solucionar posibles fallos antes de la instalación definitiva. La puesta en marcha en la aldea africana fue un éxito, garantizando un suministro constante de agua y mejorando significativamente la calidad de vida de los habitantes. Además, el proyecto ha demostrado ser replicable y escalable, lo que ha motivado la planificación de nuevas instalaciones en otras comunidades de la región.

Las conclusiones del proyecto destacan la importancia de un enfoque holístico que integra aspectos técnicos, sociales y ambientales, y se subrayan las oportunidades para futuras mejoras e innovaciones. Este TFG no solo proporciona una solución técnica eficaz, sino que también contribuye al desarrollo sostenible y al empoderamiento de comunidades rurales en África.

Palabras clave: Monitoreo, Energía solar, Bombeo de agua, HTTP, RS232, RS485, MODBUS TCP/RTU, Node.js, Subsistema, Planta Híbrida

Abstract

This Final Degree Project (TFG) addresses the design, development and implementation of a hybrid photovoltaic and water pumping solar plant in an African village (Zimbabwe). This project has as its main objective to provide a sustainable solution for the supply of drinking water and electricity in disadvantaged rural communities through the integration of renewable energies. Throughout the work, a careful selection of subsystems, energy storage system and monitoring software has been made, all aimed at maximizing efficiency and minimizing costs.

Initially, a demo plant was developed in a controlled environment, which made it possible to identify and solve possible faults before the final installation. The start-up in the African village was a success, ensuring a constant supply of water and significantly improving the quality of life of the inhabitants. In addition, the project has proven to be replicable and scalable, which has motivated the planning of new facilities in other communities in the region.

The project's findings highlight the importance of a holistic approach that integrates technical, social and environmental aspects, and highlight opportunities for future improvements and innovations. This TFG not only provides an effective technical solution, but also contributes to the sustainable development and empowerment of rural communities in Africa.

Key words: Monitoring, Solar Energy, Water Pumping, HTTP, RS232, RS485, MODBUS TCP/RTU, Node.js, Subsystem, Hybrid Plant

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En un mundo donde la energía y el agua son recursos fundamentales para el desarrollo y bienestar de las comunidades, muchas regiones remotas, especialmente en África, enfrentan desafíos significativos para acceder a estas necesidades básicas. La escasez de agua potable y la falta de acceso a fuentes de energía confiables son obstáculos cruciales para el progreso económico y social en estas áreas. Sin embargo, en la búsqueda de soluciones sostenibles y eficientes, las plantas solares híbridas de bombeo de agua y fotovoltaicas han surgido como una prometedora respuesta a este dilema.

Este Trabajo de Fin de Grado se centra en implantar un sistema de monitorización y control en una planta solar híbrida de bombeo de agua y fotovoltaica, con el propósito de llevar agua y electricidad a una pequeña aldea africana. Esta iniciativa representa un esfuerzo importante para abordar las crecientes preocupaciones ambientales y las necesidades humanas básicas en comunidades marginadas. La combinación de la energía solar fotovoltaica y los sistemas de bombeo de agua proporciona una solución versátil y sostenible que puede mejorar significativamente la calidad de vida de las personas en estas áreas, al tiempo que reduce la dependencia de fuentes de energía no renovable y contribuye a la mitigación del cambio climático.

En este contexto, este trabajo se propone investigar y analizar la implementación, operación y monitorización de una planta solar híbrida de bombeo de agua y fotovoltaica en Zimbabue. A través de un enfoque multidisciplinario que abarca la ingeniería eléctrica, la gestión energética, la tecnología de paneles solares y la gestión de recursos hídricos, se busca comprender cómo estas plantas pueden ser diseñadas, operadas y mantenidas de manera eficiente, y cómo sus beneficios pueden transformar la vida de las comunidades locales, ya que esta misma planta se implantará en otras regiones del continente africano.

Este estudio se presenta en el contexto más amplio de la transición hacia una energía más limpia y sostenible, y en la lucha contra la desigualdad en el acceso a servicios básicos esenciales. La exploración de esta solución híbrida no solo tiene el potencial de abordar cuestiones cruciales de desarrollo, sino que también representa un paso importante hacia un futuro más equitativo y sostenible para las poblaciones africanas y, en última instancia, para el mundo entero.

1.1 Motivación

La motivación detrás de la elección de este Trabajo de Fin de Grado se basa en la convicción de que la tecnología y la innovación pueden ser catalizadores para el cambio positivo en las comunidades más desfavorecidas del mundo. La necesidad urgente de abordar la escasez de agua potable y la falta de acceso a la electricidad en muchas

aldeas africanas es un llamado a la acción. Además, el compromiso con la sostenibilidad ambiental y la reducción de la dependencia de fuentes de energía no renovable impulsa nuestro interés en las plantas solares híbridas. Al fusionar la pasión por la ingeniería y la tecnología con un deseo genuino de mejorar la calidad de vida de las personas, este trabajo representa un esfuerzo concreto para contribuir a un mundo más justo y equitativo, al tiempo que se abordan los desafíos globales relacionados con el acceso a recursos esenciales y la mitigación del cambio climático.

1.2 Objetivos

Con el propósito de alcanzar un funcionamiento óptimo y una gestión efectiva de la planta híbrida de bombeo de agua y energía fotovoltaica, se emprendió un ambicioso proyecto. Este proyecto tuvo como objetivo central el diseño e implementación de un completo sistema de monitoreo en tiempo real. Esto incluye desde la generación de energía eléctrica y el proceso de bombeo de agua, hasta el manejo y diagnóstico de los posibles fallos del sistema. Para lograr este objetivo, se siguieron una serie de pasos fundamentales que se detallan a continuación:

1. **Selección de Dispositivos de Medición:** El primer objetivo es realizar un análisis exhaustivo de las variables críticas a monitorear en la planta solar híbrida. Esto incluye factores como la radiación solar, el nivel de agua, la temperatura ambiente, entre otros. Luego, se procederá a la identificación y selección de los dispositivos de medición más adecuados para capturar con precisión estas variables. Esta etapa es fundamental para garantizar la efectividad y precisión de la monitorización durante todas las fases del proyecto, permitiendo establecer una base sólida para el desarrollo de esta, asegurando que los sensores y sistemas de adquisición de datos sean seleccionados de manera estratégica para cumplir con las necesidades específicas de la planta solar híbrida.

2. **Programación y Desarrollo de Dispositivos de Monitorización:** El segundo objetivo es diseñar y desarrollar los dispositivos de monitorización necesarios para la planta solar híbrida de bombeo de agua y fotovoltaica. Esto incluye la programación de sensores, sistemas de adquisición de datos y la interfaz de usuario para supervisar el rendimiento de la planta en tiempo real.

3. **Puesta en Marcha de la Planta Solar:** El tercer objetivo consiste en llevar a cabo la puesta en marcha de la planta solar híbrida en la pequeña aldea africana. Esto implica la instalación física de los componentes del sistema, la conexión a la red eléctrica local y la verificación de su funcionamiento adecuado.

4. **Monitorización Continua y Evaluación de Rendimiento:** Una vez que la planta esté en funcionamiento, el cuarto objetivo es llevar a cabo una monitorización continua de su rendimiento. Esto implica la recopilación y análisis de datos en tiempo real para garantizar la eficiencia operativa y la optimización de la producción de agua y electricidad.

5. **Identificación y Resolución de Problemas:** En caso de que surjan problemas operativos o de rendimiento durante la monitorización, el quinto objetivo es identificar y resolver estos problemas de manera eficiente. Esto podría implicar ajustes en el sistema, mantenimiento preventivo o reparaciones necesarias.

6. **Informe y Documentación:** Finalmente, se espera que este trabajo genere un informe completo que documente todo el proceso, desde el diseño de los dispositivos de

monitorización hasta la puesta en marcha y la monitorización continua de la planta. Este informe proporcionará una guía valiosa para futuros proyectos similares y contribuirá al conocimiento en el campo de la energía renovable y la gestión de recursos hídricos en áreas rurales de África.

En la siguiente ilustración 1.1, se presenta un diagrama de bloques, el cual proporciona una visión general de los elementos claves y su interrelación en el desarrollo del sistema de monitoreo de la planta híbrida.

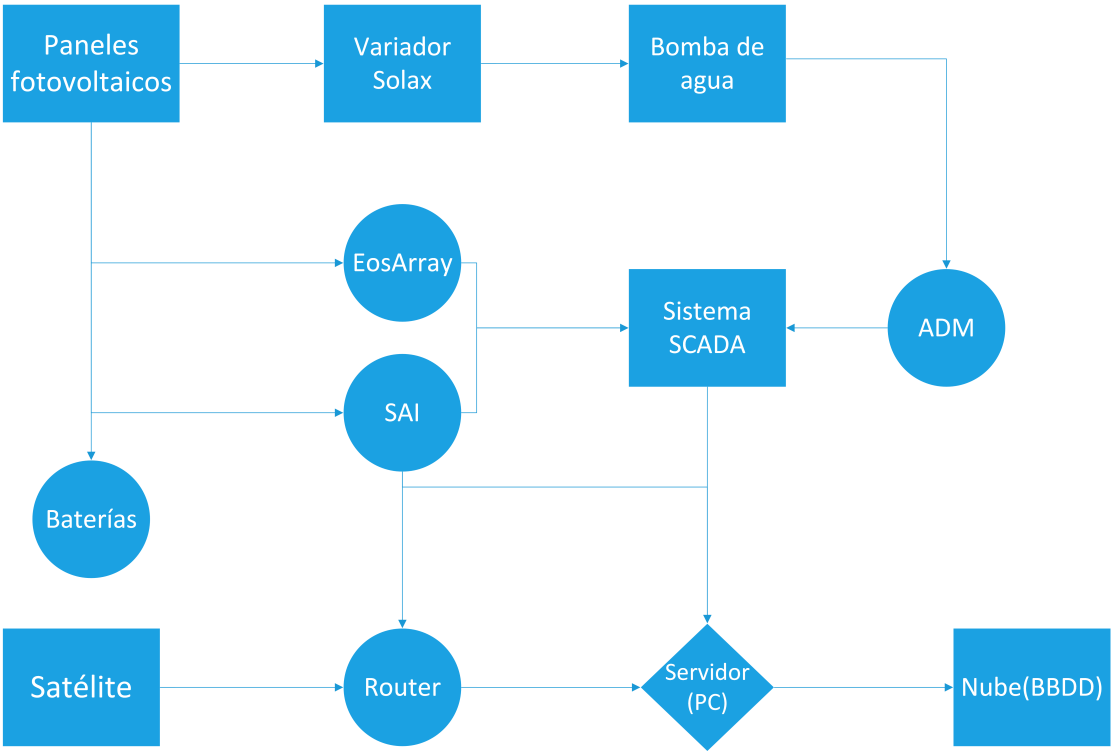


Figura 1.1: Diagrama de bloques de la planta

El resultado de este TFG no solo mejoró la eficiencia y la operatividad de la planta solar híbrida, sino que también ha tenido un impacto directo en la comunidad al proporcionar un acceso constante y confiable a dos recursos vitales: *luz* y *agua*. Además, el proyecto sirve como modelo replicable para otras comunidades en situaciones similares, demostrando cómo la tecnología solar puede ser utilizada para superar las barreras de infraestructura en áreas remotas.

Este trabajo encapsula la integración de soluciones tecnológicas avanzadas en un contexto social y ambiental desafiante, reflejando un compromiso con la innovación y la sostenibilidad en el ámbito de la ingeniería.

1.3 Colaboraciones

Este proyecto ha sido realizado por el departamento de electrónica de la Universidad politécnica de Valencia y financiado por una ONG, llamada IOM, en el cual yo he formado parte como becario, encargándome de la parte software en la creación de la planta híbrida de bombeo de agua y fotovoltaica.

IOM(Organización Internacional para las Migraciones) es una ONG cuya labor consiste en ofrecer servicios y asesoramiento a gobiernos y migrantes, cerciorarse de una ges-

tión ordenada y humana de la migración; promover la cooperación internacional sobre cuestiones migratorias; ayudar a encontrar soluciones prácticas a los problemas migratorios: y ofrecer asistencia humanitaria a los migrantes que lo necesitan, ya se trate de refugiados, de personas desplazadas o desarraigadas. La Constitución de IOM reconoce explícitamente el vínculo entre la migración y el desarrollo económico, social y cultural, así como el respeto del derecho a la libertad de movimiento de las personas.

La función de este proyecto es dotar de red eléctrica y de agua potable a una pequeña aldea africana de Zimbabue, para poder mejorar el estilo de vida y las condiciones adversas en las que viven los habitantes de esta aldea remota del continente africano.

Este proyecto ha sido desarrollado por cuatro miembros, donde cada uno hemos tenido una función específica y crucial en el desarrollo del mismo. A continuación, se describen las funciones y contribuciones de cada miembro del equipo:

- **Integrante A: Desarrollo del Frontend**
  - Responsable de crear la interfaz de usuario de la aplicación de monitorización de la planta.
  - Diseño y desarrollo web para la visualización de datos y la interacción del usuario.
  - Implementado mediante Angular, usando tecnologías como HTML, CSS y JavaScript para asegurar una experiencia de usuario óptima.
- **Integrante B: Desarrollo del Predictor de Limpieza de Placas Solares mediante IA**
  - Encargado de desarrollar un modelo de inteligencia artificial para predecir el estado de limpieza de las placas solares.
  - Recolectó y procesó los datos necesarios para entrenar el modelo.
  - Utilizó técnicas de machine learning para desarrollar y evaluar el predictor, asegurando su precisión y fiabilidad.
- **Miembro Supervisor: Encargado de la Parte Electrónica**
  - Supervisó y dirigió el proyecto, proporcionando orientación técnica y estratégica.
  - Responsable del diseño e implementación de la parte electrónica del proyecto.
  - Aseguró que todos los componentes electrónicos funcionaran correctamente y se integraran con el sistema global.
  - Encargado de la puesta en marcha final en África.
- **Mi Contribución: Desarrollo del Backend y Selección de Dispositivos**
  - Probé y seleccioné los dispositivos necesarios para la monitorización, asegurando su compatibilidad y eficiencia.
  - Encargado de desarrollar toda la parte del backend de la aplicación de monitorización de la planta.
  - Diseñé y escribí el código necesario para la recolección, almacenamiento y procesamiento de los datos provenientes de la planta.

En resumen, cada miembro del equipo aportó su experiencia y habilidades específicas para completar diferentes aspectos del proyecto, logrando un producto final cohesivo y funcional. Mi contribución principal fue en el área del backend, donde me aseguré de que los datos de la planta fueran recopilados, procesados y almacenados de manera eficiente, para poder llevar a cabo una monitorización precisa y completa de esta.

1.4 Estructura de la memoria

- **Capítulo 1 (Introducción):** Breve descripción del proyecto, contexto, motivación, objetivos y organización del documento.
- **Capítulo 2 (Estado de la cuestión):** Teoría acerca de las plantas solares híbridas, desafíos que presenta y solución desarrollada.
- **Capítulo 3 (Plan de trabajo):** Diagrama de Gantt con la cronología y las tareas realizadas con una breve descripción de que consiste cada una.
- **Capítulo 4 (Elementos del sistema de monitoreo):** Dispositivos empleados en la monitorización de la planta.
- **Capítulo 5 (Modos de comunicación):** Tipo de comunicación empleada para conectar todos los subsistemas de la planta.
- **Capítulo 6 (Desarrollo del software):** Creación del software de cada dispositivo y tecnologías utilizadas en su desarrollo.
- **Capítulo 7 (Diseño y desarrollo de la solución empleada):** Diseño final desarrollado y explicación del mismo.
- **Capítulo 6 (Pruebas y resultados):** : Pruebas con los sensores, datos de consumos y de autonomía y análisis de resultados.
- **Capítulo 7 (Conclusiones):** Conclusiones obtenidas tras finalizar el proyecto.

CAPÍTULO 2
Estado de la cuestión

2.1 Planta Solar Híbrida: Fundamentos y aplicaciones

Una planta solar híbrida de bombeo de agua y fotovoltaica es un sistema innovador que combina la generación de energía solar fotovoltaica con una aplicación específica de bombeo de agua, todo integrado en una sola instalación. Este tipo de planta aprovecha la energía solar, una fuente limpia y renovable, para operar bombas de agua que pueden ser utilizadas en diversas aplicaciones, como la irrigación agrícola, el suministro de agua potable o la gestión de recursos hídricos.

La planta incluye paneles solares que convierten la luz solar directamente en electricidad. Esta electricidad puede ser utilizada para alimentar cualquier carga eléctrica, además de destinarla para el funcionamiento de las bombas de agua. Estas a su vez son accionadas por la electricidad generada por los paneles solares.

Este sistema elimina la necesidad de combustibles fósiles o electricidad de la red, lo que es especialmente útil en áreas remotas o en regiones donde el suministro eléctrico es inestable o demasiado costoso. Lo que distingue a estas plantas de otros sistemas es su capacidad para operar de manera híbrida. Esto puede incluir el uso de baterías para almacenar energía solar excedente para asegurar un suministro constante y confiable.

Este sistema consta de los siguientes elementos:

- **Paneles Solares:** Capturan la energía del sol y la convierten en electricidad.
- **Inversores:** Convierten la corriente continua (DC) de los paneles en corriente alterna (AC) para alimentar bombas de AC, o directamente a bombas de DC si así se requiere.
- **Bombas de Agua:** dispositivos mecánicos diseñados para mover agua de un lugar a otro mediante la conversión de energía eléctrica en energía mecánica.
- **Controladores de Bombas:** Optimizan el rendimiento de la bomba según la disponibilidad de energía solar, pudiendo ajustar la velocidad de la bomba y otras variables.
- **Sistema de Almacenamiento de Energía:** Almacenan energía solar para usar en períodos de baja irradiación solar o durante la noche.
- **Sensores y Sistemas de Monitoreo:** Equipos para monitorear el flujo de agua, la presión, el nivel de agua en los depósitos y la irradiación solar, entre otros. Estos datos permiten ajustar operativamente el sistema para optimizar el uso del agua y la energía.

Las principales ventajas que ofrece una planta híbrida de bombeo de agua y fotovoltaica son:

- **Sostenibilidad:** Utiliza energía solar, reduciendo la dependencia de combustibles fósiles y disminuyendo las emisiones de carbono.
- **Costo-Efectividad:** A largo plazo, reduce los costos operativos al minimizar el gasto en energía convencional.
- **Flexibilidad:** Capaz de operar en áreas aisladas sin necesidad de infraestructura eléctrica convencional.
- **Escalabilidad:** Puede adaptarse a diferentes escalas, desde pequeñas instalaciones rurales hasta grandes operaciones agrícolas.

Estos sistemas representan una solución eficiente para la gestión de recursos hídricos y energéticos, proporcionando una herramienta valiosa para el desarrollo sostenible en comunidades en vías de desarrollo.

2.2 Problemas y desafíos de las plantas solares híbridas

Las plantas solares híbridas enfrentan varios desafíos que pueden afectar a su implementación y operación.

- **Costo inicial:** La inversión inicial para una planta solar híbrida puede ser significativa, especialmente por los costos de los paneles solares, el sistema de bombeo y las baterías. El retorno de la inversión depende de varios factores, como la disponibilidad de sol, el costo de alternativas energéticas y las tarifas de agua. La ventaja que disponemos en este proyecto es que todo está financiado a través de la ONG *IOM(Organización Internacional para las Migraciones)*.
- **Mantenimiento:** Los sistemas solares requieren mantenimiento regular para asegurar su eficiencia. El polvo, la suciedad y otros residuos pueden reducir la capacidad de los paneles solares para generar energía. Las bombas y otros equipos mecánicos también necesitan mantenimiento regular para prevenir fallos. Por lo que es importante disponer de los mejores dispositivos para que la planta perdure, además de un buen sistema de mantenimiento periódico para mantener la eficiencia de la planta durante todo su ciclo de vida.
- **Eficiencia energética:** Maximizar la eficiencia energética en la conversión de energía solar en electricidad y en el bombeo de agua es crucial. La eficiencia puede verse afectada por la calidad de los componentes utilizados, la instalación y la integración del sistema.
- **Variabilidad solar:** La producción de energía solar es intermitente y depende de las condiciones climáticas, lo que puede ser un desafío para garantizar un suministro constante de agua y electricidad, especialmente en áreas con alta variabilidad meteorológica, por lo que es crucial disponer de un buen sistema de almacenamiento energético.
- **Acceso a tecnología y experiencia:** En áreas remotas o menos desarrolladas, el acceso a la tecnología avanzada y a la experiencia técnica necesaria para instalar y mantener estos sistemas puede ser limitado.

- **Integración y gestión del sistema:** Integrar eficazmente la generación de energía y el bombeo de agua en un sistema que reaccione dinámicamente a las variaciones en la demanda de agua y la disponibilidad de energía solar requiere una gestión sofisticada y a menudo una automatización avanzada.

Este TFG se ha basado en abordar el problema descrito en el último punto, integración y gestión del sistema, el cual consiste en desarrollar e instalar un sistema de monitorización y optimización capaz de controlar toda la instalación de la planta híbrida de forma remota. Para ello es necesario disponer de un software avanzado de control para poder manipular y controlar toda la instalación en tiempo real y hacer uso de los dispositivos más robustos y eficaces, ya que la planta va a ser instalada en un lugar remoto del continente africano donde la infraestructura es inexistente. Con esto conseguiremos abastecer de luz y agua a la pequeña aldea de *Zimbabue*, lo que mejorará el día a día de muchas personas, haciéndoles la vida más fácil.

En conclusión, las plantas solares híbridas de bombeo de agua y fotovoltaica representan una solución innovadora y sostenible para la gestión de recursos en áreas donde la conexión a la red eléctrica es inexistente o poco fiable. A pesar de sus desafíos, ofrecen un considerable potencial para mejorar la eficiencia del uso de recursos naturales, reduciendo la dependencia de combustibles fósiles y minimizando el impacto ambiental, a la vez que mejoras las condiciones de vida de personas desfavorecidas en lugares remotos del mundo. Por lo que mi misión ha sido conseguir crear un software capaz de abarcar este problema.

2.3 Estado del arte

Este proyecto se basa en el diseño, implementación y monitoreo de una planta híbrida fotovoltaica y de bombeo de agua en una pequeña aldea africana. El objetivo principal es proporcionar energía eléctrica y agua potable a la comunidad, mejorando su calidad de vida y fomentando el desarrollo sostenible. Para ello, vamos a diseñar un sistema híbrido, el cual mediante la energía solar recogida en paneles solares pueda proporcionar energía eléctrica y proporcionar abastecimiento de agua gracias al proceso de bombeo de la misma. A su vez, se implementará un sistema de monitoreo en tiempo real para gestionar y optimizar el rendimiento de la planta.

Ahora vamos a introducir aplicaciones de mercado similares a nuestra elección y los motivos por los que nos hemos decantado por este proyecto para nuestro TFG:

- **Planta Solar Híbrida en Nigeria (2023):** La planta solar híbrida en Nigeria, desarrollada en 2023, se enfoca en proveer energía y agua a una comunidad rural. Utiliza paneles solares y un sistema de almacenamiento en baterías de iones de litio para garantizar un suministro continuo de energía eléctrica y una bomba de agua solar para el abastecimiento de agua. Este proyecto ha logrado resultados significativos, como la provisión constante de energía y agua durante todo el año, reduciendo la dependencia de fuentes de energía no renovables. Además, ha mejorado la producción agrícola gracias a un riego confiable y ha elevado la calidad de vida de los residentes al facilitar el acceso a agua potable y energía eléctrica. La planta no solo mejora la calidad de vida de los residentes al proporcionar una fuente de energía confiable y sostenible, sino que también reduce significativamente las emisiones de CO2.

- **Sistema Híbrido Solar-Diésel en Mali (2022):** El sistema híbrido solar-diésel en Mali, implementado en 2022, combina paneles fotovoltaicos con generadores diésel de respaldo para asegurar un suministro constante de agua mediante bombas solares. Este enfoque ha demostrado ser eficaz en la reducción de costos operativos, gracias al uso predominante de energía solar. La presencia de generadores diésel garantiza un suministro ininterrumpido de agua, especialmente en períodos de baja irradiación solar. Además, el proyecto ha promovido la capacitación y el empleo de técnicos locales, mejorando la autosuficiencia y la sostenibilidad del sistema. El impacto socioeconómico ha sido notable, con mejoras en la salud y la economía local debido a un suministro fiable y económico de agua.
- **Planta Solar de Pumpmakers en Namibia (2022):** En Namibia, se ha implementado una planta solar que destaca por su capacidad de almacenamiento de energía mediante baterías, lo que permite un suministro constante incluso durante la noche o en días nublados. Esta planta no solo abastece de energía a la comunidad local, sino que también impulsa actividades económicas como la agricultura y pequeños comercios. La integración de sistemas de almacenamiento en proyectos solares es una tendencia creciente, ya que mejora la estabilidad y confiabilidad del suministro eléctrico.

Proyecto	Eficiencia Energética	Costo	Impacto Ambiental	Sostenibilidad	Fiabilidad
Nigeria	Alta (solar + baterías)	Moderado (alto costo inicial)	Bajo (energía renovable)	Alta (bajo mantenimiento)	Alta (almacenamiento en baterías)
Mali	Moderada (solar + diésel)	Moderado (respaldo diésel)	Moderado (uso de diésel)	Moderada (depende del diésel)	Alta (respaldo diésel)
Namibia	Alta (solar puro)	Bajo (menos componentes)	Bajo (energía renovable)	Alta (bajo mantenimiento)	Moderada (solo energía solar)

Tabla 2.1: Tabla comparativa proyectos

El camino elegido para este TFG se basa en la integración de un sistema híbrido fotovoltaico con capacidad de bombeo de agua, inspirado en el éxito de proyectos similares. La elección de este enfoque responde a varias razones:

- **Sostenibilidad y Reducción de Emisiones:** Siguiendo el ejemplo de la planta de Nigeria, se busca implementar una solución que no solo proporcione energía, sino que también contribuya a la reducción de emisiones de CO2.
- **Fiabilidad y Continuidad del Suministro:** La combinación de fuentes de energía y la incorporación de almacenamiento, como se ha visto en Namibia, garantiza un suministro eléctrico constante, vital para el desarrollo económico y social de la comunidad.
- **Adaptabilidad y Escalabilidad:** Al igual que el sistema en Malí, la capacidad de adaptarse a diferentes condiciones y la posibilidad de escalar el proyecto para atender a más usuarios en el futuro son aspectos clave que hacen de este enfoque una opción viable y eficiente.

Estos proyectos demuestran la viabilidad y el impacto positivo de las plantas híbridas en comunidades rurales, proporcionando una base sólida y justificada para la implementación de este proyecto en la aldea africana de Zimbabue.

2.4 Solución desarrollada

En el contexto de la monitorización y optimización de una planta solar híbrida de bombeo de agua y fotovoltaica, se ha desarrollado una solución que aborda tanto los aspectos tecnológicos como operativos de la planta. La solución está diseñada para garantizar una operación eficiente, confiable y sostenible, proporcionando energía y agua a una comunidad rural en África. A continuación, se presenta una descripción general de las funcionalidades clave de la solución implementada.

1. **Sistema de Monitorización y Adquisición de Datos:** La base de la solución es crear un sistema de monitorización robusto que recopile y analice datos en tiempo real, a través de sensores para medir los parámetros claves de la planta, que a su vez están conectados a un módulo de adquisición de datos proporcionando una interfaz confiable y precisa para la recopilación y gestión de estos.
2. **Comunicación y Conectividad:** Para asegurar una supervisión y control eficiente, se implementó un sistema de comunicación robusto que asegura una conexión continua a Internet mediante tecnología satelital. Esta configuración permite una comunicación fiable y estable con la planta, además de con los propios dispositivos entre ellos, haciendo uso de la comunicación *RS485* lo que facilita el monitoreo remoto y el control de las operaciones desde cualquier ubicación.

La solución desarrollada no solo asegura un suministro constante y confiable de energía y agua, sino que también mejora la sostenibilidad y eficiencia de la planta. Los principales beneficios incluyen:

- **Acceso Ininterrumpido a Recursos Vitales:** La comunidad rural tiene acceso continuo a electricidad y agua, mejorando significativamente su calidad de vida.
- **Sostenibilidad y Autosuficiencia:** La planta solar híbrida contribuye a la sostenibilidad ambiental y promueve la autosuficiencia de la comunidad al reducir la dependencia de fuentes de energía no renovables.
- **Reducción de Costos Operativos:** La optimización del uso de la energía ayuda a reducir los costos operativos y a extender la vida útil de los equipos.
- **Algoritmos de Diagnóstico** Implementé algoritmos para el diagnóstico de fallos, basados en los datos en tiempo real. Esto ayuda a prevenir problemas antes de que ocurran y a mantener la planta operando a su máxima eficiencia.

En resumen, la solución desarrollada integra tecnologías avanzadas y prácticas de gestión eficientes para crear una planta solar híbrida que no solo satisface las necesidades energéticas y de agua de la comunidad, sino que también sirve como un modelo replicable para otras regiones con desafíos similares.

CAPÍTULO 3
Plan de trabajo

En este apartado vamos a introducir y explicar todas las tareas realizadas durante las prácticas de empresa. Estas se van a reflejar en un diagrama de Gantt, mostrado en la siguiente figura 3.1.

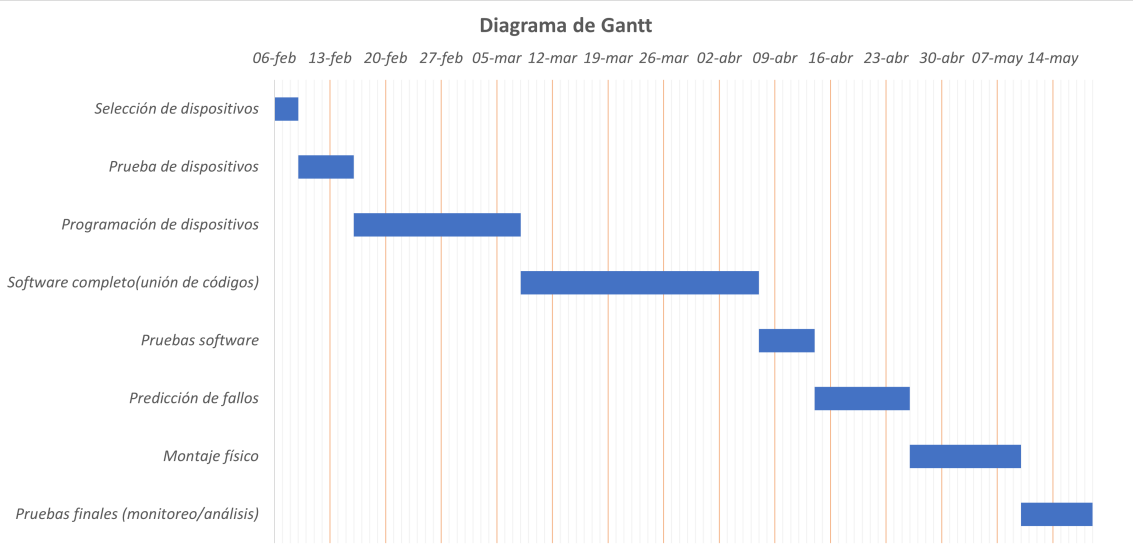


Figura 3.1: Tareas y cronología

Como podemos observar en la figura 3.1, como primera tarea tuvimos que seleccionar los dispositivos necesarios para llevar a cabo la monitorización completa de la planta, por lo que estuvimos buscando los periféricos que mejor se adaptaran a nuestro caso de uso. La elección se basaba en elegir dispositivos robustos y capaces de gestionar la planta de forma remota, ya que toda la monitorización se va a llevar a cabo desde España y se necesita una buena infraestructura para que todo funcione correctamente y de la manera más eficiente posible, intentando reducir los fallos a lo mínimo posible.

En segundo lugar, me centré en probar cada uno de los dispositivos seleccionados para asegurar que cumplen con los requisitos necesarios para el proyecto. Esto incluye la verificación de funcionalidades y compatibilidades.

A continuación, tras disponer de todos los dispositivos compatibles con la planta, me centré en crear un software para cada periférico capaz de controlar y de mostrar los parámetros claves en tiempo real según la función del mismo. Para ello, tuvimos que crear una pequeña infraestructura en miniatura de prueba para disponer de datos y poder testear cada dispositivo, acercándonos en la medida de lo posible al escenario real.

Tras disponer de un software particular para cada dispositivo y comprobar que cada uno cumplía con su función específica, era hora de pasar a la unión de estos. Una vez programados los dispositivos, se procede a la integración de todos los códigos de software desarrollados en un único sistema completo. Esto requiere pruebas y ajustes para garantizar la compatibilidad y el correcto funcionamiento del software, que será el encargado de gestionar la planta al completo. Esto me llevó mucho tiempo, ya que cada dispositivo disponía de parámetros propios, al tratarse de dispositivos de distintas entidades, y al juntarlos tuve que ajustar cada parte del código para poder crear un software definitivo capaz de integrar cada dispositivo siendo un único sistema.

Una vez disponemos del código final, llegamos a la fase donde se llevan a cabo pruebas exhaustivas del software integrado para identificar y corregir errores. Para ello, conectando todos los dispositivos previamente a nuestra infraestructura de prueba, lanzamos el software y lo dejamos corriendo durante una semana ininterrumpidamente para comprobar que todos los parámetros leídos y el código funcionan correctamente. Es una fase crucial para asegurar la calidad del software antes de avanzar a la predicción de fallos.

Tras disponer del software completo y acabar con la fase más sustancial del proyecto, ahora tocaba darle un punto más de eficiencia al sistema, mediante la creación de un sistema de predicción de fallos, para reducir posibles fallos en el sistema y ayudarnos a prevenir problemas futuros. Este se integró dentro del código para asegurar un sistema aún más robusto. Una vez finalizada toda la parte software del proyecto, como fase final quedaba el ensamblado de todos los componentes físicos del sistema según el diseño planteado. Esto incluye la instalación de hardware y la conexión de todos los dispositivos programados y probados. Este punto se entraba en el montaje completo de la planta, tal y como se iba a instalar en Zimbabue, por lo que era muy importante ser meticulosos y conectar e instalar todo correctamente, ya que era el montaje de prueba definitivo antes de trasladar la planta a su lugar de destino.

Por último, tras tener todo el proyecto en marcha, nos quedaba la etapa final del proyecto, que consiste en realizar pruebas completas del sistema montado, mediante un monitoreo detallado y un análisis exhaustivo para asegurar que todos los componentes funcionan correctamente y que el sistema cumple con los objetivos del proyecto. Cualquier problema identificado en esta fase se corrige antes de la finalización del proyecto, ya que es decisivo a la hora de trasladar el proyecto a Zimbabue.

En resumen, este diagrama de Gantt muestra una planificación bien estructurada para llevar a cabo un proyecto de integración de dispositivos y software. Cada fase está claramente definida con fechas de inicio y fin, lo que permite una gestión efectiva del tiempo y de los recursos. Al seguir este cronograma, se asegura que todas las tareas críticas se completen de manera ordenada y se minimiza el riesgo de retrasos y errores en el proyecto final.

CAPÍTULO 4
Elementos del sistema de monitoreo

En el desarrollo de la planta solar híbrida de bombeo de agua y fotovoltaica, la precisión y confiabilidad en la recopilación de datos son elementos cruciales para garantizar un rendimiento óptimo y sostenible. Este capítulo se centra en la fase preliminar e instrumental de este proyecto: *la selección de los dispositivos de medición*. Antes de sumergirnos en la programación y desarrollo de los sistemas de monitorización, es imperativo establecer una base sólida mediante la identificación cuidadosa de los dispositivos que capturarán las variables clave del sistema. Desde la radiación solar hasta el nivel de agua, cada elemento seleccionado desempeñará un papel vital en la eficiencia operativa y la supervisión en tiempo real de la planta. Este proceso de selección no solo determinará la calidad de los datos recopilados, sino que también sentará las bases para el éxito continuo de la planta solar en la aldea africana, asegurando una gestión eficaz de los recursos energéticos e hídricos en la región.

4.1 Dispositivo EOS-ARRAY

Tal y como se describe en la referencia [1], **Eos Array** es un innovador sistema de monitorización para instalaciones fotovoltaicas, desarrollado por *CARLO GAVAZZI*. Esta solución domótica para monitorización de plantas fotovoltaicas, destaca por ser un sistema modular para el control local de estas, permitiendo la configuración de un máximo de 16 módulos.
Al ser modular, este puede estar compuesto por distintos módulos, y en nuestro caso hemos empleado los siguientes:

- Módulo**VMU-M:** unidad maestra y registradora, encargada del control de todo el dispositivo, que gestiona el bus local de las unidades de medición VMU-S y VMU-P, asignando automáticamente la dirección de la unidad local y recogiendo todas las mediciones locales que proceden de estas.
- Módulo**VMU-S:** unidad de medición de variables con portafusible de protección incorporado adecuado para la medición de intensidad CC, tensión, potencia y energía en aplicaciones solares fotovoltaicas. Las entradas/salidas de intensidad y las entradas de tensión facilitan las conexiones de los string. Además, la unidad está provista de un bus de comunicación auxiliar. Las alarmas, la detección de fusible fundido, la conexión de paneles fotovoltaicos y la comunicación se gestionan por medio del módulo VMU-M.

- Módulo **VMU-P:** Unidad de medición de las variables ambientales, adecuado para medir la temperatura del panel fotovoltaico, la temperatura del aire, la irradiancia solar y la velocidad del viento en aplicaciones solares fotovoltaicas. Además, la unidad está provista de un bus de comunicación que se gestiona por medio del módulo adicional VMU-M.

Como podemos observar en la figura 4.1, tenemos la representación gráfica de nuestro dispositivo con sus módulos correspondientes empleados en la planta.

Figura 4.1: Ejemplo dispositivo Eos-Array empleado en la planta

Al bombear de forma directa necesitamos el eos para extraer los valores, no nos basta únicamente con el inversor Solax. Además, dispone de un software gratuito.

4.2 Dispositivos Smart Power(Shelly)

Tal y como se describe en la referencia [2], los dispositivos Shelly son dispositivos de automatización diseñados para proporcionar una amplia gama de funciones que mejoran la comodidad, la eficiencia energética y la seguridad del espacio de trabajo donde se instalan. Estos ofrecen soluciones inteligentes para controlar diversos dispositivos eléctricos y sistemas y han sido diseñados para ser compactos, versátiles y fáciles de instalar.

Entre muchas de sus funciones vamos a destacar las que más nos conciernen respecto al uso en nuestra planta:

- **Control Remoto y Monitorización en Tiempo Real:** Los dispositivos *Shelly* permiten a los operadores de plantas solares híbridas controlar y monitorizar los dispositivos eléctricos de la planta desde cualquier ubicación a través de una conexión a Internet. Esto significa que pueden supervisar la generación de energía, el estado de los inversores, la carga de las baterías y otros parámetros importantes en tiempo real, lo que facilita la toma de decisiones rápidas y eficientes.
- **Programación y Optimización de Horarios de Operación:** La capacidad de programar horarios para activar o desactivar dispositivos de la planta solar, como los inversores o los sistemas de almacenamiento de energía, es crucial para optimizar el rendimiento y maximizar la eficiencia energética. Los dispositivos *Shelly* permiten a los operadores crear horarios personalizados que se adapten a las condiciones específicas de la planta y a las necesidades de energía de los usuarios.
- **Integración con Sistemas de Gestión de Energía:** Los dispositivos *Shelly* pueden integrarse fácilmente con sistemas de gestión de energía (EMS) y sistemas de control de supervisión (SCADA), lo que permite una monitorización y control más avanzados de la planta solar híbrida. Esta integración proporciona una visión holística de la operación de la planta y permite a los operadores tomar decisiones informadas para optimizar el rendimiento y minimizar los costos operativos.
- **Monitorización del Consumo y la Producción de Energía:** Los dispositivos *Shelly* pueden proporcionar datos detallados sobre el consumo y la producción de energía en la planta solar híbrida. Esto incluye información sobre la energía generada por los paneles solares, la energía almacenada en las baterías y el consumo de energía de la carga conectada a la planta. Esta monitorización detallada permite a los operadores identificar áreas de mejora y optimización para maximizar la eficiencia operativa y reducir los costos.
- **Escalabilidad y Flexibilidad:** Los dispositivos *Shelly* son altamente escalables y pueden adaptarse a plantas solares híbridas de cualquier tamaño, desde instalaciones pequeñas y descentralizadas hasta grandes parques solares. Además, su flexibilidad en términos de integración con diferentes sistemas de gestión de energía y protocolos de comunicación los hace ideales para su implementación en una variedad de entornos industriales.

En nuestro caso, vamos a hacer uso y a integrar los siguientes dispositivos Shelly a nuestra planta, con la finalidad de que actúen como switches, para activar o desactivar los dispositivos conectados, según nos convenga:

- **Dispositivo Shelly 1PM:** Interruptor inteligente de montaje en carril DIN con contactos libres de potencial, se puede acceder, controlar y monitorear de forma remota desde cualquier lugar. En la siguiente ilustración 4.2, podemos observar un ejemplo del dispositivo *Shelly 1PM* y en la ilustración 4.3, sus correspondientes puertos de conexión.

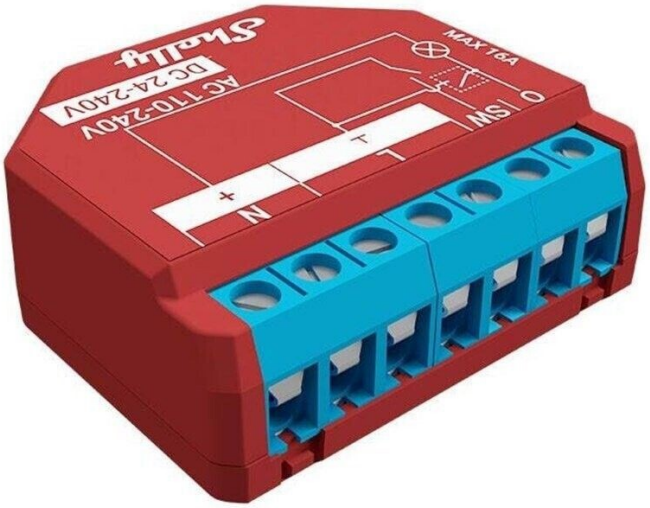


Figura 4.2: Ejemplo dispositivo Shelly 1PM

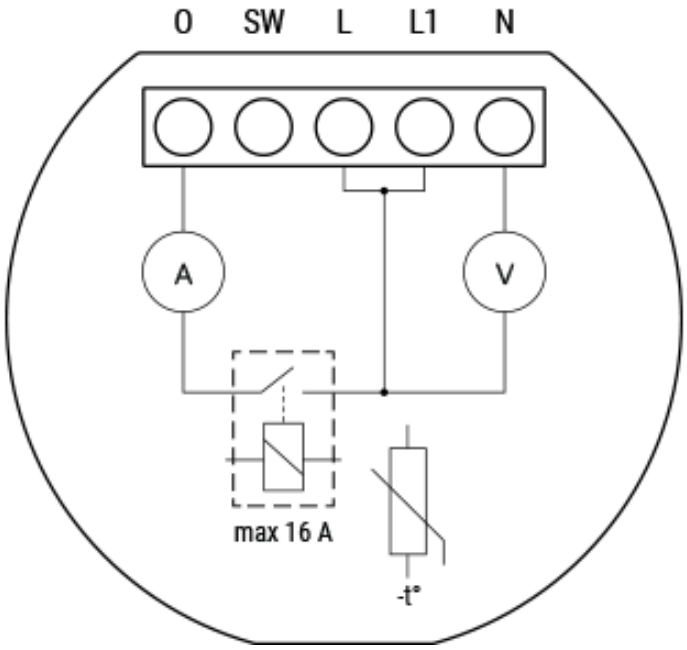


Figura 4.3: Puertos de conexión del dispositivo Shelly 1PM

- **Dispositivo Shelly Pro 3:** Interruptor inteligente de 3 canales con contactos libres de potencial, montable en carril DIN, capaz de conmutar 3 circuitos completamente independientes, incluidos los trifásicos. En la siguiente ilustración 4.4, podemos observar un ejemplo del dispositivo *Shelly 3 Pro* y en la ilustración 4.5, sus correspondientes puertos de conexión.

Figura 4.4: Ejemplo dispositivo Shelly Pro 3 empleado en la planta

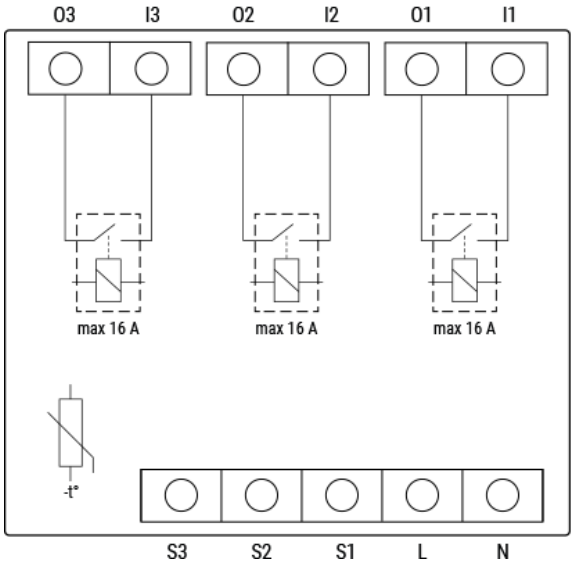


Figura 4.5: Puertos de conexión de dispositivo Shelly Pro 3

Ambos dispositivos disponen de una interfaz web integrada que se puede utilizar para monitorear y controlar el dispositivo, así como para ajustar su configuración.

En resumen, los dispositivos *Shelly* ofrecen una solución completa y versátil para la monitorización y el control de plantas solares híbridas, permitiendo a los operadores optimizar el rendimiento, maximizar la eficiencia energética y garantizar un suministro de energía fiable y sostenible.

4.3 Relay Board

Una relay board, o placa de relés, como podemos observar en la siguiente imágen 4.6, es un componente esencial en sistemas de control. Su función principal es actuar como un interruptor, permitiendo el control de dispositivos.

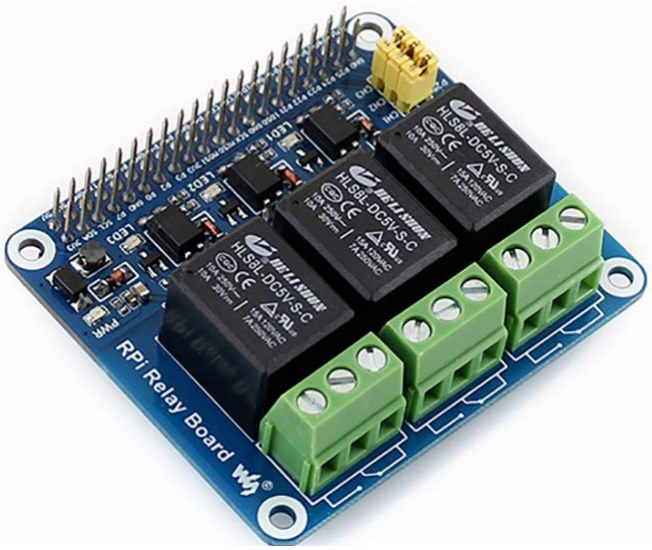


Figura 4.6: Placa de relés

A continuación se detallan las funciones específicas de una relay board en una planta híbrida fotovoltaica y de bombeo de agua:

- **Control de la Conmutación de Fuentes de Energía:** En una planta híbrida fotovoltaica, puede haber varias fuentes de energía. Esta permite la conmutación entre estas fuentes de energía de manera automática o manual según la disponibilidad y las necesidades de la planta.
- **Gestión de la Energía de los Paneles Solares:** La relay board puede ser utilizada para conectar y desconectar los paneles solares del sistema según la disponibilidad de luz solar y la demanda de energía. Esto ayuda a optimizar la generación y el uso de energía, evitando sobrecargas o desperdicios.
- **Control de las Bombas de Agua:** En un sistema de bombeo de agua, la relay board controla el funcionamiento de las bombas. Esto incluye encender y apagar las bombas según las necesidades de riego, niveles de agua en tanques, y otros parámetros operativos.
- **Automatización del Sistema:** Las relay boards pueden ser integradas con sistemas de control y monitoreo para automatizar operaciones, tales como:
  - Activación/desactivación de bombas basándose en niveles de agua sensados por sensores.
  - Cambio entre modos de operación.
  - Implementación de secuencias de encendido/apagado para minimizar el desgaste del equipo.

- **Protección y Seguridad:** Las relay boards también pueden servir como mecanismos de protección. Por ejemplo, pueden ser programadas para desconectar componentes en caso de sobrecarga, cortocircuito, o fallos, protegiendo así tanto el equipo como a los operarios.
- **Control Remoto y Supervisión:** Con la incorporación de tecnologías de comunicación, las relay boards permiten el control y supervisión remotos del sistema. Esto es especialmente útil para plantas situadas en ubicaciones remotas o de difícil acceso.

En resumen, una relay board es crucial para el control eficiente, seguro y automatizado de la planta. Las funciones que ofrecen son muy similares a la de los dispositivos Shelly, mientras que las relay boards ofrecen una solución robusta y flexible para el control de sistemas en plantas híbridas, los dispositivos Shelly proporcionan una opción más accesible y fácil de usar con capacidades avanzadas de monitoreo y control remoto, por lo que la elección entre ambos dependerá de las necesidades específicas del sistema.

4.4 Dispositivos ADAM

Tal y como se describe en la referencia [3], los dispositivos **ADAM**(Advanced Data Acquisition Modules) son una serie de módulos de adquisición de datos desarrollados por Advantech, diseñados para recopilar datos de sensores y dispositivos en entornos industriales.

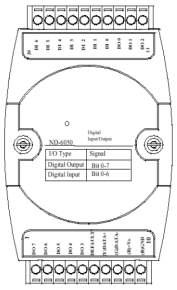
Pueden ser empleados en sistemas hidráulicos para adquirir datos de sensores como presión, flujo, temperatura y nivel de líquido, entre otros parámetros relevantes. Estos datos pueden ser luego utilizados para monitorear el rendimiento del sistema, realizar diagnósticos, controlar procesos, y llevar a cabo tareas de mantenimiento predictivo.

En nuestro caso vamos a emplear los dispositivos: **ADM-4280-C** y **ADAM-4017+**. Podemos observar un ejemplo del dispositivo destinado a la adquisición de datos relacionados con la energía solar en la ilustración siguiente 4.7 y un ejemplo del dispositivo destinado como módulo de control del bombeo de agua, en la ilustración 4.8.

Figura 4.7: Ejemplo dispositivo ADAM 4017+ empleado en la planta

En la siguiente ilustración 4.9, podemos observar el esquema de conexión de los dispositivos ADAM empleados en nuestra planta.

Figura 4.8: Ejemplo dispositivo ADM-4280-C empleado en la planta



Pin #	Signal Name	Description
1	DO 7	Digital output channel 7
2	DO 6	Digital output channel 6
3	DO 5	Digital output channel 5
4	DO 4	Digital output channel 4
5	DO 3	Digital output channel 3
6	Default*	Initial state setting
7	(Y) DATA+	RS-485 series signal, positive
8	(G) DATA-	RS-485 series signal, negative
9	(R) +Vs	Power supply, +10V~+30V
10	(B) GND	Ground
11	DO 2	Digital output channel 2
12	DO 1	Digital output channel 1
13	DO 0	Digital output channel 0
14	DI 0	Digital input channel 0
15	DI 1	Digital input channel 1
16	DI 2	Digital input channel 2
17	DI 3	Digital input channel 3
18	DI 4	Digital input channel 4
19	DI 5	Digital input channel 5
20	DI 6	Digital input channel 6

Figura 4.9: Entradas y salidas digitales de los dispositivos ADAM

La elección de estos dispositivos ADAM se debe a sus siguientes características:

4.4.1. ADAM-4017+

El ADM-4017+ es un módulo de adquisición de datos diseñado para aplicaciones industriales de control y monitoreo. En el contexto de una planta solar de bombeo de agua y fotovoltaica, el ADM-4017+ puede desempeñar varias funciones clave:

1. **Monitoreo de Parámetros Fotovoltaicos:**
   - **Voltaje y Corriente:** El módulo puede medir y monitorear el voltaje y la corriente de los paneles solares y los sistemas de bombeo.
   - **Temperatura:** Monitoreo de la temperatura de los paneles solares para evaluar el rendimiento y detectar posibles sobrecalentamientos.
   - **Radiación Solar:** Medición de la radiación solar para optimizar el rendimiento de los paneles fotovoltaicos.
2. **Control del Sistema de Bombeo:**
   - **Niveles de Agua:** Monitoreo de los niveles de agua en los tanques de almacenamiento y en los pozos para controlar las operaciones de bombeo.
   - **Presión:** Supervisión de la presión en las tuberías de agua para asegurar un funcionamiento eficiente del sistema de bombeo.

3. **Integración con Sistemas SCADA:** El ADM-4017+ puede integrarse con sistemas SCADA (Supervisory Control and Data Acquisition) para proporcionar datos en tiempo real y permitir el control remoto del sistema.

Además de disponer de una serie de características que lo hacen desempeñar un papel fundamental en nuestra planta:

- **Entradas analógicas:** El ADM-4017+ cuentan con ocho canales de entrada analógica, lo que permite la conexión de múltiples sensores o dispositivos de campo para la adquisición de señales analógicas.
- **Rango de Entrada:** El módulo ADM-4017+ admite una amplia gama de voltajes de entrada, lo que lo hace compatible con una variedad de sensores y dispositivos industriales. Dependiendo de la configuración, puede admitir rangos de entrada desde ±150 mV hasta ±10 V, lo que permite la conexión de diferentes tipos de sensores.
- **Comunicación en red:** Estos dispositivos ADAM se comunican con un sistema de control o una computadora a través de una interfaz de comunicación estándar, como RS-485 o RS-232. La comunicación se realiza utilizando protocolos de comunicación comunes, como Modbus RTU, lo que facilita la integración con sistemas de control más amplios y con softwares de supervisión.
- **Montaje y Conexión:** se puede montar en riel DIN para una fácil instalación en gabinetes eléctricos o armarios de control. Los terminales de conexión desmontables facilitan la conexión de cables y sensores.
- **Robustez y fiabilidad:** Los dispositivos ADAM están diseñados para funcionar en entornos industriales adversos, por lo que suelen tener una construcción robusta y resistente a condiciones ambientales severas, lo que los hace adecuados para aplicaciones hidráulicas en entornos exigentes, como puede ser una aldea remota en *Zimbabue*.

En el caso de nuestra planta, este dispositivo se va a utilizar específicamente para la parte de fotovoltaica.

En resumen, el ADM-4017+ es una herramienta crucial para el monitoreo y control eficiente de una planta solar de bombeo de agua y fotovoltaica, proporcionando datos precisos y en tiempo real que permiten optimizar el rendimiento y asegurar la operatividad continua del sistema.

4.4.2. ADM-4280-C

El ADM-4280-C es un transceptor RS-485 con aislamiento digital y protección contra sobretensiones. En una planta solar de bombeo de agua y fotovoltaica, este dispositivo puede cumplir varias funciones importantes para asegurar la comunicación y protección del sistema. A continuación, se detallan las principales funciones que podría desempeñar en dicho entorno:

1. **Comunicación de Datos:**
    - **Control y Monitoreo de Bombas:** En una planta solar de bombeo de agua, es crucial controlar y monitorear el funcionamiento de las bombas de agua y los sensores asociados, como los sensores de nivel de agua, presión y flujo. El

ADM-4280-C permite la comunicación fiable entre estos dispositivos utilizando el protocolo RS-485, que es robusto y adecuado para largas distancias.

- **Monitoreo de Paneles Solares:** En la sección fotovoltaica de la planta, el ADM-4280-C puede facilitar la comunicación entre los inversores solares y el sistema de monitoreo central, asegurando que el rendimiento de los paneles solares se pueda supervisar y controlar de manera efectiva.

2. **Aislamiento y Protección:**

   - **Aislamiento Galvánico:** El ADM-4280-C proporciona aislamiento galvánico, lo que permite separar físicamente las señales eléctricas entre diferentes partes del sistema. Esto protege los equipos sensibles de variaciones de voltaje y picos de corriente, que son comunes en entornos solares.
   - **Protección contra Sobretensiones:** La protección contra sobretensiones incorporada en el ADM-4280-C protege los componentes electrónicos de daños causados por picos de voltaje, que pueden ocurrir debido a condiciones climáticas (como rayos) o fallos del sistema.

3. **Interoperabilidad y Flexibilidad:**

   - **Compatibilidad con Otros Equipos:** Al utilizar el protocolo RS-485, el ADM-4280-C puede interoperar con una amplia gama de dispositivos industriales. Esto facilita la integración de diferentes componentes del sistema de monitoreo.

En nuestro caso, este dispositivo se centrará únicamente en la parte de bombeo de agua de la planta, así cada módulo estará encargado de una sección en nuestra planta híbrida, uno encargado de la parte de energía y el otro encargado de la parte hídrica.

En resumen, el dispositivo ADM-4280-C es crucial en una planta solar de bombeo de agua y fotovoltaica para asegurar una comunicación fiable y protegida entre los distintos componentes del sistema, mejorando la eficiencia y la fiabilidad del mismo. Su capacidad de aislamiento galvánico y protección contra sobretensiones hace que sea un componente esencial en entornos remotos donde la robustez y la integridad de los datos son vitales para un funcionamiento eficiente de la planta.

4.5 Variador de frecuencia

En nuestro caso vamos a emplear un inversor **Solax X1**, que será el encargado de controlar la velocidad del motor ajustando la frecuencia de salida de la planta.

Tal como se muestra en la referencia [4], el **Inversor SolaX X1** es un inversor monofásico de conexión a red. Por ello, debe funcionar necesariamente con paneles solares y con la red eléctrica. Incorporan dos reguladores MPPT independientes con una eficiencia superior al 99 %, además de incorporar también protección IP65 y diversos protocolos de comunicaciones, tanto para poder monitorizar su funcionamiento como para comunicarse con el vatímetro de medición de potencia.

Aquí hay algunos aspectos clave sobre los variadores de frecuencia Solax:

- **Funcionamiento:** Los variadores de frecuencia Solax funcionan convirtiendo la energía de corriente alterna (CA) suministrada por la red eléctrica en corriente continua (CC) mediante un rectificador, y luego nuevamente en corriente alterna de frecuencia variable mediante un inversor.

- **Control de velocidad:** Los variadores de frecuencia Solax permiten controlar la velocidad de motores eléctricos de manera precisa y eficiente. Al ajustar la frecuencia de salida, es posible variar la velocidad del motor según los requisitos de la aplicación.
- **Aplicaciones:** Estos variadores de frecuencia se utilizan en una amplia gama de aplicaciones industriales y comerciales donde se requiere control de velocidad, como son los sistemas de bombeo.
- **Características:** Como principal característica nos permiten interconectarnos con la red eléctrica sin necesidad de baterías, es por ello que este tipo de instalaciones son económicas y muy rentables a largo plazo, dado su bajo coste de adquisición, nulo mantenimiento y una sencilla puesta en marcha. Además, también ofrecen protección contra sobrecargas, control de par, modos de operación programables, interfaces de usuario intuitivas, comunicación en red para integración con sistemas de control, y monitoreo de parámetros de operación.
- **Eficiencia energética:** Al controlar la velocidad del motor según las necesidades de carga, los variadores de frecuencia Solax pueden ayudar a mejorar la eficiencia energética y reducir el consumo de energía en comparación con los métodos de control tradicionales, como las válvulas de estrangulamiento o los arrancadores directos.
- **Compatibilidad con energía solar:** Dado el enfoque de Solax Power en soluciones de energía renovable, es posible que algunos de sus variadores de frecuencia estén diseñados para integrarse con sistemas de energía solar fotovoltaica, permitiendo un control más inteligente y eficiente de los motores en aplicaciones donde se utiliza energía solar.

En la siguiente imagen 4.10, podemos observar las conexiones del dispositivo a emplear en nuestra planta:

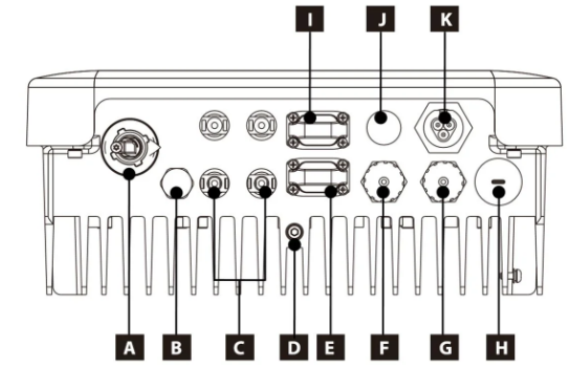


Figura 4.10: Conexiones del inversor Solax

4.6 Sai

Tal y como se describe en la referencia [5], el sistema de alimentación ininterrumpida (SAI) es un dispositivo de alimentación que puede utilizarse para proporcionar energía de reserva temporal a los dispositivos electrónicos. De esta forma, cuando se produce una interrupción en el suministro eléctrico, el SAI cambia automáticamente a la energía de la batería y suministra a los dispositivos suficiente energía hasta que se restablece el suministro.

A diferencia de las fuentes de alimentación tradicionales, que proporcionan energía instantánea cuando se encienden, un SAI vierte continuamente su electricidad almacenada en el dispositivo que está protegiendo en caso de fallo de la red eléctrica.

En definitiva, en caso de que se vaya la luz, el SAI sirve para mantener los dispositivos que tengamos conectado a él con energía suficiente como para seguir usándolo durante corto espacio de tiempo, guardar todo lo necesario y apagarlo de forma segura.

En nuestro caso, como dispositivo de emergencia ante caídas de la red eléctrica, vamos a emplear los dispositivos In-Line **Salicru**.

Este tipo de SAI Interactivo o SAI In-Line destaca por ofrecer una protección eléctrica intermedia, donde su funcionamiento se basa en tecnología similar a la de los Sais Off-Line, pero con la diferencia de que el Sai In-Line si incorpora una serie de filtros activos, como un microprocesador, que controla las fluctuaciones de la red eléctrica regulando la tensión de salida.

En la siguiente ilustración, 4.11 podemos observar un ejemplo de nuestro dispositivo sai empleado en la planta.

Figura 4.11: Ejemplo dispositivo Sai

4.7 Power Banks

En el continente africano, las zonas rurales a menudo enfrentan desafíos significativos en cuanto al acceso a la electricidad. Para muchas comunidades, la red eléctrica puede ser limitada o incluso inexistente, dejando a los residentes en un estado de desconexión energética. Esta falta de acceso a la energía plantea numerosos obstáculos en la vida cotidiana, desde la imposibilidad de cargar teléfonos móviles hasta la incapacidad de utilizar dispositivos esenciales para la comunicación, el trabajo y la seguridad.

En este contexto, las power banks emergen como una solución portátil y accesible para abordar estas necesidades energéticas en áreas remotas de África. Estos dispositivos compactos y versátiles tienen el potencial de transformar la vida de las comunidades al proporcionar una fuente de energía confiable y móvil para una variedad de dispositivos electrónicos.

Tal y como se describe en la referencia [6], las power banks, también conocidas como baterías externas o cargadores portátiles, son dispositivos diseñados para almacenar energía eléctrica y suministrarla a dispositivos electrónicos y ofrecen diversas soluciones para mejorar el estilo de vida de personas en lugares subdesarrollados. Estas soluciones son:

- **Acceso a la Energía en Áreas Remotas:** En muchas zonas rurales de África, el acceso a la electricidad puede ser limitado o inexistente. Las power banks pueden proporcionar una solución portátil y accesible para cargar dispositivos electrónicos, lo que permite a los residentes de la aldea mantener sus teléfonos móviles, linternas solares u otros dispositivos funcionando incluso cuando no hay acceso a la red eléctrica.
- **Carga de Dispositivos Esenciales:** Las power banks pueden utilizarse para cargar dispositivos esenciales para la vida diaria, como teléfonos móviles. En muchas comunidades, los teléfonos móviles son herramientas vitales para acceder a servicios de salud, comunicarse con familiares y amigos, o incluso para actividades comerciales. Mantener estos dispositivos cargados puede marcar la diferencia en situaciones de emergencia o para mantener la conectividad con el mundo exterior.
- **Apoyo en Casos de Emergencia:** En situaciones de emergencia, como cortes de energía prolongados o desastres naturales, las power banks pueden ser fundamentales para mantener la comunicación y la seguridad. Por ejemplo, pueden utilizarse para cargar radios portátiles, linternas recargables o dispositivos médicos básicos que requieran alimentación eléctrica.
- **Promoción de la Actividad Económica:** Al proporcionar acceso a la energía para cargar dispositivos móviles, las power banks pueden facilitar actividades económicas como el comercio electrónico, la banca móvil y la comunicación comercial. Esto puede impulsar el desarrollo económico local al permitir que los residentes participen en la economía digital global.

En resumen, las power banks pueden desempeñar un papel significativo en mejorar la calidad de vida y promover el desarrollo en áreas rurales de África al proporcionar acceso a la energía eléctrica de manera portátil y accesible. Su versatilidad y capacidad para cargar una variedad de dispositivos electrónicos los convierten en herramientas valiosas para mejorar la conectividad y el bienestar de las comunidades locales.

En la siguiente imagen 4.12 podemos observar un ejemplo de power bank llevada a la aldea para proveer a sus habitantes de energía eléctrica.

Figura 4.12: Ejemplo PowerBank

4.8 Router

En el contexto de la implementación de una planta solar híbrida de bombeo de agua y fotovoltaica en una aldea remota de África, la conectividad a Internet juega un papel crucial en el funcionamiento eficiente y la supervisión efectiva de la planta. Sin embargo, en áreas donde la infraestructura de red tradicional es limitada o inexistente, como es común en muchas zonas rurales, el acceso a la conectividad puede ser un desafío significativo.

Para abordar esta necesidad vital de conectividad, se ha optado por la implementación de un router industrial, diseñado específicamente para entornos exigentes y condiciones adversas. Este router no solo actuará como el centro neurálgico de la red dentro de la planta, sino que también será la puerta de acceso a Internet, estableciendo una conexión vital a través de tecnología satelital.

El objetivo principal de este router industrial es proporcionar una conexión a Internet confiable y estable, permitiendo así la supervisión remota, el monitoreo en tiempo real y la gestión eficiente de la planta desde cualquier lugar del mundo. En un entorno donde la comunicación instantánea y la toma de decisiones basada en datos son esenciales para el funcionamiento óptimo de la planta, la disponibilidad de una conexión a Internet robusta se vuelve indispensable.

Este router industrial no solo garantizará la conectividad dentro de la planta, sino que también permitirá la transmisión de datos críticos relacionados con el rendimiento de los paneles solares, el bombeo de agua y otros aspectos operativos de la planta. Además, facilitará la comunicación para asegurar una respuesta rápida y eficaz ante cualquier eventualidad.

En el desafiante entorno de África, especialmente en áreas remotas y rurales, la infraestructura de comunicaciones enfrenta una serie de desafíos únicos que van desde condiciones climáticas extremas hasta una infraestructura subdesarrollada. En este contexto, la implementación de un router industrial no solo como un puente hacia la conectividad

a Internet, sino también como una infraestructura resistente capaz de soportar las hostilidades del entorno africano, es crucial para el éxito operativo de la planta solar híbrida.

El router industrial seleccionado para este propósito está diseñado para funcionar en condiciones adversas y entornos hostiles. Estas son algunas de las características que lo hacen apto para resistir los desafíos específicos de África:

- **Robustez y Resistencia:** Este router industrial está construido con materiales robustos y duraderos, capaces de resistir condiciones climáticas extremas como altas temperaturas, humedad, polvo y vibraciones. Su diseño robusto garantiza un funcionamiento confiable incluso en entornos hostiles.
- **Amplio Rango de Temperatura:** Diseñado para soportar fluctuaciones extremas de temperatura, desde las altas temperaturas del día hasta las bajas temperaturas nocturnas, este router industrial garantiza un rendimiento estable y confiable en cualquier condición climática.
- **Protección contra Polvo y Agua:** Equipado con certificaciones de protección IP (Ingress Protection), este router está sellado herméticamente para protegerlo contra la intrusión de polvo y la entrada de agua, lo que lo hace apto para entornos polvorientos y húmedos.
- **Resistencia a Impactos y Vibraciones:** Diseñado para soportar golpes, vibraciones y condiciones adversas comunes en entornos industriales y de campo, este router industrial garantiza un funcionamiento ininterrumpido incluso en situaciones de alto impacto.
- **Estabilidad de la Conexión:** Además de su resistencia física, este router industrial está equipado con tecnología avanzada para garantizar una conexión estable y confiable en todo momento. Esto es crucial para mantener la conectividad vital en áreas donde la infraestructura de red puede ser irregular o limitada.

En resumen, la implementación de un router industrial para proporcionar conexión a Internet a través de tecnología satelital representa un paso crucial en la operación exitosa y la supervisión efectiva de una planta solar híbrida en una ubicación remota, como es *Zimbabue*. Esta infraestructura de red robusta y confiable no solo garantiza la conectividad vital para la planta, sino que también sienta las bases para un funcionamiento eficiente y una gestión efectiva a largo plazo.

En la siguiente imagen, 4.13, podemos observar un ejemplo del router industrial empleado en la planta para controlar toda la infraestructura de red.

4.9 PC Industrial

En el corazón de la infraestructura de la planta solar híbrida se encuentra un componente vital: un PC industrial con sistema operativo Windows, diseñado específicamente para controlar y supervisar todos los aspectos operativos de la planta. Este PC industrial no solo actúa como el cerebro central de la planta, sino que también desempeña un papel crucial en la gestión eficiente de los dispositivos y en la recopilación y análisis de datos en tiempo real.

Aquí hay algunas características y funciones clave de este PC industrial y cómo contribuye a la operación efectiva de la planta:

Figura 4.13: Ejemplo Router Inhand empleado en la planta

- **Robustez y Confiabilidad:** El PC industrial seleccionado está diseñado para soportar las condiciones rigurosas y exigentes del entorno. Construido con componentes de alta calidad y un diseño robusto, este PC industrial garantiza un funcionamiento confiable y continuo incluso en entornos con temperaturas extremas y polvo, como es la pequeña aldea africana de Zimbabue.
- **Sistema Operativo Windows:** La elección del sistema operativo Windows proporciona una plataforma familiar y fácil de usar para el personal de la planta. Windows ofrece una interfaz intuitiva y una amplia compatibilidad con software de supervisión y control, lo que facilita la implementación de soluciones personalizadas para las necesidades específicas de la planta.
- **Control Centralizado de Dispositivos:** Como centro de la infraestructura, este PC industrial es responsable de controlar y supervisar todos los dispositivos de la planta, incluidos los inversores solares, las bombas de agua, los sensores y otros equipos. Utilizando software especializado y protocolos de comunicación estándar, el PC industrial coordina las operaciones de los dispositivos para garantizar un funcionamiento eficiente y coordinado de toda la planta.
- **Recopilación y Análisis de Datos:** El PC industrial recopila datos en tiempo real de todos los sensores y dispositivos de la planta, proporcionando información valiosa sobre el rendimiento operativo, la generación de energía, el consumo de agua y otros parámetros clave. Estos datos se utilizan para realizar un seguimiento del rendimiento de la planta, identificar posibles problemas y tomar decisiones informadas para optimizar la eficiencia y la productividad.
- **Supervisión Remota:** El PC industrial permite la supervisión remota de la planta a través de una conexión a Internet. Esto permite al personal de operaciones monitorear y gestionar la planta desde cualquier lugar, lo que facilita una respuesta rápida a cualquier problema o emergencia que pueda surgir.

En resumen, el PC desempeña un papel fundamental en la operación eficiente y confiable de la planta solar híbrida. Como centro de la infraestructura, este PC industrial controla y supervisa todos los dispositivos, recopila y analiza datos en tiempo real, y facilita la supervisión remota de la planta, garantizando así un funcionamiento óptimo y una gestión efectiva de los recursos en todo momento.

CAPÍTULO 5
Comunicaciones

Tal y como se describe en la referencia [7], RS-485 es una interfaz estándar de la capa física de comunicación, un método de transmisión de señales, el primer nivel del modelo OSI (Interconexión de Sistemas Abiertos). Este fue creado para ampliar las capacidades físicas de la interfaz RS-232. La conexión serie RS485 es realizada utilizando un cable de tres hilos: un hilo de datos, un hilo con datos invertidos y un hilo cero (tierra, 0 V). De este modo, los transmisores y los receptores intercambian los datos a través de un cable de par trenzado de hilos rígidos.

Su principal función es transportar una señal a través de dos cables. Uno de los cables transmite la señal original y el otro transporta su copia inversa. Este método de transmisión ofrece una gran resistencia a las interferencias en modo común.

Cuando la línea de comunicación RS485 está preparada para funcionar a nivel físico, hay que pensar en el protocolo de transferencia de datos: es decir, es necesario que los dispositivos del sistema establezcan un formato común para la transmisión de los paquetes de datos.

En el protocolo de comunicación RS485, los comandos son enviados por el nodo establecido como maestro. Todos los demás nodos conectados al maestro reciben los datos a través de los puertos RS485. Dependiendo de la información enviada, ninguno o varios nodos de la línea responden al maestro.

La tecnología RS-485 sigue siendo la base de muchas redes de comunicación. Las principales ventajas de la interfaz RS-485 son:

- Intercambio de datos bidireccional a través de un par de hilos trenzados.
- Admite varios transceptores conectados a la misma línea, es decir, permite crear una red.
- Gran longitud de la línea de comunicación.
- Alta velocidad de transmisión.

La comunicación RS485 es una tecnología ampliamente utilizada en aplicaciones industriales y de automatización debido a sus características robustas y fiables. En el contexto de una planta solar híbrida de bombeo de agua y fotovoltaica, la implementación de la comunicación RS485 para conectar los dispositivos empleados en la planta ofrece numerosos beneficios. A continuación, se presentan los principales beneficios de utilizar RS485 en esta aplicación:

- **Robustez y Fiabilidad:** La comunicación RS485 puede transmitirse a distancias de hasta 1200 metros sin necesidad de repetidores, lo que es particularmente útil en instalaciones extensas como una planta solar híbrida.
- **Capacidad Multidrop:** Soporta conexión de múltiples dispositivos en una sola línea de comunicación , permitiendo que varios sensores, controladores y otros dispositivos de la planta se comuniquen en el mismo bus. Esto simplifica el cableado y reduce los costos de instalación. Ofrece topología en Bus, donde todos los dispositivos pueden comunicarse con un solo controlador central (PC industrial) o entre sí.
- **Velocidad y Flexibilidad:** RS485 permite configurar diversas tasas de baudios, adaptándose a las necesidades específicas de la planta en términos de velocidad y volumen de datos. Además de ofrecer flexibilidad en la configuración de la comunicación, facilitando la integración de diferentes tipos de dispositivos y sistemas en la planta, desde sensores hasta actuadores y sistemas de control.
- **Costos Reducidos:** La capacidad de conectar múltiples dispositivos en una sola línea reduce la cantidad de cableado necesario, lo que disminuye los costos materiales y de instalación. La estandarización de RS485 permite el uso de equipos y componentes ampliamente disponibles y económicos, lo que reduce los costos operativos y de mantenimiento.
- **Integración y Escalabilidad:** RS485 es compatible con muchos protocolos de comunicación industrial estándar, como Modbus RTU, facilitando la integración de una amplia gama de dispositivos y sistemas de control. La capacidad de añadir fácilmente más dispositivos al bus de comunicación RS485 permite la escalabilidad del sistema, adaptándose a futuras expansiones o modificaciones en la planta sin necesidad de una reestructuración significativa de la infraestructura de comunicación.
- **Consumo de Energía Eficiente:** Los dispositivos que utilizan RS485 tienden a ser eficientes en términos de consumo de energía, lo que es beneficioso en una planta solar donde la gestión eficiente de la energía es crucial.

En resumen, la utilización de la comunicación RS485 en la planta solar híbrida ofrece una solución robusta, fiable y económica para la interconexión de dispositivos, facilitando una operación eficiente y coordinada de la planta. Su capacidad para resistir condiciones adversas, junto con su flexibilidad y escalabilidad, lo convierte en una opción ideal para garantizar la comunicación efectiva y la supervisión integral de la planta, en entornos remotos del continente africano.

5.1 Protocolo ModBus

Una de las principales características y que diferencia la comunicación RS485 de las otras comunicaciones serie es el formato de los datos intercambiados. Mientras que los dispositivos RS232 se conectan a través de protocolos de texto (ASCII), la mayoría de los dispositivos RS485 utilizan Modbus.

Modbus es un protocolo de comunicaciones serie muy utilizado en los dispositivos electrónicos industriales. En Modbus, la conexión se establece entre un maestro (host) y los esclavos (dispositivos basados en COM). Modbus permite acceder a la configuración de los dispositivos y leer las mediciones.

Para describir las comunicaciones Modbus se hace uso del modelo Cliente-Servidor, caracterizado por la comunicación entre el dispositivo o dispositivos clientes, que inician la comunicación y realizan peticiones al dispositivo o dispositivos servidores, que procesan las peticiones y devuelven la respuesta adecuada (o el mensaje de error).

El intercambio de datos es iniciado por un servidor. El servidor puede conmutar su controlador RS-485 al modo de transmisión por sí mismo, mientras que los otros controladores RS-485 (esclavos) trabajan en el modo de recepción. Para que un esclavo responda al maestro a través de la línea de comunicación, el maestro debe enviarle un comando especial, que da al dispositivo en cuestión el derecho de cambiar su controlador al modo de transmisión durante un tiempo determinado.

Modbus es uno de los protocolos más sencillos para que los dispositivos interactúen entre sí. Es muy fácil de implementar para los fabricantes de equipos, lo cual es la principal razón de su prevalencia, pero al mismo tiempo es compleja para un ingeniero o un programador. Esta solución traslada todos los problemas de implementación a la solución final, ya que requiere que el ingeniero o programador trabaje con tablas de varias páginas de registros y variables, sus direcciones, varias funciones de escritura y lectura y la conversión de datos.

Uno de los puntos que más tiempo me llevó controlar fue precisamente la complejidad de la implementación de Modbus. Aunque es un protocolo ampliamente adoptado por su facilidad de integración para los fabricantes, la realidad es que requiere un esfuerzo significativo por parte del ingeniero. La necesidad de gestionar tablas extensas de registros y variables, determinar sus direcciones exactas, manejar las diferentes funciones de lectura y escritura, y realizar conversiones precisas de datos representa un desafío considerable. Este proceso implica una atención meticulosa a los detalles y una comprensión profunda de cómo se estructuran y comunican los datos en el sistema, lo cual me llevó bastante tiempo dominar completamente, ya que cada dispositivo disponía de una tabla de registros propia y cada uno se regía por los intereses propios del fabricante.

CAPÍTULO 6
Diseño y desarrollo

En este apartado se abordará el desarrollo del software específico para cada dispositivo escogido en la fase anterior, buscando extraer las máximas funcionalidades de los periféricos que posiblemente formen parte de la versión final del proyecto. Por lo que habrá funcionalidades y dispositivos que a la hora de juntar el software no nos sean útiles o sean redundantes, ya que hay dispositivos que son capaces de extraer datos similares y tenemos que escoger lo que mejor nos convenga en cuanto a la eficiencia final de la planta. Este análisis es fundamental para comprender no solo el propósito y la funcionalidad de cada script desarrollado, sino también las razones detrás de su implementación y la elección de tecnologías utilizadas. A través de ejemplos concretos, se ilustrará cómo cada pieza de software contribuye al objetivo general del proyecto y cómo las decisiones tecnológicas apoyan su eficiencia y eficacia.

Para ello haremos uso de **Node.js**. Tal y como se describe en la referencia [9], Node.js es un entorno *Javascript* del lado del servidor, basado en eventos, lo que proporciona una plataforma robusta y versátil para el desarrollo del backend. Su capacidad para manejar interactividad, asincronía y su amplio ecosistema de herramientas y bibliotecas lo convierten en la elección ideal para cumplir con los requisitos del proyecto y ofrecer una experiencia de usuario superior.

Esto significa que todas las tareas que se desarrollan por el servidor se hacen de forma paralela, por lo que pueden efectuarse de forma simultánea y sin que se produzca ningún tipo de bloqueo en el flujo de trabajo. El resultado es una ventaja competitiva considerable que proporciona a la arquitectura de las aplicaciones web una mayor potencia y velocidad de procesamiento, lo que mejora considerablemente la monitorización en tiempo real de nuestro proyecto.

6.1 Arquitectura General del Sistema

La arquitectura general del sistema está diseñada para integrar y gestionar de manera eficiente múltiples subsistemas y componentes, asegurando la generación, almacenamiento, distribución, y monitoreo de energía solar, así como el suministro de agua mediante bombeo. El sistema se organiza de manera distribuida, donde cada dispositivo desempeña un rol específico en la operación y supervisión del sistema. Para ello, en la siguiente imagen 6.1, se muestra el correspondiente diagrama de bloques asociado a esta sección, en el cual se ofrece una visión más clarificadora a cerca de la arquitectura general del diseño y desarrollo del software antes de su cohesión final. En este esquema, el *router* actúa como el nodo central de comunicación, interconectando todos los componentes, como los sensores, inversores y sistemas de almacenamiento de energía, a través del protocolo pertinente. Por otro lado, disponemos del PC(servidor), el cual se posiciona

como el núcleo operativo, recopilando y procesando los datos transmitidos por los diferentes dispositivos. Cada subsistema dispone de un script propio de lectura o escritura, mediante el cual se lanzan los procesos necesarios para la prueba de estos periféricos antes de crear el software general de monitoreo. Tras lanzar cada uno de los procesos, toda esta información es almacenada localmente por el servidor, cuya finalidad es analizar y comparar con los resultados esperados en la monitorización final de la planta. Esta estructura asegura una gestión eficiente y coordinada de todos los subsistemas. Además, la arquitectura del sistema está diseñada para ser modular y escalable, lo que permite la integración de nuevos dispositivos y subsistemas según sea necesario. Esta flexibilidad del sistema permite su adaptación a diferentes entornos y necesidades, facilitando la replicación de la planta en otras comunidades o la expansión del sistema en la comunidad actual.

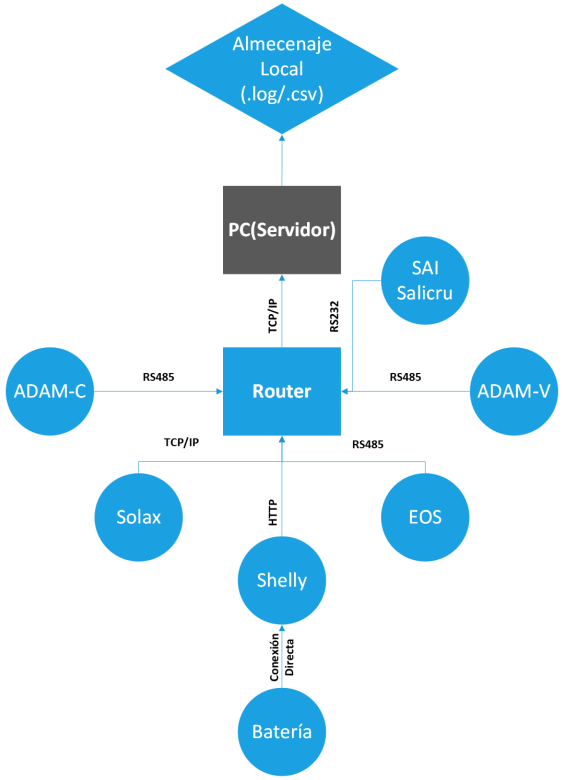


Figura 6.1: Esquema general de la arquitectura del sistema durante la fase de desarrollo

En la figura 6.1, se puede observar el esquema gráfico de la arquitectura general del sistema, el cual ilustra los componentes que componen esta primera parte de prueba de los subsistemas y la interacción entre ellos. Las siguientes secciones explican más en detalle cada una de las partes que intervienen en el diseño y desarrollo previo al montaje final del software completo de monitoreo.

6.2 Interacción y Comunicación entre Dispositivos

Cada subsistema está diseñado para interactuar de manera fluida con los demás, enviando y recibiendo mensajes a través de protocolos de comunicación estándar. A continuación, se detalla la interacción entre los principales dispositivos empleados en la planta:

- **Inversor SOLAX:** El inversor es responsable de convertir la corriente continua(CC) generada por los paneles solares en corriente alterna(CA) utilizable por los equipos de la planta. El inversor se comunica con el servidor, enviando datos sobre su estado operativo, eficiencia de conversión y cualquier fallo detectado. Además, puede recibir comandos para ajustar su funcionamiento según las necesidades del sistema. El inversor está en constante comunicación con el servidor. Recibe datos sobre la producción de energía solar y el consumo en tiempo real, permitiendo ajustar su operación para maximizar la eficiencia y minimizar las pérdidas. También puede recibir comandos del servidor para modificar su configuración según las necesidades del sistema.
- **Smart Power:** Los dispositivos Shelly son módulos de control inteligente que permiten gestionar el encendido y apagado de cargas conectadas, como las bombas de agua, sistemas de iluminación, o cualquier otro dispositivo eléctrico que forme parte de la planta. Están diseñados para automatizar la operación de estos dispositivos, siguiendo las órdenes del servidor basadas en las condiciones del sistema y las demandas energéticas. Estos dispositivos se comunican con el servidor mediante Wi-Fi o Ethernet, recibiendo instrucciones basadas en las condiciones del sistema y enviando confirmaciones de estado. Los Shelly también pueden recopilar datos sobre el consumo energético de las cargas conectadas, los cuales son enviados al servidor para su análisis.
- **Dispositivos ADAM:** Los módulos ADAM de voltaje y corriente son dispositivos de adquisición de datos esenciales, encargados de monitorear tanto los parámetros eléctricos como variables ambientales y operativas críticas para el sistema. El ADAM de voltaje está orientado a obtener datos relacionados con la energía solar, como la irradiancia solar, la temperatura ambiente o la presión atmosférica, mientras que el ADAM de corriente está vinculado al sistema de bombeo de agua, centrado en variables del sistema de bombeo, como son el nivel del tanque, el caudal de la bomba o la temperatura del agua. Los mensajes enviados por ambos ADAM incluyen tanto los valores medidos como alertas en caso de que se detecten condiciones fuera de lo normal, como sobrevoltaje, niveles bajos de agua, o presión inadecuada, permitiendo al servidor tomar decisiones rápidas y efectivas para mantener la estabilidad y eficiencia de la planta. Los datos recopilados se envían al servidor a través del router, utilizando protocolos de comunicación como Modbus. Esta información es crucial para monitorizar el rendimiento de la planta y detectar posibles anomalías en el suministro de energía.
- **Eos Array:** El Eos Array es un dispositivo dedicado a la gestión y optimización de la generación de energía solar a partir de los paneles fotovoltaicos. Su función principal es monitorear el rendimiento de los paneles, controlar su orientación y asegurar que la producción de energía sea lo más eficiente posible, ajustando automáticamente parámetros operativos como el ángulo de inclinación o la conexión en serie/paralelo de los paneles. Está estrechamente integrado con el inversor para ajustar la carga y asegura que el rendimiento del sistema esté alineado con las condiciones de la red y las necesidades de la planta. Envía información sobre la

producción de energía al servidor, como son la irradiancia, los vatios generados o la temperatura, entre otros parámetros clave.

- **SAI Salicru:** El Sistema de Alimentación Ininterrumpida(SAI) Salicru proporciona energía de respaldo en caso de interrupciones del suministro eléctrico, asegurando que los dispositivos críticos dela planta sigan operando sin interrupciones. Este sistema es vital para mantener la estabilidad y la continuidad operativa de la planta solar, especialmente en situaciones de emergencia. El SAI está directamente conectado al PC(servidor) y al router para enviar alertas sobre su estado y recibir comandos que ajusten su operación en función de las condiciones del sistema. En caso de fallo en la red eléctrica, el SAI notifica al servidor para que se inicien los protocolos de emergencia. Además, el SAI también reporta cierta información al servidor, como por ejemplo, su estado operativo o el SOC de batería.

- **Power Banks:** Las power banks se utilizan como fuente de energía de respaldo para dispositivos críticos en caso de fallos prolongados en el suministro de energía. En nuestro caso, como ese rol lo desempeña el SAI, estas serán utilizadas como fuentes de energía alternativas e independientes para proveer a la comunidad, como por ejemplo tener energía eléctrica de manera portátil, se emplearan principalmente para funciones específicas. Además, están conectadas al sistema de gestión energética y pueden enviar datos sobre su nivel de carga al servidor, que las gestiona para maximizar su duración durante emergencias.

- **Router:** El router actúa como el centro de comunicación de la planta, facilitando la conexión y el flujo de datos entre todos los dispositivos. Es esencial para garantizar que todos los subsistemas puedan interactuar de manera eficiente y sin interrupciones, manteniendo la sincronización entre ellos y el servidor. El router está conectado a todos los dispositivos de la planta, facilitando la transmisión de datos y comandos entre estos, asegurando que la comunicación sea rápida y confiable. El router en sí no recopila información sobre la operación de la planta, pero es responsable de garantizar que los datos enviados por otros dispositivos lleguen al destino correcto. Además, puede enviar mensajes de estado de la red, como disponibilidad de conexión, velocidad de transmisión, y latencia, que son críticos para la operación fluida del sistema.

- **PC Industrial(Servidor):** El servidor es el centro neurálgico del sistema, donde se recopilan, almacenan y procesan todos los datos. Aquí, se ejecutan scripts de control que analizan la información recibida de los diferentes dispositivos, toman decisiones automatizadas, y envían comandos de vuelta a los subsistemas. El servidor se comunica con todos los dispositivos de la planta a través del router. Recibe datos de todos los componentes del sistema, y en base a esta información, envía comandos para ajustar la operación de estos. También puede interactuar con el operador humano a través de una interfaz gráfica, permitiendo el monitoreo y control manual.

- **Tabla de Relés:** La tabla de relés es un componente crítico en la planta solar híbrida, utilizado para controlar y conmutar la conexión de dispositivos eléctricos dentro del sistema. Funciona como un interruptor programable, permitiendo al servidor o a otros controladores activar o desactivar circuitos específicos en función de las necesidades operativas. Esto es fundamental para la gestión eficiente de la energía, asegurando que solo se utilice la electricidad cuando y donde sea necesario. La tabla de relés está directamente conectada al servidor, que emite comandos para abrir o cerrar los relés según las condiciones operativas del sistema. Por ejemplo, el servidor puede desactivar ciertas cargas durante periodos de baja generación de energía solar para priorizar el suministro a dispositivos críticos.

Sin embargo, durante el desarrollo y la integración del sistema, se determinó que la función de la tabla de relés podía ser realizada eficientemente por otro de los dispositivos ya empleados, específicamente los dispositivos Shelly. Estos módulos de control inteligente ya proporcionan la capacidad de gestionar el encendido y apagado de cargas eléctricas, lo que hacía redundante el uso de una tabla de relés adicional. Por esta razón, se decidió no incluir la tabla de relés en la versión final de la planta, simplificando la arquitectura y evitando la duplicación de funciones.

6.3 Recopilación y Procesamiento de Datos

Cada dispositivo del sistema recopila información específica del entorno. Los *ADAM* recogen datos eléctricos e hídricos, el *Eos Array* monitorea la producción solar, los dispositivos *Shelly* controlan el estado de las cargas y actúan como switches, el *SAI* vigila la estabilidad del suministro, y el inversor *Solax* supervisa la conversión de energía. Toda esta información se envía al servidor, donde se procesan los datos en tiempo real. La recopilación y procesamiento de datos es una de las funciones más críticas de la planta solar híbrida, ya que permite el monitoreo continuo, la toma de decisiones automatizadas, y la optimización del rendimiento del sistema.

Una vez que los datos han sido recopilados por los diferentes dispositivos, son transmitidos al PC(servidor) a través del router, que actúa como el nodo central de comunicación. El procesamiento de estos datos ocurre en varias etapas:

- **Preprocesamiento:** Los datos recibidos pueden estar en diferentes formatos o requerir limpieza antes de ser utilizados. El servidor realiza tareas de preprocesamiento como la conversión de unidades, la corrección de errores, y el filtrado de datos ruidosos. Esto garantiza que solo se analicen datos precisos y consistentes.
- **Análisis en Tiempo Real:** Una vez que los datos han sido procesados, el servidor los analiza en tiempo real utilizando algoritmos y scripts predefinidos. Este análisis incluye la detección de anomalías y la evaluación del rendimiento de los subsistemas.
- **Almacenamiento e Historial de Datos:** Todos los datos recopilados y procesados se almacenan en el servidor para su posterior análisis. Este historial de datos permite realizar análisis a largo plazo para evaluar la eficiencia del sistema, y planificar mejoras futuras. También es útil para realizar auditorías y asegurar que el sistema está operando dentro de los parámetros establecidos.
- **Visualización de Datos:** El servidor también está encargado de generar informes y visualizaciones de los datos en tiempo real. Estas visualizaciones son accesibles a través de interfaces de usuario diseñadas para los operadores de la planta, proporcionando una visión clara y comprensible del estado y rendimiento del sistema en cualquier momento.

Por último, una vez recopilada y procesada la información, es utilizada para una variedad de propósitos que son clave para el éxito en la operabilidad de la planta:

- **Optimización del Rendimiento:** Al analizar continuamente los datos, se puede ajustar el sistema y los componentes, asegurando la máxima eficiencia en todo momento.

- **Mantenimiento Predictivo:** Los datos sobre el estado de los componentes pueden ser utilizados para predecir fallos antes de que ocurran. Esto permite realizar el mantenimiento preventivo y evitar interrupciones inesperadas en el sistema.
- **Respuesta a Emergencias:** En caso de que se detecten condiciones anómalas, como una caída significativa en la producción de energía o un fallo en la red eléctrica, el sistema puede activar automáticamente los mecanismos de respaldo, como las power banks o el SAI, para asegurar la continuidad de la operación.
- **Toma de Decisiones Estratégicas:** A largo plazo, los datos históricos nos permiten tomar decisiones estratégicas, como la expansión del sistema, la incorporación de nuevas tecnologías, o la optimización de los recursos existentes para mejorar la eficiencia y la sostenibilidad de la planta.

En resumen, la recopilación y procesamiento de datos es un proceso central en la operación de la planta solar híbrida, que permite no solo el monitoreo y control eficiente de todos los subsistemas, sino también la optimización continua del rendimiento, la prevención de fallos, y la toma de decisiones informadas para el futuro desarrollo de la planta.

6.4 Scripts de Lectura/Escritura

Los scripts de lectura y escritura son fundamentales para el funcionamiento de la planta, ya que permiten la interacción automatizada entre los diferentes dispositivos y el servidor. Estos scripts son programas o fragmentos de código diseñados específicamente para extraer datos de cada dispositivo(lectura), enviar comandos(escritura), y realizar acciones basadas en la información procesada. Este apartado detalla la importancia, estructura, y funcionamiento de estos scripts, así como su papel en la integración y operación del sistema.

En un sistema tan distribuido y automatizado como lo es esta planta, es esencial que todos los dispositivos puedan comunicarse de manera fluida y coordinada. Estos scripts son los encargados de facilitar esta comunicación, permitiendo al servidor obtener datos en tiempo real y emitir comandos a los dispositivos según sea necesario. Sin estos scripts, sería imposible supervisar y controlar el sistema de manera eficiente, ya que la interacción entre los componentes sería manual y propensa a errores.

6.4.1. Estructura de los Scripts de Lectura

Los scripts de lectura están diseñados para extraer información específica de los dispositivos conectados a la planta. Cada dispositivo tiene su propio conjunto de datos que puede proporcionar, y los scripts deben estar configurados para capturar esta información en un formato que el servidor pueda procesar.

1. **Conexión y Autenticación:** Antes de que un script pueda leer datos de un dispositivo, debe establecer una conexión con él. Esto implica la apertura de una conexión TCP/IP o el uso de protocolos específicos como ModbusRTU o TCP.
2. **Solicitud de Datos:** Una vez establecida la conexión, el script envía una solicitud al dispositivo para que proporcione los datos requeridos. Esta solicitud puede ser tan simple como una llamada a una función de la API del dispositivo o un comando específico que indica qué tipo de datos se necesitan.

3. **Recepción y Formateo de Datos:** Los datos que se reciben suelen estar en un formato bruto o codificado. El script de lectura se encarga de decodificar estos datos y transformarlos en un formato estándar que pueda ser fácilmente procesado por el servidor. Esto puede incluir la conversión de unidades.

4. **Almacenamiento Temporal y Transmisión:** Los datos leídos se almacenan temporalmente antes de ser transmitidos al servidor. Esto es útil para reducir la carga de la red si se están recolectando grandes volúmenes de datos o si se necesitan procesar múltiples lecturas antes de enviarlas. Finalmente, los datos son enviados al servidor para su análisis y almacenamiento a largo plazo.

6.4.2. Estructura de los Scripts de Escritura

Los scripts de escritura permiten al servidor enviar comandos a los dispositivos de la planta, ya sea para ajustar su operación, iniciar o detener procesos, o modificar configuraciones según las necesidades operativas.

1. **Preparación del Comando:** El primer paso en un script de escritura es preparar el comando que se va a enviar al dispositivo. Esto implica definir el comando en un formato que el dispositivo pueda entender y ejecutar.

2. **Verificación de Estado:** Antes de enviar un comando, el script debe realizar una verificación de estado para asegurarse de que el dispositivo está en condiciones de recibir y ejecutar el comando. Esto incluye comprobar que el dispositivo está conectado, que no hay errores previos, o que el sistema está en un estado que permite la acción solicitada.

3. **Envío del Comando:** Una vez verificado el estado, el script envía el comando al dispositivo. Este proceso puede implicar abrir una conexión de red, establecer una sesión segura, y finalmente, transmitir la instrucción.

4. **Confirmación de Ejecución:** Después de enviar el comando, el script de escritura espera una confirmación del dispositivo de que la acción ha sido ejecutada correctamente. Esto es crucial para asegurar que los cambios se han aplicado y para evitar inconsistencias en la operación del sistema. Si se detecta un error o una falta de confirmación, el script reenvía el comando o generar una alerta para revisar la situación.

5. **Registro de la Acción:** Finalmente, el script registra la acción realizada, incluyendo detalles como la hora de ejecución, el comando enviado, y la respuesta del dispositivo. Este registro es útil para auditorías y para diagnosticar problemas si surgen discrepancias.

6.4.3. Ejemplos Scripts Lectura/Escritura

Para ilustrar cómo funcionan los scripts de lectura y escritura en la planta, se incorporan dos ejemplos concretos. Estos ejemplos muestran el proceso completo, desde la extracción de datos de un dispositivo hasta la emisión de un comando para ajustar las operaciones del sistema. Estos ejemplos ayudan a entender mejor la función de cada subsistema.

- **Script de Lectura**

```
const ModbusRTU = require('modbus-serial');
// Configurar la conexion
const client = new ModbusRTU();
client.connectRTUBuffered('/dev/cu.usbserial-14120', { baudRate: 9600 })
    .then(() => {
        console.log('Conectado al puerto serie');
    })
    .catch((err) => {
        console.log('Error al conectar:', err.message);
    });
// Leer 2 registros cada 2 segundos
setInterval(() => {
    client.readHoldingRegisters('0x308', 2)
        .then((data) => {
            console.log('Registros leidos:', data.data);
        })
        .catch((err) => {
            console.log('Error al leer los registros:', err.message);
        });
}, 2000);
```

- **Script de Escritura**

```
const ModbusRTU = require("modbus-serial");
const client = new ModbusRTU();
const address = 23; // Direccion del mapa de registros(16-23) un registro por cada salida(Do0-Do7)
const binary = 0; // Activar(0)/desactivar(1) salida

// Abrir conexion con el puerto serial
client.connectRTUBuffered("COM3", { baudRate: 9600 }, write);
console.log('Conexion establecida con el dispositivo');
function write() {
    client.setID(1);

    client.writeCoil(address, binary)
    .then(() => {
        console.log(`Registro ${address} modificado con exito`);
      })
      .catch((error) => {
        console.error(`Error al modificar el registro ${address}`,
            error);
      })
      .finally(() => {
        client.close();
        //console.log('Conexion cerrada');
      });
}
```

La incorporación de estos ejemplos de scripts de lectura y escritura proporciona una visión práctica y concreta de cómo se gestionan las operaciones en la planta solar híbrida. Estos scripts ilustran cómo los datos son leídos y utilizados para tomar decisiones automatizadas que optimizan el rendimiento del sistema. Además, estos ejemplos pueden servir como referencia para futuros desarrollos, facilitando la creación de scripts adicionales que expandan o mejoren las capacidades del sistema.

En resumen, los scripts de lectura y escritura son el núcleo operativo que conecta los dispositivos de la planta solar híbrida con el servidor central. A través de estos scripts, el sistema puede recopilar datos precisos, tomar decisiones en tiempo real, y ejecutar acciones que optimizan el funcionamiento de la planta, asegurando así un rendimiento eficiente y confiable.

6.5 Pruebas

Antes de proceder con la integración completa del sistema, es esencial realizar pruebas exhaustivas de cada subsistema por separado. Este enfoque permite asegurar que cada componente funcione correctamente de manera independiente, lo que facilita la identificación de posibles problemas antes de combinar todos los elementos en un único sistema cohesivo.

Durante esta fase, se evalúa el rendimiento individual de los dispositivos, verificando que cumplen con sus especificaciones técnicas y que interactúan adecuadamente con el software de control. Además, se revisan las capacidades de comunicación y la fiabilidad de cada subsistema, asegurando que todos los datos se transmitan de manera precisa y que los comandos se ejecuten sin fallos. Esta metodología permite detectar y corregir errores tempranamente, lo que reduce significativamente el riesgo de fallos durante la integración final y la operación en campo.

A continuación, se muestran una serie de pruebas relacionadas con algunos de los subsistemas integrados en la solución final de monitorización:

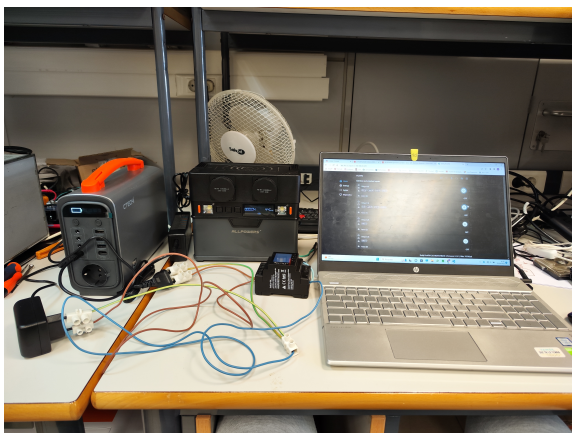


Figura 6.2: Ejemplo prueba dispositivo Shelly

En la ilustración 6.2 podemos observar un ejemplo de prueba del dispositivo shelly, el cual está conectado a varias power banks para comprobar el correcto funcionamiento del software, analizando los valores obtenidos al leer los distintos dispositivos.

Por otro lado, en la ilustración 6.3, podemos observar un ejemplo de prueba del dispositivo ADAM, en el cual podemos apreciar como este está conectado a una fuente de alimentación, dispone de un conversor analógico digital y de un potenciómetro, median-

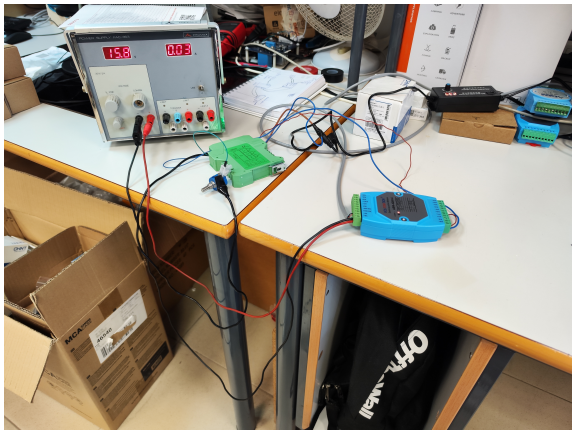


Figura 6.3: Ejemplo prueba dispositivo ADAM

te el cual, el dispositivo es capaz de ir midiendo la corriente a la par que vamos variando esta mediante el potenciómetro.

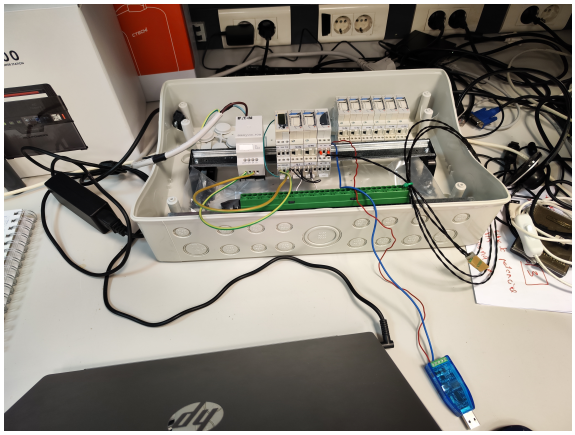


Figura 6.4: Ejemplo prueba dispositivo Eos-Array

En la figura 6.4, se muestra el montaje para la prueba del dispositivo EOS Array, al cual se le conecta una sonda de temperatura para medir los valores de temperatura en un entorno específico. Como podemos observar, los cables de la sonda de temperatura están conectados al dispositivo en el módulo P, permitiendo que los datos de temperatura sean leídos y registrados para su posterior análisis.

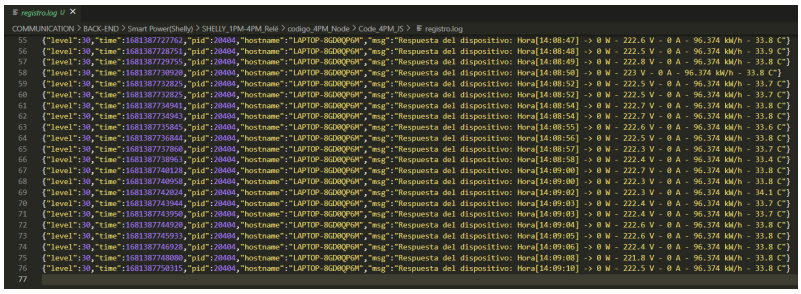


Figura 6.5: Ejemplo de recopilación de datos del dispositivo Shelly(log)

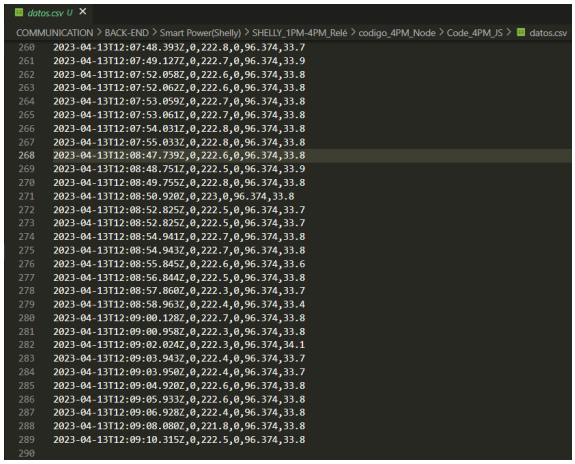


Figura 6.6: Ejemplo de recopilación de datos del dispositivo Shelly(CSV)

Las dos imágenes presentadas corresponden a la recopilación de datos obtenidos del dispositivo Shelly que monitorea parámetros energéticos y ambientales. La primera imagen, 6.5, muestra un archivo de registro en formato *.log*, donde los datos se presentan en un formato de texto estructurado y detallado, facilitando la visualización directa de cada entrada con su respectiva marca de tiempo, valores de voltaje, corriente, potencia, energía acumulada y temperatura. La segunda imagen, 6.6, muestra estos mismos datos exportados en un archivo CSV(.csv), un formato ampliamente utilizado para la manipulación de datos en software de análisis. Aunque ambos archivos contienen la misma información, el archivo *.log* es más adecuado para una revisión directa y rápida de los registros, mientras que el archivo *.csv* es ideal para análisis posteriores, permitiendo una fácil integración en herramientas de procesamiento de datos.

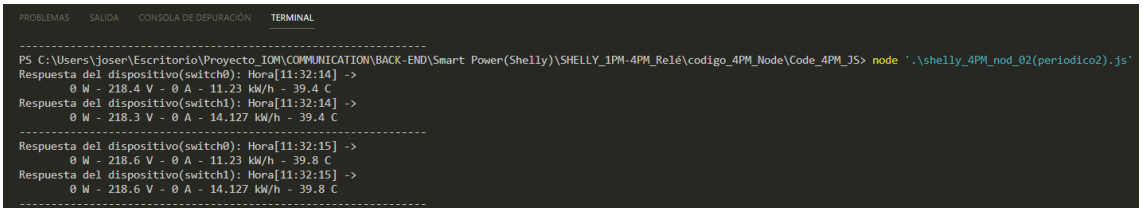


Figura 6.7: Ejemplo de valores de lectura del dispositivo Shelly

Para finalizar esta sección, se presenta la imagen 6.7, que complementa el análisis anterior al mostrar la misma información energética y ambiental del dispositivo Shelly, pero directamente desde la consola, antes de ser almacenada en archivos de tipo *log* o *CSV*. Esta imagen muestra los datos en bruto extraídos por el Shelly y visualizados en la consola de salida del código.

En resumen, las pruebas individuales de los subsistemas son un paso crítico para garantizar la solidez y fiabilidad del sistema antes de proceder con la integración completa y las pruebas de conjunto. Este proceso permite identificar y corregir cualquier fallo o deficiencia en cada componente específico, asegurando que estos funcionen de manera óptima y cumplan con los requisitos técnicos y operativos establecidos. Al validar cada subsistema por separado, se minimizan los riesgos de problemas inesperados durante la integración, facilitando una transición más fluida hacia la fase de pruebas finales, en la cual se evaluará el rendimiento global del sistema bajo condiciones reales de operación. Este enfoque meticuloso no solo mejora la calidad y la robustez del sistema, sino que también reduce el tiempo y los costos asociados con la resolución de problemas complejos en etapas posteriores.

CAPÍTULO 7
Solución completa de monitorización

En esta sección se presenta el código completo del software de monitorización desarrollado para la planta solar híbrida de bombeo de agua y fotovoltaica. Este software permite la recopilación de los datos de los sensores, su transmisión al servidor y su visualización en tiempo real, obteniendo la información correspondiente a los valores de la planta elegidos para un monitoreo óptimo. La monitorización continua es fundamental para garantizar la eficiencia operativa, la longevidad de los equipos, y la sostenibilidad económica y ambiental del proyecto.

En primer lugar, en el desarrollo de la solución de monitorización para la planta, se ha diseñado un algoritmo que gestiona y supervisa los diferentes componentes del sistema.

Para ilustrar el funcionamiento del algoritmo, en la siguiente imagen 7.1, se presenta un flujograma que detalla las distintas etapas y decisiones involucradas en el proceso. Este diagrama de flujo representa de manera visual y secuencial las operaciones clave del software de monitoreo.

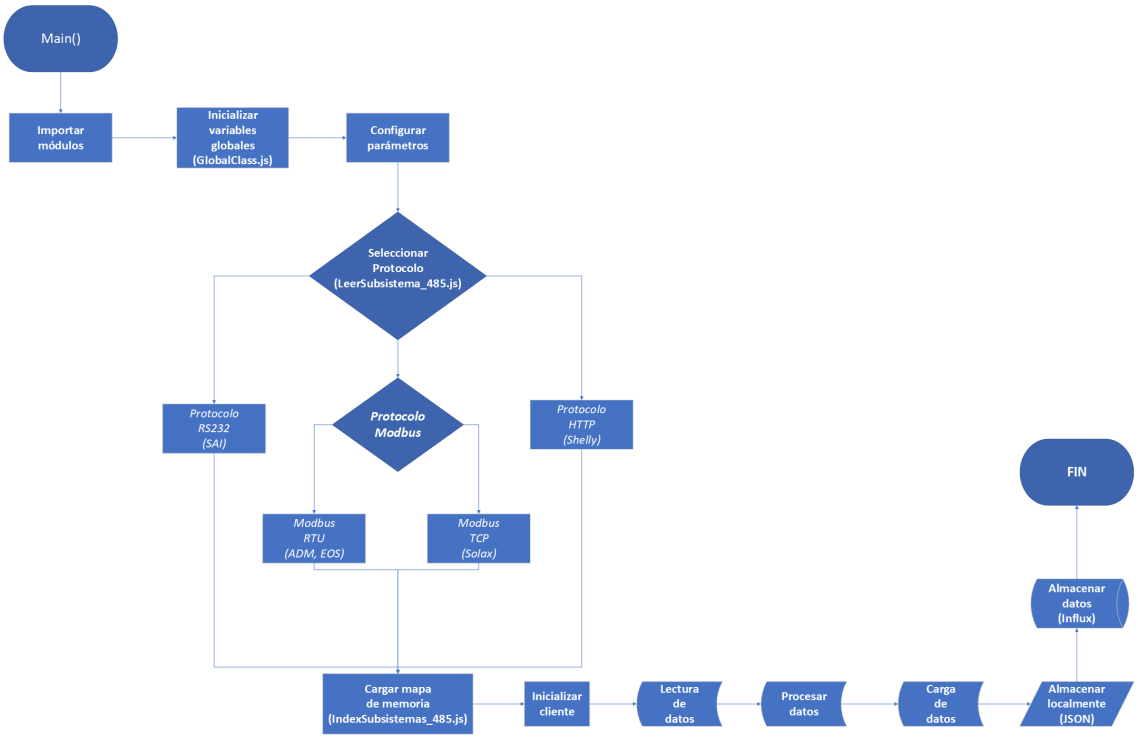


Figura 7.1: Flujograma del software de monitorización

Tras disponer del flujograma correspondiente al software de monitorización de la planta, nos introduciremos con la explicación del software creado.

El software desarrollado para la monitorización de la planta híbrida se organiza en varias clases principales, cada una de las cuales cumple una función específica dentro del sistema. Estas clases están diseñadas para trabajar de manera conjunta, permitiendo la lectura, el procesamiento, y la gestión de los datos de los distintos subsistemas de la planta. A continuación, se presenta un resumen de cada clase y su propósito:

- **Main_scada_IOM.js:** Es la clase principal del sistema, encargada de iniciar y gestionar el ciclo de vida del programa.
- **GlobalClass.js:** Esta clase se encarga de almacenar todas las variables globales utilizadas en el código, facilitando su acceso y modificación.
- **LeerSubsitema_485.js:** Aquí se gestionan los subsistemas que deben ser monitorizados. Se decide el protocolo de comunicación a utilizar y se carga el mapa de memoria asociado a cada subsistema.
- **IndexSubsistemas_485.js:** Contiene los subsistemas necesarios para realizar las lecturas.
- **Subsistemas:** En esta sección se presentan los mapas de memoria de cada subsistema integrado en la planta.
- **TiposDeDatos.js:** Clase encargada del manejo de los diferentes tipos de datos, proporcionando un conjunto predefinido de configuraciones para cada caso.
- **utiles.js:** Clase de utilidades encargada de agrupar y organizar funciones o métodos que son de uso común o general en diferentes partes del software.
- **Storage(Influx_Util.js):** Clase de utilidades encargada del almacenamiento de datos en la InfluxBBDD.
- **Errores(logger_pino.js, errorTable_pino.js):** Clase encargada de la captura de errores mediante la librería *Pino*.

En resumen, este software de monitorización desarrollado para la planta híbrida está cuidadosamente estructurado en varias clases, cada una cumpliendo un papel específico para garantizar el correcto funcionamiento del sistema. Esta organización modular permite una gestión eficiente y escalable de los datos, facilitando la supervisión de los distintos subsistemas de la planta. Gracias a esta arquitectura, el software puede procesar la información en tiempo real, almacenar datos de manera segura y gestionar cualquier error que pueda surgir, asegurando así la estabilidad y fiabilidad de la operación de la planta.

A continuación, se detalla, el propósito y funcionalidad de cada clase integrada en el software completo de monitoreo:

7.1 Clase Principal(main)

Es la clase principal del sistema, encargada de iniciar y gestionar el ciclo de vida del programa. Además, es la encargada de orquestar todas las operaciones desde el inicio hasta el cierre del programa. Coordina la ejecución de otras clases y subsistemas, asegurando que el software funcione de manera sincronizada.

Esta clase es parte de una aplicación SCADA, que monitorea y controla varios subsistemas a través de diferentes protocolos de comunicación como MODBUS RTU, MODBUS TCP, RS232, y HTTP.

El objetivo principal es leer datos de diferentes dispositivos, procesar la información y, finalmente, registrar los datos en un archivo JSON y en una base de datos InfluxDB.

La estructura general del código se divide en varias secciones: inicialización, configuración, ejecución del bucle principal, y lectura y procesamiento de datos de los subsistemas.

En resumen, este código representa un ciclo continuo de lectura y procesamiento de datos de un sistema SCADA. Cada ciclo consiste en:

- Leer la configuración y parámetros de los subsistemas.
- Establecer la comunicación con cada subsistema mediante el protocolo adecuado.
- Leer y procesar los datos del subsistema.
- Guardar los datos y prepararse para la siguiente iteración.

7.2 Clase para Variables Compartidas(GlobalClass)

Esta segunda clase del software tiene como objetivo centralizar las variables globales, los objetos de configuración y las funciones que serán utilizadas a lo largo de la aplicación. Esta estructura no solo facilita la organización del código, sino que también permite que diferentes módulos accedan a estas variables y configuraciones sin necesidad de duplicar código, lo que mejora la consistencia y eficiencia del sistema. Además, al mantener variables globales como configuraciones de red, parámetros de funcionamiento y estados de los dispositivos, esta clase asegura un acceso rápido y eficiente a la información crucial desde cualquier parte del software. Asimismo, facilita la modificación de estos parámetros globales sin necesidad de alterar múltiples partes del código, lo que incrementa significativamente la flexibilidad y mantenibilidad del sistema.

7.3 Clase para gestionar los subsistemas(LeerSubsistema_485)

Esta tercera clase para la monitorización de la planta tiene como objetivo gestionar la comunicación con los subsistemas conectados a través de los distintos protocolos de comunicación. Su principal función es establecer la conexión, enviar los comandos adecuados, recibir los datos desde los dispositivos, y procesar la información obtenida para integrarla en el sistema de monitoreo global.

Dentro de esta clase se implementan métodos para analizar y transformar los datos recibidos en formatos legibles y útiles para el sistema, permitiendo que los operadores de la planta puedan tomar decisiones informadas basadas en datos precisos. Cada uno de

estos métodos desempeña un papel clave en la funcionalidad general de la clase, facilitando una comunicación efectiva y el procesamiento preciso de datos desde los subsistemas de la planta.

Además, no solo se limita a la simple lectura de datos, sino que también maneja la interpretación de estos datos en función de los distintos subsistemas monitorizados, ajustándose a las especificaciones de cada uno. De esta manera, la clase juega un rol esencial en la monitorización continua y eficiente de la planta.

En resumen, este código está diseñado para monitorizar y registrar los datos de los subsistemas, manejando varios tipos de protocolos de comunicación con eficacia. Además, asegura robustez en el sistema al gestionar errores de comunicación y registrar logs, facilitando la solución de problemas y mejorando la fiabilidad.

7.4 Clase para lectura de subsistemas(IndexSubsistema_485)

Esta clase actúa como un índice o directorio que organiza y referencia todos los subsistemas monitorizados por el sistema, facilitando la gestión de las lecturas y la interacción con cada uno de ellos. En ella se definen y exportan un conjunto de configuraciones específicas para los diferentes subsistemas de la planta, lo que permite listar todos los subsistemas disponibles y acceder a ellos mediante un índice centralizado. Esto simplifica el manejo del software y mejora la eficiencia en las operaciones de lectura y escritura. Además, trabaja en conjunto con LeerSubsistema_485.js para identificar y gestionar las lecturas de los subsistemas adecuados. Asimismo, la estructura de esta clase facilita la incorporación de nuevos subsistemas en el futuro, permitiendo que se integren fácilmente en el sistema de monitorización existente.

7.5 Subsistemas(Mapas de Memoria)

En el software desarrollado para la monitorización de la planta híbrida, los mapas de memoria desempeñan un papel crucial en la estructura y el procesamiento de los datos de cada subsistema. Cada subsistema tiene su propio mapa de memoria, que define la disposición y el formato de los datos que se deben leer a través de la comunicación con el hardware. Estos mapas de memoria se utilizan para interpretar correctamente los registros leídos de los dispositivos y asegurar que los datos sean procesados de manera precisa.

A continuación, se presenta un ejemplo de un registro del mapa de memoria del inversor Solax integrado en el sistema de monitorización:

- **SOLAX_FIN.js**

```
var exports = module.exports = {};
const tipo = require("../utiles/Tipo_datos/TiposDeDatos").Radix;
exports.MemoryMap = Object.freeze([
    [
    { RegType: "InputRegister", Description: "GridVoltage", Address:
        0x000, Length: 1, Radix: tipo.BE.Uint16, Gain: 0.1, parse:
        true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX',
        tags: ['Subsistema'], tags_values: ['Red In'], field: "
        GridVoltage" } },
    ]
]);
```

Este código para cada uno de los subsistemas integrados en la planta define su mapa de memoria específico. En este mapa se utiliza un objeto que contiene los registros con información detallada sobre cómo leer y procesar los datos desde cada subsistema.

En resumen, este código es responsable de definir cómo se deben leer y procesar los datos para cada registro dentro de cada subsistema. En él, se definen la dirección del registro, su tipo de dato, la manera en que debe ser interpretado y transformado, y cómo se debe almacenar en una base de datos de series temporales, en nuestro caso InfluxDB.

7.6 Clase para el manejo de los datos(TiposDeDatos)

Encargada del manejo de los diferentes tipos de datos, esta clase proporciona un conjunto predefinido de configuraciones para cada caso, asegurando que los datos recogidos y procesados sean tratados correctamente según su naturaleza, ya sean numéricos, booleanos, o de otro tipo.

Este código define un objeto en JavaScript que categoriza y organiza varios tipos de datos en función de su orden de bytes(*endianness*). Este es un concepto importante en la programación de bajo nivel y la comunicación con hardware, donde el orden de los bytes en la memoria puede variar dependiendo de la arquitectura del sistema o del protocolo de comunicación utilizado.

El objeto Radix actúa como un punto central de referencia para la interpretación de datos en los mapas de memoria de los subsistemas. Define los tipos de datos y su formato de bytes(Big Endian o Little Endian), lo que se utiliza para asegurar que los datos sean correctamente leídos y escritos. Esta estructura modular y centralizada facilita la gestión de la variedad de subsistemas y protocolos que están presentes en nuestra planta híbrida, asegurando la robustez y flexibilidad del sistema.

En resumen, este código es crucial en un sistema donde se interactúa con hardware o protocolos que requieren la lectura y escritura de datos binarios en un formato específico. Dependiendo de la arquitectura o el dispositivo con el que se esté comunicando, es posible que se necesite trabajar con datos en Big Endian o Little Endian. Este objeto Radix facilita el manejo de estos diferentes tipos de datos, proporcionando un conjunto predefinido de configuraciones para cada caso. Esto permite que el código sea más modular, reutilizable y fácil de entender, especialmente en aplicaciones que manejan múltiples dispositivos con distintos protocolos y órdenes de bytes.

7.7 Clase de Utilidades(Utiles)

En este software de monitorización, la clase **utiles.js**(también conocida como *helper class*) tiene como objetivo agrupar y organizar funciones o métodos que son de uso común o general en diferentes partes del software. Estas funciones son operaciones que no dependen de un estado específico del objeto, por lo que pueden ser implementadas como métodos estáticos o funciones independientes.

En resumen, la clase de utilidades es fundamental para centralizar y reutilizar funciones comunes, evitando la repetición de código y facilitando el mantenimiento y escalabilidad del software.

7.8 Clase de utilidades para BBDD(Influx_Util)

Este script es una pieza clave dentro del sistema de monitorización de la planta. Su diseño modular, combinado con el uso de InfluxDB, permite la gestión eficiente de los datos de la planta, incluyendo la recolección, almacenamiento, análisis y notificación de eventos críticos. La configuración detallada y la implementación cuidadosa de las funciones aseguran que el sistema pueda manejar de manera eficaz los datos en tiempo real, lo cual es esencial para mantener la planta funcionando de manera óptima. Además, InfluxDB es ideal para gestionar datos de series temporales, como los generados por los sensores(subsistemas) de la planta.

En resumen, este script está diseñado para interactuar con InfluxDB de manera robusta y eficiente, proporcionando funcionalidades clave para la monitorización y gestión de la planta. Desde la lectura de datos históricos hasta el registro de alertas en tiempo real, cubriendo operaciones necesarias para mantener la planta operativa y bien supervisada.

7.9 Clases para el manejo y registro de errores(logger_pino, error_pino)

Por último, como scripts claves en la creación del software, nos encontramos con los códigos relacionados con el registro de fallos. Estos se basan en la detección y grabado de los fallos generales del software y serán registrados mediante la librería *Pino*. A continuación, se muestran y se detallan ambos códigos:

- **logger_pino.js:** Este script se encarga de gestionar el registro(logging) de errores y mensajes en un archivo log(log_error.js) utilizando la biblioteca *pino*, que es una librería de logging de Node.js.
- **errorTable_pino.js:** Este archivo define un objeto de mapeo de errores que se utiliza para proporcionar descripciones detalladas de errores en diferentes tipos de comunicación y componentes del sistema. Se integra con el sistema de logging(logger_pino.js) para proporcionar información detallada sobre los errores registrados, facilitando el diagnóstico y la resolución de problemas.

En resumen, los códigos proporcionan un sistema integrado para la gestión y registro de errores, mejorando la capacidad para mantener, monitorear y resolver los problemas de la planta.

CAPÍTULO 8
Puesta en marcha de la planta

Tras finalizar con la selección de los dispositivos y completar el desarrollo del software necesario para la operación y monitoreo de la planta, el siguiente paso crítico en este proyecto es la puesta en marcha del sistema. Este proceso implica no solo la instalación física de todos los componentes, sino también la integración de los diferentes subsistemas y la realización de pruebas exhaustivas para asegurar un funcionamiento óptimo desde el primer día. La puesta en marcha es la culminación de todos los esfuerzos previos y marca el inicio de la lucha para proporcionar energía y agua de manera sostenible a la aldea africana.

Para asegurar el éxito de la puesta en marcha, se tomó la decisión de montar inicialmente una planta de prueba en un entorno controlado. Esta fase piloto fue crucial para validar la integración de todos los subsistemas, probar el software desarrollado, y garantizar que todos los componentes funcionaran correctamente en conjunto.

En las siguientes imágenes se puede observar los cuadros eléctricos integrados en la infraestructura de nuestra planta de prueba. En el primer cuadro, figura 8.1 podemos observar dispositivos como el ADAM y el EOS y en el segundo cuadro, figura 8.2 se pueden observar el router, el PC y el otro dispositivo ADAM.

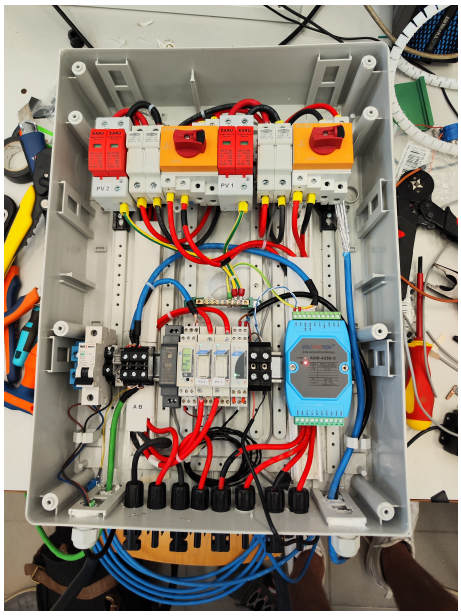


Figura 8.1: Cuadro de prueba 1

Figura 8.2: Cuadro de prueba 2

El montaje de la planta demo en un entorno controlado fue una decisión estratégica que resultó esencial para el éxito del proyecto. Esta fase previa permitió identificar y solucionar posibles problemas antes de trasladar la planta a su destino final, minimizando así los riesgos de fallos durante la instalación en campo. A través de pruebas exhaustivas y ajustes operativos, se logró una integración óptima de los componentes, lo que aseguró un funcionamiento fluido y sin contratiempos una vez instalada la planta en la aldea.

Una vez que la planta demo estuvo completamente operativa y se confirmaron su eficiencia y estabilidad, se procedió a replicarla en la aldea. En la siguiente imagen, figura 8.3, se puede observar un ejemplo de la planta demo montada en una zona controlada. Este enfoque sistemático no solo redujo los riesgos asociados con la instalación en campo, sino que también garantizó que el sistema estuviera optimizado y listo para enfrentar las condiciones reales del entorno donde se implementó de manera definitiva.

Figura 8.3: Planta de Prueba

En esta sección, se detallan los pasos llevados a cabo para la instalación y puesta en marcha del sistema en la comunidad, asegurando su funcionamiento óptimo y sostenible a largo plazo:

1. **Instalación Física:** Con todos los componentes y equipos listos, empezamos con la instalación física de la planta. Este proceso consiste en la correcta colocación de los paneles solares, la instalación del sistema de bombeo, y la integración de los sensores y dispositivos de monitoreo, asegurando que cada subsistema esté adecuadamente conectado y preparado para su operación. Una instalación precisa es fundamental para garantizar el rendimiento y la durabilidad del sistema a largo plazo.

    - **Infraestructura de la Planta Solar:** El primer paso en la instalación fue la colocación de las estructuras de soporte para los paneles solares. Se seleccionó una ubicación óptima con máxima exposición solar durante todo el día. Los paneles fueron montados con una inclinación ajustada para captar la mayor cantidad de energía posible. Posteriormente, se realizó el cableado entre los paneles y el controlador de carga, asegurando una correcta conexión eléctrica. En la siguiente ilustración 8.4, se muestra la infraestructura instalada en la pequeña aldea africana.

Figura 8.4: Infraestructura solar en Zimbabue

    - **Instalación del Sistema de Bombeo:** La bomba de agua fue instalada a un pozo de agua potable, cuya función era bombear y almacenar el agua en unos bidones para su posterior uso, tanto para uso particular como para regadío, como se puede observar en la siguiente imagen 8.5, en la cual se muestra la infraestructura correspondiente al almacenaje de esta. La instalación incluyó el acoplamiento de la bomba con tuberías adecuadas para asegurar un flujo de agua constante. Por último, conectada a las baterías, garantizando un suministro eléctrico estable para su operación.

Figura 8.5: Infraestructura para bombeo de agua en Zimbabue

- **Implementación del Sistema de Monitoreo:** Se instalaron los sensores para el monitoreo de la producción de energía, estado de las baterías, y flujo de agua. Estos sensores fueron estratégicamente colocados para maximizar la precisión de los datos recolectados. Todos los sensores fueron conectados al sistema *scada*, el cual estaba configurado para transmitir la información al servidor remoto y a la aplicación local.

2. **Integración y Pruebas:** Una vez completada la instalación física, era esencial proceder con la integración de todos los subsistemas y realizar las pruebas pertinentes. Este paso garantiza que los componentes trabajen en conjunto de manera eficiente y que el sistema funcione conforme a lo diseñado. Durante esta fase, se realizan las verificaciones de conectividad, los ajustes de configuración, y las pruebas operativas en campo para asegurar que la planta esté lista para su operación ininterrumpida.

    - **Pruebas de Conexión:** Se realizaron pruebas de conexión para verificar la continuidad eléctrica y la integridad de todas las conexiones. Estas pruebas incluyeron la comprobación de dispositivos y la detección de posibles fallos de conexión y configuración.
    - **Pruebas de Funcionamiento de la Bomba:** La bomba fue activada bajo condiciones controladas para asegurar su correcto funcionamiento. Se monitoreó el flujo de agua y la respuesta del sistema bajo diferentes niveles de carga, ajustando los parámetros del controlador de la bomba según fuera necesario.
    - **Monitoreo de Energía:** Durante un período de prueba, el sistema de monitoreo registró datos sobre la producción de energía solar, el consumo energético de la bomba, y el estado de las baterías. Estos datos fueron analizados para asegurar que el sistema estaba operando dentro de los parámetros esperados.

3. **Capacitación Local(Formación Técnica):** Se organizó un programa de capacitación para los miembros de la comunidad local. Durante este programa, se enseñó a los participantes a operar el sistema, interpretar las lecturas de los sensores, y realizar mantenimiento básico. La capacitación incluyó sesiones prácticas en las que los participantes pudieron interactuar directamente con el sistema.

4. **Monitoreo Remoto:** Por último, se implementó el sistema de monitoreo remoto para seguir evaluando el rendimiento del sistema a largo plazo desde nuestro laboratorio de trabajo. Este monitoreo permite detectar problemas de forma temprana y coordinar intervenciones si es necesario. Además, la planta de prueba sigue en continuo funcionamiento para poder adelantarse a posibles errores que pudieran surgir a posterior y así poder intervenir con antelación, evitando posibles problemas en la aldea, ya que el sistema debe de estar funcionando de la manera más óptima posible y cualquier fallo en la comunidad puede ser crítico para los aldeanos.

En la siguiente imagen 8.6, se puede observar la instalación de toda la parte hardware del sistema en Zimbabue.

Figura 8.6: Hardware de la planta en Zimbabue

La puesta en marcha de la planta ha sido un proceso meticulosamente planificado y ejecutado, reflejando el esfuerzo conjunto de la fase previa de desarrollo e integración de los subsistemas. Desde la selección inicial de los equipos hasta la implementación del software de monitoreo y control, cada paso ha ido orientado a garantizar la máxima eficiencia y sostenibilidad del sistema en condiciones reales.

Por último, como paso final de este proyecto, únicamente nos queda hablar sobre los resultados obtenidos en la monitorización mediante una serie de capturas que reflejan el estado actual de la planta en tiempo real.

En este punto se presentan una serie de capturas de pantalla obtenidas durante la monitorización de la planta, las cuales ilustran el funcionamiento y la eficiencia del sistema. Estas imágenes, extraídas del software de monitorización utilizado, reflejan cómo se gestionan y analizan los datos recopilados por los diferentes subsistemas. A continuación, se ofrece una explicación detallada de cada imagen, destacando los parámetros críticos supervisados y cómo esta información es utilizada para optimizar las operaciones de la planta y asegurar su correcto rendimiento.

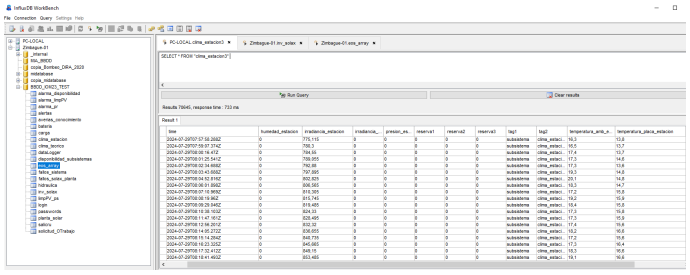


Figura 8.7: Ejemplo de recopilación en InfluxDB

La imagen 8.7, muestra la interfaz de InfluxDB, en la cual se visualiza la estructura de la base de datos que almacena los datos recopilados por diversos subsistemas, como las condiciones climáticas y el rendimiento de los paneles solares. La consulta SQL ejecutada en el panel derecho despliega una tabla con registros de parámetros clave como la humedad, irradiancia, presión, y temperatura. Esta herramienta permite acceder y analizar estos datos en tiempo real, facilitando la toma de decisiones para optimizar el funcionamiento de la planta solar.

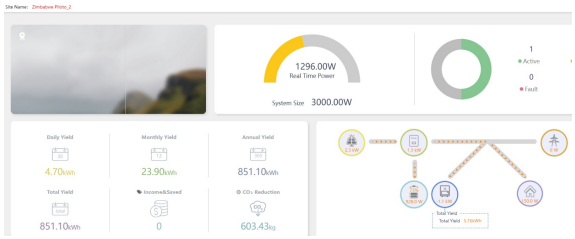


Figura 8.8: Ejemplo de lectura del Inversor Solax

La imagen 8.8, muestra la interfaz de monitoreo del inversor SolaX, que se utiliza para visualizar en tiempo real los parámetros de generación de energía solar de la planta. En la parte superior, se destaca la potencia generada en tiempo real, que en este caso es de 1296 W, junto con el tamaño total del sistema, que es de 3000 W.

La interfaz también presenta un resumen del rendimiento energético del sistema en diferentes periodos: rendimiento diario(Daily Yield) con 4.70 kWh, rendimiento mensual(Monthly Yield) con 23.90 kWh, y rendimiento anual(Annual Yield) con 851.10 kWh. Estos valores ayudan a entender la eficiencia y la producción energética acumulada en distintos intervalos de tiempo.

En la parte inferior derecha de la imagen, se muestra un diagrama de flujo de energía, que indica cómo se distribuye la energía generada por los paneles solares. Por ejemplo, de los 2.3 kW generados por los paneles solares, 1.3 kW se están utilizando para cargar la batería, que está al 73 % de su capacidad, mientras que 150 W están siendo consumidos por la red doméstica, y el resto de la energía está siendo almacenada o gestionada de acuerdo a la demanda.

El sistema también informa sobre la reducción de emisiones de CO2, indicando un total de 603.43 kg de CO2 evitados, lo que refleja el impacto ambiental positivo de la planta.

Esta herramienta de monitoreo es crucial para supervisar y optimizar el funcionamiento de la planta solar, asegurando que se aproveche al máximo la energía solar generada y se mantenga un control eficiente del almacenamiento y uso de la energía.

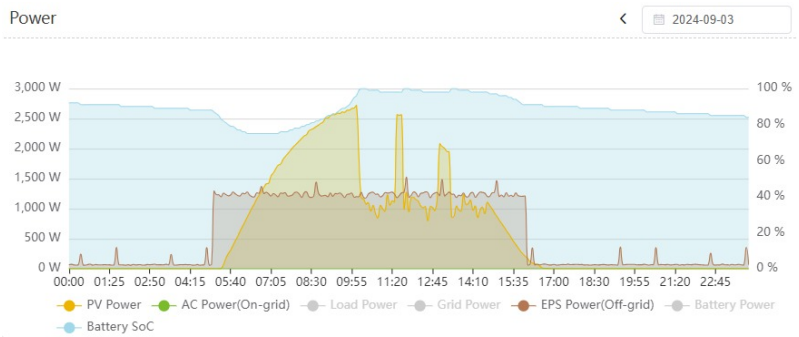


Figura 8.9: Ejemplo de gráfica de monitoreo de la planta

La imagen 8.9, muestra un gráfico generado por el software de monitoreo del inversor Solax, diseñado para visualizar el comportamiento de la potencia generada y consumida en una planta solar a lo largo del día.

El gráfico presenta varias curvas que representan diferentes parámetros clave de la planta:

- **PV Power(Potencia Fotovoltaica) - Representada en color amarillo:** Muestra la potencia generada por los paneles solares a lo largo del día. Se observa un incremento gradual de la potencia conforme avanza la mañana, alcanzando su pico máximo alrededor del mediodía, cuando la irradiación solar es mayor.
- **AC Power(On-grid) - Representada en color marrón:** Indica la potencia que está siendo utilizada desde la red eléctrica convencional. Este parámetro es crucial para entender cuánto de la energía consumida proviene de la red, en lugar de los paneles solares.
- **Battery SoC(Estado de Carga de la Batería) - Representada en color azul:** Muestra el nivel de carga de la batería en porcentaje a lo largo del tiempo. Se observa cómo la batería se carga durante las horas de mayor generación solar y cómo se descarga para suministrar energía durante las horas de menor generación.
- **Load Power (Potencia de la Carga):** La potencia demandada por los dispositivos conectados en la planta.
- **EPS Power(Off-grid):** La potencia suministrada cuando el sistema está operando fuera de la red(modo de respaldo), si es que ocurre.

En conjunto, este gráfico permite una visualización integral del rendimiento del sistema solar, mostrando cómo la energía solar generada es utilizada directamente, almacenada en baterías, o complementada con energía de la red. Es una herramienta esencial para optimizar el uso de la energía en la planta y para garantizar que el sistema funcione de manera eficiente y sostenible.

Figura 8.10: Ejemplo de recopilación de datos mediante Grafana

En la última imagen 8.10, se presentan una serie de capturas de pantalla obtenidas del software Grafana, que es utilizado para la visualización de datos en tiempo real de diversos parámetros críticos en la operación de la planta. Estas gráficas permiten un análisis detallado y una comprensión más profunda del comportamiento del sistema bajo diferentes condiciones operativas. A continuación, se describen brevemente las gráficas incluidas:

- **Hydraulic Pressure + Irradiance:** Esta gráfica muestra la relación entre la presión hidráulica y la irradiancia solar. Es crucial para evaluar cómo las variaciones en la irradiancia afectan la presión en los sistemas hidráulicos de la planta.
- **Total Plots for Article:** Aquí se muestra una recopilación de datos clave que se utilizan para análisis más profundos. Esta gráfica permite visualizar tendencias generales y anomalías potenciales en el rendimiento del sistema.
- **Irradiancia del Dispositivo EOS:** Esta gráfica se centra en la medición de la irradiancia captada específicamente por el dispositivo EOS, permitiendo monitorear de cerca el rendimiento de esta parte del sistema.
- **Irradiancia, PV Power and EPS Output Power:** En esta última gráfica, se visualiza la correlación entre la irradiancia solar, la potencia generada por los paneles fotovoltaicos y la salida de energía del sistema de alimentación de emergencia(EPS). Es fundamental para entender la eficiencia del sistema en la conversión de la energía solar en electricidad utilizable y cómo se gestiona la energía en situaciones de emergencia.

En resumen, la puesta en marcha de la planta solar híbrida ha sido un proceso exitoso que refleja la importancia de una planificación cuidadosa, la realización de pruebas previas, y la ejecución metódica de cada etapa. Este proyecto no solo proporciona soluciones tecnológicas sostenibles para las necesidades energéticas y de agua de la aldea, sino que también sienta un precedente para futuras implementaciones en otras comunidades con desafíos similares. La experiencia adquirida durante este proceso será invaluable para futuras expansiones o mejoras, garantizando que la tecnología siga evolucionando en beneficio de las comunidades más necesitadas.

CAPÍTULO 9
Conclusiones

Al culminar este trabajo de fin de grado, resulta fundamental reflexionar sobre los objetivos alcanzados, los desafíos superados y las implicaciones más amplias de las soluciones propuestas. Este proyecto ha sido una travesía que no solo ha abordado un problema técnico específico a la implementación de una planta solar híbrida para el bombeo de agua en una aldea africana, sino que también ha explorado las intersecciones entre tecnología, sostenibilidad, y desarrollo comunitario. A lo largo de las diferentes etapas de diseño, desarrollo, implementación y puesta en marcha, se han adquirido valiosas lecciones que no solo validan la viabilidad del proyecto, sino que también aportan un marco de referencia para futuras iniciativas en contextos similares.

Las conclusiones de este proyecto no se limitan a un mero balance de éxitos y fracasos, sino que se busca ofrecer una visión general de los logros alcanzados y su impacto a corto, mediano y largo plazo. Asimismo, la evaluación del impacto social, económico y ambiental de la planta solar híbrida permite contextualizar el proyecto dentro de un marco más amplio de desarrollo sostenible, subrayando su relevancia más allá del ámbito técnico.

Este apartado también invita a una reflexión crítica sobre las limitaciones encontradas y las oportunidades que surgen a partir de ellas. En un mundo en constante cambio, donde las necesidades energéticas e hídricas de las comunidades vulnerables siguen siendo apremiantes, es fundamental que los proyectos de ingeniería se enfoquen no solo en soluciones técnicas, sino en la creación de modelos replicables, escalables y sostenibles. Por tanto, las conclusiones que se presentan a continuación no solo cierran un ciclo, sino que también abren nuevas vías de investigación y acción, orientadas a maximizar el impacto positivo de este y otros proyectos futuros.

Con esto en mente, las siguientes conclusiones ofrecen un resumen detallado de los aprendizajes más significativos, los logros alcanzados y las perspectivas futuras que emergen a partir de este proyecto, con el objetivo de contribuir al avance del conocimiento en el campo de las energías renovables y su aplicación en el desarrollo rural.

1. **Cumplimiento de Objetivos Técnicos:** Uno de los logros más significativos del proyecto ha sido la correcta integración de todos los subsistemas y componentes involucrados en el proyecto. Cada componente ha sido diseñado, probado y ajustado para garantizar un funcionamiento óptimo en condiciones reales. Además, la planta ha demostrado ser capaz de generar energía suficiente para alimentar el sistema completo, garantizando un suministro estable y continuo en la comunidad beneficiada. El diseño del sistema ha permitido que la planta opere de manera autónoma, reduciendo la dependencia de combustibles fósiles y minimizando el impacto ambiental.

2. **Superación de Desafíos Técnicos y Logísticos:** Durante la fase de implementación, se enfrentaron varios desafíos técnicos y logísticos que requirieron soluciones innovadoras y adaptativas. El proceso de montaje y prueba en una planta demo en un entorno controlado fue crucial para identificar posibles fallos y realizar ajustes antes de la instalación final en campo. Este enfoque proactivo no solo permitió optimizar el funcionamiento del sistema, sino que también proporcionó valiosas lecciones sobre la importancia de la flexibilidad y la capacidad de respuesta en proyectos de ingeniería complejos.

3. **Impacto en la Comunidad Local:** La instalación de la planta no solo ha resuelto problemas críticos relacionados con el acceso a la energía y al agua en la aldea, sino que también ha capacitado a los miembros de la comunidad para operar y mantener el sistema de manera autónoma. Esto no solo fortalece la sostenibilidad del proyecto, sino que también empodera a la comunidad, mejorando su calidad de vida a largo plazo. Al garantizar un suministro de agua confiable y sostenible, se han generado beneficios sociales y económicos significativos para la comunidad. El acceso constante al agua ha permitido mejorar las condiciones de salud y sanidad, reducir el tiempo dedicado a la recolección de agua y, en consecuencia, liberar tiempo para actividades productivas y educativas.

4. **Lecciones Aprendidas y Desafíos Superados:** Durante el desarrollo del proyecto, se presentaron desafíos que requirieron soluciones creativas y adaptaciones. Desde el montaje hardware hasta la adaptación del software a las necesidades específicas del entorno, cada reto fue abordado con una perspectiva orientada a la solución propuesta para el monitoreo. Estas experiencias han proporcionado un aprendizaje valioso, que será fundamental para futuras implementaciones ene proyectos similares en otras comunidades del continente africano.

5. **Potencial de Replicabilidad y Escalabilidad:** El éxito de este proyecto abre la puerta a su replicación en otras aldeas y comunidades que enfrentan desafíos similares a los de la aldea zimbabuense. La metodología desarrollada, junto con el enfoque en pruebas previas y capacitación local, sirve como modelo para proyectos futuros. Además, la escalabilidad del sistema, garantiza que los proyectos futuros puedan crecer o reducirse en función de las demandas locales, lo que lo convierte en una solución versátil y eficaz para una amplia gama de escenarios. La replicación en otras comunidades no solo amplía el impacto del proyecto, sino que también contribuye a la creación de un modelo sostenible de desarrollo que puede ser aplicado en otras regiones con condiciones similares.

6. **Futuros Proyectos y Mejoras:** A pesar del éxito del proyecto, se han identificado áreas que podrían beneficiarse de futuras investigaciones y mejoras. Por ejemplo, la integración de sistemas de almacenamiento de energía más avanzados, el uso de tecnologías de monitoreo más avanzadas, como sensores IoT de última generación y algoritmos de inteligencia artificial para el análisis predictivo, podría mejorar la eficiencia operativa del sistema. Estas mejoras permitirían un monitoreo en tiempo real más preciso, la detección anticipada de fallos y una gestión energética más inteligente, optimizando el uso de los recursos disponibles y reduciendo los tiempos de inactividad, pudiendo aumentar aún más la eficiencia y confiabilidad del sistema. Asimismo, la exploración de nuevas aplicaciones para la energía generada, como la electrificación de otros servicios comunitarios, representa una oportunidad para ampliar el impacto del proyecto. Investigaciones futuras podrían centrarse en reducir aún más los costos de instalación y operación, lo que haría que la implementación de este tipo de sistemas sea más accesible para una mayor cantidad de comunidades. Esto podría incluir el desarrollo de soluciones tecnológicas o el uso

de componentes más económicos. Además, la simplificación de los procesos de instalación y mantenimiento podrían reducir la necesidad de personal altamente especializado, lo que facilitaría la expansión del proyecto a comunidades con recursos limitados, como es el caso de estas pequeñas comunidades rurales del continente africano.

El desarrollo y la implementación de la planta solar híbrida ha representado un paso significativo en la integración de tecnologías renovables con el desarrollo comunitario sostenible. Este proyecto ha demostrado que, con una planificación cuidadosa, un enfoque adaptativo y una comprensión profunda de las necesidades locales, es posible no solo resolver problemas técnicos complejos, sino también generar un impacto positivo y duradero en comunidades vulnerables.

La experiencia adquirida a lo largo de este proceso resalta la importancia de abordar los proyectos de ingeniería con una visión integral, que vaya más allá de la solución inmediata y considere el bienestar social, económico y ambiental de las comunidades involucradas. Los logros obtenidos desde el suministro estable de agua hasta la creación de oportunidades de desarrollo local subrayan la capacidad transformadora que tienen las tecnologías limpias cuando se implementan de manera consciente y responsable.

Sin embargo, también queda claro que este proyecto es solo un punto de partida en un campo vasto y dinámico. Las lecciones aprendidas y las áreas identificadas para mejora ofrecen un rico terreno para futuras investigaciones y desarrollos, que podrían amplificar aún más los beneficios alcanzados y extenderlos a otras regiones. La replicabilidad del modelo y su potencial de escalabilidad auguran un impacto creciente, conforme se continúe perfeccionando la tecnología y adaptándola a diferentes contextos.

En definitiva, la planta no solo ha cumplido con sus objetivos iniciales, sino que ha demostrado la eficacia de combinar tecnología, sostenibilidad y empoderamiento comunitario para resolver problemas fundamentales en zonas rurales de difícil acceso, además de establecer un marco de referencia valioso para futuras iniciativas en el campo de las energías renovables y el desarrollo rural. Los objetivos propuestos han sido alcanzados con éxito, y el impacto positivo en la comunidad es innegable. Este trabajo no solo ha mejorado las condiciones de vida de la comunidad, sino que ha sentado las bases para un enfoque más amplio y sostenible hacia la provisión de recursos esenciales en áreas rurales. Con esto, se cierra una etapa exitosa y se abre una puerta hacia nuevas posibilidades y desafíos que, sin duda, seguirán enriqueciendo el campo de estudio y su aplicación práctica en los próximos años.

Bibliografía

[1] Dispositivo Carlo Gavazzi con módulos M, S y P. Consultado en https://www.gavazziautomation.com/docs/download_area/BRO%20EOS%20SPA.pdf.

[2] Dispositivos Shelly. Consultado en https://www.shellyespana.com/manual-shelly-en-espanol/.

[3] Dispositivos ADAM. Consultado en https://store.sdindustrial.com.mx/pdf/techTips/Tech%20Tips-Ene17.pdf.

[4] Inversor Solax X1. Consultado en https://www.solaxpower.com/uploads/file/x1-boost-g4-user-manual-es.pdf.

[5] Dispositivo SAI Salicru. Consultado en https://www.pccomponentes.com/que-es-un-sai-y-para-que-sirve.

[6] Power Bank. Consultado en https://almamater.es/docs/tecno/info_PB.pdf.

[7] Comunicación RS485. Consultado en https://www.eltima.com/es/article/rs485-communication-guide/.

[8] Comunicación Modbus. Consultado en https://www.cursosaula21.com/modbus-que-es-y-como-funciona/.

[9] Node.js. Consultado en https://apasionados.es/blog/nodejs-4430/#:~:text=Ligero%20y%20escalable.-,Node.,considerar%20en%20el%20desarrollo%20web..

[10] Planta solar híbrida Nigeria. Consultado en https://nep.rea.gov.ng/about-nep/.

[11] Sistema Híbrido Solar-Diésel en Malí. Consultado en https://www.mdpi.com/2071-1050/11/8/2356.

[12] Planta Solar en Namibia. Consultado en https://www.facebook.com/profile.php?id=100067923200074&paipv=0&eav=AfZJ7-gQW30sdpSjjzVD5HMOfESMa1TSiJSYMicqpu3QUsmk51MD7UgTrC5Q0z73_vw.

APÉNDICE A
Scripts y Software de Monitorización

Este apéndice tiene como objetivo proporcionar una referencia completa y detallada de todo el código desarrollado para la operación de los dispositivos individuales y el sistema de monitorización de la planta solar híbrida. En él se incluye el código fuente de cada subsistema, así como el software general que integra y gestiona las diferentes partes del sistema. Este material es esencial para comprender cómo se implementan las funcionalidades descritas en el cuerpo principal del proyecto, permitiendo reproducir y mantener el sistema en el futuro.

A.1 Script de cada Subsistema

En esta sección, se presentan los scripts desarrollados específicamente para cada subsistema de la planta solar híbrida. Estos scripts son fundamentales para la operación independiente de cada componente, ya que permiten la recopilación de datos, la ejecución de acciones automatizadas, y la comunicación entre dispositivos. Antes de integrarse en la solución final, cada script ha sido diseñado y probado para asegurar su correcto funcionamiento en el contexto particular de su subsistema.

A continuación, se detalla el código fuente de cada uno de estos scripts, proporcionando una visión clara de cómo se gestionan y procesan los datos en cada dispositivo. Esta documentación es esencial para comprender cómo cada subsistema contribuye al funcionamiento general de la planta.

A.1.1. Smart Power(Shelly)

```
1   const http = require('http');
2   const fs = require('fs');
3   // Direccion IP del dispositivo Shelly 4PM
4   const deviceIP = '192.168.33.1';
5   // Ruta para poder leer el estado actual del dispositivo
6   const url = `http://${deviceIP}/rpc/Shelly.GetStatus`;
7   setInterval(() => {
8       http.get(url, (response) => {
9           let data = '';
10
11          response.on('data', (chunk) => {
12              data += chunk;
13          });
14          response.on('end', () => {
```

```
                let power = JSON.parse(data)["switch:0"].apower;
                let current = JSON.parse(data)["switch:0"].current;
                let voltage = JSON.parse(data)["switch:0"].voltage;
                let energy = JSON.parse(data)["switch:0"].aenergy.total;
                let temperatura = JSON.parse(data)["switch:0"].temperature.tC;
                let datos=[power,voltage,current,energy,temperatura];
                let date = new Date();
                // Crear la linea del archivo CSV
                let line = `${date.toISOString()},${datos.join(',')}\n`;
                // Escribir la linea en el archivo CSV
                fs.appendFile('datos.csv', line, (err) => {
                    if (err) throw err;
                });
                console.log(`Respuesta del dispositivo: Hora[${date.
                    toLocaleTimeString()}] ->
                ${datos[0] + " W"} - ${datos[1] + " V"} - ${datos[2] + " A"} -
                    ${datos[3] + " kW/h"} - ${datos[4] + " C"}`);
            });
        }).on('error', (error) => {
            console.error(`Error al enviar la solicitud: ${error.message}`);
        });
    }, 1000);
```

A.1.2. ADAM

```
const ModbusRTU = require("modbus-serial");
const client = new ModbusRTU();
const address = 23; // Direccion del mapa de registros(16-23) un registro
    por cada salida(Do0-Do7)
const binary = 0; // Activar(0)/desactivar(1) salida
// Abrir conexion con el puerto serie
client.connectRTUBuffered("COM3", { baudRate: 9600 }, write);
console.log('Conexion establecida con el dispositivo');
function write() {
    client.setID(1);
    client.writeCoil(address, binary)
    .then(() => {
        console.log(`Registro ${address} modificado con exito`);
      })
      .catch((error) => {
        console.error(`Error al modificar el registro ${address}`, error);
      })
      .finally(() => {
        client.close();
        //console.log('Conexion cerrada');
      });
}
```

A.1.3. EosArray

```
const ModbusRTU = require('modbus-serial');
// Configurar la conexion
const client = new ModbusRTU();
client.connectRTUBuffered('/dev/cu.usbserial-14120', { baudRate: 9600 })
    .then(() => {
        console.log('Conectado al puerto serie');
    })
    .catch((err) => {
        console.log('Error al conectar:', err.message);
    });
```

```
    // Leer 2 registros cada 2 segundos
    setInterval(() => {
        client.readHoldingRegisters('0x308', 2)
            .then((data) => {
                console.log('Registros leidos:', data.data);
            })
            .catch((err) => {
                console.log('Error al leer los registros:', err.message);
            });
    }, 2000);
```

A.1.4. SAI

```
    const SerialPort = require('serialport').SerialPort;
    // Configuracion del puerto serial
    const portName = 'COM2';
    const baudRate = 2400;
    const dataBits = 8;
    const stopBits = 1;
    const parity = 'none';
    // Comando a enviar al dispositivo
    const command1 = 'QGS\r'; // The general status parameters inquiry
    const command2 = 'QFS\r'; // Fault Status Inquiry
    const command3 = 'QBTAH\r'; // The battery Total AH information Inquiry
    // Crear una instancia del puerto serial
      const port = new SerialPort({path:portName,
      baudRate,
      dataBits,
      stopBits,
      parity
    });
    // Funcion para enviar el comando al dispositivo
    function sendCommand() {
      port.write(command1, (err) => {
        if (err) {
          console.error('Error al enviar el comando:', err.message);
        }
      });
    }
    // Funcion de Parseo!!
    let datos = Buffer.from([]);
    function parseito (data) {
      const response = data.toString('utf8').trim();
      const values = response.split(' ');
      // Definimos el nombre de los campos a analizar
      const nombreCampos = ['InputVoltage', 'InputFrequency', 'OutputVoltage',
          'OutputFrequency', 'OutputCurrent',
                                'OutputLoad', 'PositiveBusVoltage', '
                                   NegativeBusVoltage', 'PBatteryVoltage', '
                                   N-BatteryVoltage', 'Temperature', 'More'
                                   ];
      //Campos a omitir
      const fieldToOmit = ['OutputFrequency', 'PositiveBusVoltage', '
          NegativeBusVoltage','N-BatteryVoltage'];     ;
      // Construimos el objeto con el nombre de los campos y sus valores
      const objectResult = {};
      values.forEach((value, indice) => {
        const nameField = nombreCampos[indice] || `Campo${indice + 1}`;
        if(!fieldToOmit.includes(nameField)) {
          objectResult[nameField] = value.trim(); // Con trim(), eliminamos los
               espacios en blanco
        }
```

```
    });
  // Extraer el valor del campo "More"
  const moreValue = objectResult['More'];
  //console.log(moreValue);
  const moreData = parseMoreValue(moreValue);
  //Agregar objeto al objeto resultante
  const objectFull = Object.assign({}, objectResult, moreData);
  console.log(objectFull);
  // Devuelve el objeto con el nombre de los campos y sus respectivos valores
  return objectResult;
  }
  function parseMoreValue(moreValue) {
    const parsedData = { // Cada bit corresponde a un campo, excepto el primer campo que se compone de dos bits
      OperatingMode: moreValue.slice(0, 2), // 00(standy), 01(lineInteractive) y 10(Online)
      UtilityFail: moreValue.slice(2, 3), // 0(Utility OK) y 1(Fail)
      BatteryLow: moreValue.slice(3, 4), // 0(Battery OK) y 1(Bateria baja)
      BypassBoostActive: moreValue.slice(4, 5), // 0(Boost mode) y 1(Bypass mode)
      UPSFailed: moreValue.slice(5, 6), // 0(OK) y 1(UPS Failed)
      EPO: moreValue.slice(6, 7),
      TestInProgress: moreValue.slice(7, 8),
      ShutdownActive: moreValue.slice(8, 9),
      BatSilence: moreValue.slice(9, 10),
      BatTestFail: moreValue.slice(10, 11),
      BatTestOk: moreValue.slice(11, 12)
    };
    return parsedData;
  }
  // Configurar el evento de recepcion de datos
  port.on('data', (dato) => {
    datos = Buffer.concat([datos, dato]);
    lastBuffer = datos; // Almacenamos el ultimo buffer recibido
  });
  // Imprimir por pantalla unicamente el ultimo buffer, concatenacion de todos los anteriores
  setTimeout(() => {
    const result = parseito(lastBuffer);
    // Acceder a cada campo y valor dentro del objeto resultante(result)
    for(const field in result) {
      if (result.hasOwnProperty(field)) {
          const valor = result[field];
          //console.log(`${field}: ${valor}`);
      }
    }
    port.close(); // Cerramos la conexion
  }, 1000);
  // Configurar el evento de apertura del puerto serial
  port.on('open', () => {
    console.log('Conexion con el puerto serial establecida');
    sendCommand();
  });
  // Configurar el evento de error en el puerto serial
  port.on('error', (err) => {
    console.error;});
```

A.1.5. Tabla de Relés

- **ModbusTCP**

```
    const net = require("net");
    var IP_Grun = "192.168.2.119";
```

```
    var Dir_reg = 0;
    var N_reg = 2;
    const socket = new net.Socket();
    socket.connect(502, IP_Grun, function () {
        console.log("Conectado");
        const message = Buffer.from([
            0x00, // Transaction identifier (2 bytes)
            0x01,
            0x00, // Protocol identifier (2 bytes)
            0x00,
            0x00, // Number of bytes in the message (2 bytes)
            0x06,
            0x01, // Unit identifier (1 byte)
            0x03, // Function code (1 byte)
            Dir_reg >> 8, // Starting address of the registers to read (2 bytes)
            Dir_reg & 0xff,
            0x00, // Number of registers to read (2 bytes)
            N_reg
        ]);
        socket.write(message);
        socket.on("data", function (data) {
            //Mostrar el mensaje
            console.log(data);
            // Obtener los valores de los reles de la lectura
            let v = data.readUInt16BE(9);
            let w = data.readUInt16BE(11);
            //Mostrar por pantalla estos valores
            console.log("Valores: " + v, w);
            //Determinar si el rele esta encendido o apagado
            var estadoRele1 = "";
            var estadoRele2 = "";
            if (w == 0) {
                estadoRele1 = "OFF";
                estadoRele2 = "OFF";
            }
            else if (w == 1) {
                estadoRele1 = "ON";
                estadoRele2 = "OFF";
            }
            else if (w == 2) {
                estadoRele1 = "OFF";
                estadoRele2 = "ON";
            }
            else {
                estadoRele1 = "ON";
                estadoRele2 = "ON";
            }
            var valorLeido = "Rele 1 = " + estadoRele1 + ", " + "Rele 2 = " + estadoRele2;
            console.log(valorLeido);
            socket.end();
        });
        socket.on("error", function (err) {
            console.log(err);
            socket.end();
        });
    });
```

- **RS485**

```
    const ModbusRTU = require("modbus-serial");
    const client = new ModbusRTU();
    client.connectRTUBuffered("COM3", { baudRate: 9600 }, write);
```

```
    function write() {
        client.setID(1);
        // write the values 0, 0xffff to registers starting at address 5
        // on device number 1.
        client.writeRegister(2, 0x0f00)
            .then(console.log)
            .then( ()=> {
              client.close();
              process.exit(1); }
    )}
```

A.2 Software Final

En esta sección del anexo se presenta el software completo desarrollado para la monitorización y gestión de la planta solar híbrida. Este software integra todos los subsistemas previamente detallados y proporciona una solución unificada para la recolección, procesamiento, almacenamiento y visualización de datos en tiempo real.

El código, organizado en distintas clases y módulos, se ha diseñado con un enfoque modular para facilitar su mantenimiento, actualización y escalabilidad. A lo largo de este apartado, se detalla la estructura del software y se incluyen los scripts utilizados, destacando su funcionalidad en la operación global del sistema.

A.2.1. Main_scada_IOM.js

```
// FICHERO: Main_scada_IOM.js

// Requerimientos de los Includes de la Aplicacion
global.Subsist_485 = require("./IndexSubsistemas_485");
const va = require("./Class/GlobalClass");
const ModbusRTU = require("modbus-serial");
const shelly = require("./Class/Shelly");
const LeerSubsistema_485 = require("./Class/LeerSubsistema_485");
const Utiles = require("./Class/utiles");
const { logError, logError_base } = require('./Class/logger_pino');  // LOGGER PINO
const config = require('./config.json'); // Fichero de configuracion de la Aplicacion ....
/////////////// Declaracion de variables globales ///////////////
// Valores de configuracion
const intervalTime = config.App.intervalTime; // Tiempo del bucle para leer todos los subsistemas (300 seg = 5 mtos)
const portName = config.salicru.portName;
const baudRate = config.salicru.baudRate;
const dataBits = config.salicru.dataBits;
const parity = config.salicru.parity;
var dato1;
var dato2;
//-----------------------------------------------------------------*//
main().catch((error) => {
    console.error(error);
    logError_base(`Error en el main: ${error.message}`);
});
//-----------------------------------------------------------------*//
async function main() {
    logError_base('Inicio Aplicacion:');
    va.t_start = Date.now(); // Inicio contar tiempo ....
    console.log("\n >>>> Inicializa el MAIN !!!");
    Utiles.retardo(3000); // Esperar 3 segundos
```

```
32          // Leer informacion de cada uno de los subsistemas existentes ... SOLAX
                , EOS, Salicru, ADAM
33          for (let subsist of Subsist_485) {
34              va.protocol = subsist.protocol;
35              const memoryMap = subsist.MemoryMap[0]; // Leer Mapa de la memoria
                    del Subsistema...
36              var esclavo = subsist.ID; // Valor del ID = xx del escalvo del
                    subsistema
37              var name = subsist.name; // Nombre del subsistema
38              //Obtener el numero de campos que dispone este subsistema
39              const narrays = Object.keys(memoryMap);
40              /**Verificar el PROTOCOLO*/
41              // Para realizar una sola inicializacion para el caso de MODBUS_TCP
                     ... ojo depende del puerto
42              // Para realizar una cada vez que sea un subsistema de RS485
                    MODBSU_RTU ... se abre , se lee y se cierra
43              if ((va.protocol == "MODBUS_RTU") || (va.protocol == "MODBUS_TCP"))
                     {
44                  va.print("\n >> Subsistema ---> ", subsist.name + " ID: " +
                        subsist.ID + ", " + "Protocolo: " + va.protocol, va.dbg);
45                  if (va.bool) { // Inicialmente = 'true'
46                      var cliente = new ModbusRTU(); // Crear instancia al
                            cliente RS485/Modbus
47                      var RS485T = new LeerSubsistema_485(va.medidas_arr); //
                            Cargar campos de cada subsistema a leer ....
48                  }
49                  // Que PROTOCOLO utiliza el subsistema ???
50                  if (va.protocol == "MODBUS_RTU") {
51                      if (va.boolConect == true) { // Si protocolo es "MODBUS_RTU
                            " ---> Abrir canal de comunicacion
52                          await cliente
53                              .connectRTUBuffered(va.port, va.ComOptions)
54                              .then(() => {
55                                  console.log("\n Conectado al puerto serie
                                        Modbus-RTU \n ");
56                                  va.boolConect = false; // Solo lo abrimos una
                                        vez ???
57                              })
58                              .catch((err) => { // Caso de Error de Apertura del
                                    Canal de Comunicacion
59                                  console.log("Error al conectar al puerto serie
                                        Modbus-RTU:", err.message);
60                                  va.boolConect = true; // Solo lo abrimos una
                                        vez ???
61                              });
62                      }
63                      va.bool = false; // Flag de estado para hacerlo solo una
                            vez ...
64                      cliente.setID(esclavo); // Asignar el valor del ID del
                            subsistema a la comunicacion...
65                  }
66                  //-----------------------------------------------------*//
67                  // Verificar el nombre del subsistema y Obtener los campos
                        configurados para escribir los datos
68                  switch (name) {
69                      case "SOLAX": // Para el Subsistema = 'SOLAX' obtener
                            campos a escribir los datos
70                          var datos_Fields = va.medidas_arr.solax[0].fields;
71                          break;
72                      case "ADM-4280-V":
73                          var datos_Fields = va.medidas_arr.adm_4280_v[0].fields;
74                          break;
75                      case "EOS-ARRAY":
76                          var datos_Fields = va.medidas_arr.eos_array[0].fields;
```

```
77                      break;
78                  case "ADM-4280-I":
79                      var datos_Fields = va.medidas_arr.hidraulica[0].fields;
80                      break;
81                  case "SALICRU":
82                      var datos_Fields = va.medidas_arr.salicru[0].fields;
83                      break;
84                  case 'SHELLY_4PM':
85                      var datos_Fields = va.medidas_arr.carga[0].fields;
86                      break;
87                  default:
88                  //break;
89              }
90              va.counter++;
91              // Leer informacion existente de los registros en el Mapa de
                    memoria de cada subsistema ???
92              for (registro of narrays) {
93                  const { Address: Address, Length: Length, Description:
                        Description } = memoryMap[registro];
94                  const { parse: parse, Radix: Radix, ginflux: grabar_influx,
                         Gain: Gain } = memoryMap[registro];
95                  if (va.protocol == "MODBUS_RTU") { // Caso del Protocolo =
                        "MODBUS_RTU"
96                      dato1 = await RS485T.LeerSub_485RTU( // Subsistemas de
                            485 ....
97                          subsist.name,
98                          Address,
99                          Length,
100                         cliente,
101                         memoryMap,
102                         parse,
103                         Radix,
104                         Gain,
105                         registro,
106                         narrays,
107                         esclavo,
108                         Description,
109                         datos_Fields
110                     );
111                 } else if (va.protocol == "MODBUS_TCP") { // Caso del
                        Protocolo = "MODBUS_TCP"
112                     dato2 = await RS485T.LeerSub_485TCP(  //SOLAX ....
113                         subsist.name,
114                         Address,
115                         Length,
116                         cliente,
117                         memoryMap,
118                         parse,
119                         Radix,
120                         Gain,
121                         registro,
122                         narrays,
123                         esclavo,
124                         Description,
125                         datos_Fields
126                     );
127                     console.log(dato2);
128                     if (Description == "Tempera_Radiador") { // Guardar en
                            estructura de Bateria en JSON
129                         console.log(datos_Fields);
130                         datos_Fields = va.medidas_arr.bateria[0].fields;
131                         console.log("---- Battery ----")
132                     }
133                     // Guardar en estructura de Fallos en JSON
```

```
                    if (Description == "wBattery1_Type") {
                        datos_Fields = va.medidas_arr.fallos_solax_planta
                            [0].fields;//wBattery1_Typer
                        console.log(datos_Fields);
                        console.log("---- Faults ----")
                    }
                }
                // Caso de ERROR de datos de recepcion ... ???
                if (dato1 < 0 || dato2 < 0) {
                    console.log(dato1);
                    console.log("Error inesperado");
                    break;
                }
                // Esperar un tiempo despues de la Lectura de datos del
                    subsistema en cuestion ???
                await new Promise(resolve => setTimeout(resolve, 600));
            }
            ////////////////////////////////////////////////////////
        } else if (va.protocol == "RS232") { // Caso del Protocolo = "RS232
            " --> SALICRU
            var RS485S = new LeerSubsistema_485(va.medidas_arr); // Cargar
                campos de cada subsistema a leer ....
            try {
                // Leer los registros del Mapa de Memoria del Subsistema en
                    funcion de su Protocolo ???
                datos_Fields = await RS485S.LeerSub_232RS(portName,
                    baudRate, dataBits, 1, parity)
                /// Formato a grabar: fields: { ups_status:,
                    voltage_bateria: , output_voltage: , output_current: ,
                    soc_battery: , comentario: ,
                // Para grabar en JSON e INFLUX
                va.medidas_arr.salicru.measurement = 'salicru';
                va.medidas_arr.salicru.fields.ups_status = parseInt(
                    datos_Fields.More, 2); // 2049
                va.medidas_arr.salicru.fields.voltage_bateria = parseFloat(
                    datos_Fields.PBatteryVoltage);
                va.medidas_arr.salicru.fields.output_voltage = parseFloat(
                    datos_Fields.OutputVoltage);
                va.medidas_arr.salicru.fields.output_current = parseFloat(
                    datos_Fields.OutputCurrent);
                // SOC de la bateria del SAI-Salicru.. calculado por tabla
                    y su ecuacion lineal ----
                va.medidas_arr.salicru.fields.soc_battery = Utiles.
                    calc_soc_salicru(parseFloat(datos_Fields.
                    PBatteryVoltage));
                va.medidas_arr.salicru.fields.comentario = ' Ninguno';
            } catch (error) {
                // Manejar el error
                console.log("Error protocolo RS232-SALICRU !!!");
            }
        } else if (va.protocol == "HTTP") { // SHELLY 4 PM (RELE Monofasico
             con medidas)
            //-----------------------------------------------------*//
            try {
                const IP_sh = '192.168.2.99'
                const se4PM = new shelly(IP_sh, 1, va.Dato1);  //
                    Instanciar a la Clase shelly ...
                const estado_sh = await se4PM.ReadStatus();
                console.log(" SHELLY 4PM Estado ---> ", estado_sh);
                const medidas_sh = await se4PM.Leer_medidas_SH4PM(IP_sh);
                // Para grabar en JSON
                va.medidas_arr.carga.measurement = 'carga';
                va.medidas_arr.carga.fields.corriente1 = medidas_sh.
                    corriente1; // Solo para caso Monofasico
```

```
180                     va.medidas_arr.carga.fields.corriente1 = medidas_sh.
                            corriente2; // Solo para caso Monofasico
181                     va.medidas_arr.carga.fields.corriente1 = medidas_sh.
                            corriente3; // Solo para caso Monofasico
182                     va.medidas_arr.carga.fields.corriente1 = medidas_sh.tension
                            ; // Solo para caso Monofasico
183                     va.medidas_arr.carga.fields.corriente1 = medidas_sh.
                            energia_s1;
184                     va.medidas_arr.carga.fields.corriente1 = medidas_sh.
                            energia_s2;
185                     va.medidas_arr.carga.fields.corriente1 = medidas_sh.
                            energia_s3;
186                     va.medidas_arr.carga.fields.corriente1 = medidas_sh.
                            energia_s4;
187                     va.medidas_arr.carga.fields.corriente1 = medidas_sh.
                            energia_histo;
188                 } catch (error) {
189                     // Manejar el error del SHELLY 4PM (Monofasico)
190                     console.log(" ERROR DE SHELLY ....");
191                 }
192             }
193             console.log("\n >>FIN DEL BUCLE de la Lectura de subsistema --->",
                    subsist.name + "\n");
194             await new Promise((resolve) => setTimeout(resolve, 2000));
195         } /**FIN DEL LOOP de la Lectura de todos los Subsistemas en curso*/
196         console.log("\n >>GUARDAR EN JSON E INFLUX !!! ");
197         // Grabar fichero JSON del subsistema Leido .. OJO hay que poner el
                nombre del fichero JSON con fecha_datos_IOM.json ???
198         // en variable: va.medidas_arr --> Es el valor de los datos de los
                subsistemas en cada iteracion del Loop
199         await Utiles.loopEscrituturaJSON();         //
                ////////////////////////////////////////////////////////////
200         // GRABAR EN INFLUX------------------------------ VER FICHERO
                GLOBALCLASS (global.medidas_arr)
201         // Grabar en INFLUX todos los datos relacionados con subsistema SOLAX:
202         // inv_solax, bateria, eos_array, hidraulica (adm_4280_I), carga,
                fallos_solax_planta, adm_4280_v
203         //------------------------------------------------------------//
204         new Promise((resolve) => setTimeout(resolve, 3000));
205         // Medir el tiempo de ejecucion del codigo anterior(Bucle de Lectura de
                todos los subsistemas)
206         var t_end = Date.now();
207         var elapsedTime = t_end - t_start;
208         // console.log("Retardo --> ", elapsedTime);
209         const TIEMPO_RESTANTE = Math.max(intervalTime - elapsedTime, 0);
210         //TIEMPO_RESTANTE = TIEMPO_RESTANTE + 10000;
211         setTimeout(main, TIEMPO_RESTANTE + 10000);
212         // Crea un temporizador que se ejecuta cada segundo
213         var tiempoRestante = TIEMPO_RESTANTE + 10000;
214         const temporizador = setInterval(() => {
215             // Calcula el tiempo restante antes de que se ejecute la tarea
216             tiempoRestante = tiempoRestante - 1000;
217             if (tiempoRestante <= 1000) clearInterval(temporizador);
218             // Muestra el tiempo restante en la pantalla
219             //contador = 0;
220             console.log(
221                 `>> La tarea del BUCLE se ejecutara en ${tiempoRestante}
                        milisegundos. !!!`
222             );
223         }, 1000);
224         if (va.boolConect == false) {
225             await cliente.close();
226         } // ES IMPORTANTE PARA PODER REABRIR LAS COMUNICACIONES ESPECIFICAS
                RTU
```

```
227         // Reiniciar flags de las conexiones de solo una vez ....
228         va.bool = true; // var cliente = new ModbusRTU(); var RS485T = new se(
                va.medidas_arr); Instanciar una sola vez
229         va.boolConect = true; // await cliente.connectRTUBuffered(va.port, va.
                ComOptions) para despues de cerrar volver crear
230         va.counter = 0; // en cada vuelta de Loop
231     }       ////////////////////  FIN DEL CODIGO ////////////////////
```

A.2.2. GlobalClass.js

```
1   // FICHERO: variablesGlobales.js
2   const LeerRS485_RTU = require('./LeerSubsistema_485');
3   /**VARIABLES COMUNES A TODOS LOS MODULOS Y CLASES*/
4   /**CONFIGURACION E INICIALIZACION*/
5
6   // Variables globales
7   global.t_start = 0;
8   global.t_end = 0;
9   global.counter = 0;
10  global.client = "";
11  global.protocol = "";
12  global.openn = true;
13  global.bool = true;
14  global.IpSolax = '158.42.82.129';
15  global.Dato1 = 0;
16  global.boolConect = true;
17
18  global.LeerRS485_TCP = new LeerRS485_RTU();
19  global.arraySolax = [];
20  global.arrayEos = [];
21  global.arrayADM_V = [];
22  global.arrayADM_I = [];
23
24  //--------------------OBJETO MEDIDAS ARRAY-----------------------//
25  global.medidas_arr = {
26
27      solax: [
28          {
29              measurement: 'inv_solax',
30              fields: {
31                  GridVoltage: 0.0,              //------ V_TENSION AC_IN
                                                0X0000      1
32                  GridCurrent: 0.0,              //------ I_CORRITE AC_IN
                                                0x0001      2
33                  GridPower: 0.0,                //------ P_POWER    AC_IN
                                                0x0002      3
34                  GridFrequency: 0.0,           //------ Hz_FRECUE AC_IN
                                                0x0007      4
35                  // PV SYSTEM
                        ------------------------------------------------------------
36                  PvVoltage1: 0.0,                //----    Tension_PV01
                                              0x0003      5
37                  PvCurrent1: 0.0,                //---- Corriente_PV01
                                              0x0005      6
38                  PvVoltage2: 0.0,                //----    Tension_PV02
                                              0x0004      7
39                  PvCurrent2: 0.0,                //---- Corriente_PV02
                                              0x0006      8
40                  Powerdc1: 0.0,                  //---- Power_PV1
                                                    0x000A      9
```

```
                    Powerdc2: 0.0,                      //---- Power_PV2
                                                        0x000B      10
                    // AC_OUT SYSTEM
                        ------------------------------------------------------------

                    OffGridVoltageX1: 0.0,              //---- V_AOUT EPS X1
                                                    0x004C      11
                    OffGridCurrentX1: 0.0,              //---- I_AOUT EPS X1
                                                    0x004D      12
                    OffGridPowerX1: 0.0,                //---- W_Power  X1
                                                      0x004E      13
                    OffGridFrequencyX1: 0.0,            //---- Hz_Frecuency
                                                     0x004F      14
                    RunMode: 0,                         //----
                        Modo_de_funcionamiento                  0x0009      15
                    //Meter1
                    feedin_power: 0.0,                  //---- In_Out-Power-Meter1
                                                0x0046      16
                    GridReactivePower_Total_Meter1: 0.0,      //IN_n_Out-
                        PowerReactive-Meter1      0x00C0      17
                    GridPowerFactor_Total_Meter1: 0.0,        //FactorPotencia-
                        Meter1                0x00C4      18
                    //Meter1
                    feeding_power_Meter2: 0.0,        // In_Out-Power-Meter2
                                                  0x00A8      19
                    //NO Meter --> Desde la entrada RED/GEN
                    wActivePowerReal: 0.0,            //
                                                                0x0106-107  20
                    wReactivePowerReal: 0.0,          //
                                                                0x0108-109  21
                    Etoday_togrid: 0.0,                 // Energia Diaria a RED
                                                  0x0050      22
                    Etotal_togrid: 0.0,                  // Energia de red total
                                                  0x0052      23
                    temperatura_interna: 0.0,         //
                        TemperaturaInverterEntorno              0x000C //24
                        TemperFaoulValue
                    Tempera_Radiador: 0.0,          // Temperatura_Radiador
                                                 0x0008 //25 temperature
                },
                tags: ['tag1', 'tag2'],
            }
        ],
        bateria: [
            {
                measurement: 'bateria',
                fields: {
                    BatVoltage_Charge1: 0.0, // V_bateria en carga
                                                  0x0014
                    BatCurrent_Charge1: 0.0, // I_bateria en carga
                                                  0x0015
                    Batpower_Charge1: 0.0,   // W_Potencia de carga
                                                  0x0016
                    BMS_Connect_state: 0.0,  // Estado BMS on/off
                                                   0x0017
                    estado_carga_bat: 0.0,            // Battery cargando/
                        descargando             0x0019
                    TemperatureBat: 0.0,     // Temperatura baterry
                                                  0x0018
                    BatteryCapacity: 0.0,    // SOC
                                                              0x001C
                    OutputEnergy_Charge_today: 0.0, // Salida de energia de
                        carga(hoy)          0x0020
```

```
77                      InputputEnergy_Charge_today: 0.0, // Entrada de energia de
                            carga(hoy)          0x0023
78                      wBattery1_Type: 0.0, // Tipo de bateria
                                                        0x008D
79                  },
80                  tags: ['tag1', 'tag2'],
81              }
82          ],
83          eos_array: [
84              {
85                  measurement: 'eos_array',
86                  fields: {
87                      irradiancia_sensor: 0.0,
88                      tension1_panel: 0.0,
89                      corriente1_panel: 0.0,
90                      tension2_panel: 0.0,
91                      corriente2_panel: 0.0,
92                      temp_panel: 0.0,
93                      temp_ambiente:0.0,
94                      canal_7: 0.0
95                  },
96                  tags: ['tag1', 'tag2'],
97              },
98          ],
99          salicru: [
100             {
101                 measurement: 'salicru',
102                 fields: {
103                     ups_status: 0.0,
104                     voltage_bateria: 0.0,
105                     output_voltage: 0.0,
106                     output_current: 0.0,
107                     soc_battery: 0.0,  // SOC de la bateria del SAI-Salicru
108                     comentario: ""
109                 },
110                 tags: ['tag1', 'tag2'],
111             }
112         ],
113
114         hidraulica: [ // ADM_4280_C_I
115             {
116                 measurement: 'hidraulica',
117                 fields: {
118                     Tanque_Nivel_Agua: 0.0,
119                     Bomba1_Caudal: 0.0,
120                     Tanque_Temp_Agua: 0.0,
121                     Bomba_Presio_alto_ch3: 0.0,
122                     Bomba_Presio_bajo_ch4: 0.0,
123                     nivel_pozo_ch5: 0.0,
124                     Valor_Canal_6: 0.0,
125                     Valor_Canal_7: 0.0,
126                 },
127                 tags: ['tag1', 'tag2'],
128             },
129         ],
130
131         adm_4280_v: [
132             {
133                 measurement: 'clima_estacion',
134                 fields: {
135                     irradiancia_estacion: 0.0,
136                     temperatura_amb_estacion: 0.0,
137                     presion_estacion: 0.0,
138                     humedad_estacion: 0.0,
```

```
                    wind_speed_estacion: 0.0,
                },
                tags: ['tag1', 'tag2'],
            }
        ],
        carga: [ // Shelly_4PM
            {
                measurement: 'carga',
                fields: {
                    corriente1: 0.0, // Solo para caso Monofasico
                    corriente2: 0.0, // Solo para caso trifasico
                    corriente3: 0.0, // Solo para caso trifasico
                    tension: 0.0,
                    energia_s1: 0.0, // Solo para Shelly 4PM
                    energia_s2: 0.0,
                    energia_s3: 0.0,
                    energia_s4: 0.0,
                    energia_histo: 0.0
                },
                tags: ['tag1', 'tag2'],
            },
        ],
        fallos_solax_planta: [
            {
                measurement: 'fallos_solax_planta',
                fields: {
                    Mgr_Fault_Msg: 0,// 0x0043 Existe un Fallo en el Inversor, ver tabla 2.5
                    Inv_Fault_Message_LSB: 0,// 0X0040 Fallo del inversor LSB
                    Inv_Fault_Message_MSB: 0,// 0X0041 Fallo del inversor MSB
                    PCSMajorFault: 0,// 0x003E Hay un Fallo en etapa DC/AC
                    TemperFaultValue: 0,// 0X000C Temperatura entorno Inversor
                    Temperature: 0,// 0X0008 Temperatura radiador Inversor
                    RunMode: 0,// 0X0009 Modo de funcionamiento, donde exite fallo general, ver tabla 2.2
                    BatteryMajor_Fault: 0,// 0x003F Fallo BUS DC Battery
                    Bat_BMS_FaultMessage_LSB: 0,//0x0044 Codificacion tabla 2.6
                    Bat_BMS_FaultMessage_MSB: 0,//0x0045 Codificacion tabla 2.7
                    wBatteryVoltFaultVal: 0,//0x0069 Fallo de Vdc Battery
                    Pv1VoltFault: 0,// 0x000D Fallo tension FV1
                    Pv2VoltFault: 0,// 0x000E Fallo tension FV2
                    GfciFaultValue: 0, //0X000F AislamientoDC_Faul_mA
                    GridVoltFaultValue: 0,//0x0010 Fallo de tension de Red
                    DciFaultValue: 0,// 0x0012 Fallo de la corriente en FV
                    GridFreqFaultValueT: 0,// 0x0011 Fallo de Frecuencia de Red
                    //////
                    fault_bomba: 0,  // Si bombea o no (caudalimetro)
                    fault_carga_aux: 0, // Si hay tension en salida y corriente
                    fault_eos_array: 0, // Si hay fallo de comunicacion o dato anomalo
                    fault_ad_4280V: 0,  // Si hay fallo de comunicacion o dato anomalo
                    fault_ad_4280I: 0,  // Si hay fallo de comunicacion o dato anomalo
                },
```

```
                tags: ['tag1', 'tag2'],
            },
        ]
    };

    global.port = 'COM3';

    global.ComOptions = {
        baudRate: 9600,
        dataBits: 8,
        parity: 'none',
        stopBits: 1,
    };

    global.portRS = 'COM1';
    global.ComOptionsRS = {
        baudRate: 2400,
        dataBits: 8,
        parity: 'none',
        stopBits: 1,
    }
    ////////////////////////////////////////
    global.sourceDatabase = 'BBDD_IOM23_TEST';
    global.IP_local = '127.0.0.1';
    global.IP_remota1 = '158.42.96.92';  // Laboratorio GEP
    global.username2 = 'pobuntu'; // Maquina LINUX-UPV
    global.password2 = 'Potencia1';// Maquina LINUX-UPV
    ////////////////////////////////////////////////////////////////
    global.print = function (texto, variable, debug) {
        if (debug == 1) console.log(texto, variable);
    };
    global.dbg = 1;
    // Exportar las funciones para que esten disponibles en otros modulos
    module.exports = {
        counter,
        t_start,
        t_end,
        protocol,
        LeerRS485_RTU,
        client,
        port,
        bool,
        openn,
        IpSolax,
        ComOptions,
        portRS,
        ComOptionsRS,
        boolConect,
        arraySolax,
        arrayEos,
        arrayADM_V,
        arrayADM_I,
        Dato1,
        medidas_arr,
        print,
        dbg,
        sourceDatabase,
        IP_local,
        IP_remota1,
        username2,
        password2,
    };
```

A.2.3. LeerSubsitema_485.js

```
//-------------------Clase LeerSubsistemas_485---------------------
global.Subsist_485 = require('../IndexSubsistemas_485');
const SerialPort = require('serialport').SerialPort;
const { logError, logError_base } = require('./logger_pino'); //LOGGER PINO
// Variables Globales ...
var objectResultx = {};
//-------------------------------------------------------------//
class LeerSubsistema_485 {
    constructor(medidas_arr) {
        this.medidas_arr = medidas_arr;
        this.protocol = protocol;
        this.Openn = true;
        //this.IP = '158.42.81.234';  // Por la UPV
        this.IP = '212.225.205.200'; // celular
        this.portRS1 = 'COM14';
        this.baudRate = 2400;
        this.dataBits = 8;
        this.stopBits = 1;
        this.parity = 'none';
        this.Dato4 = 0;
        this.Dato5 = 0;
        this.Dato6 = 0;
        this.ComOptionsRS = {

            baudRate: 2400,
            dataBits: 8,
            parity: 'none',
            stopBits: 1,
        }
        this.command1 = 'QGS\r';
        this.datos = Buffer.from([]);
        this.bool1 = true;
    }
    ////////////////////////////////////////////////////////////////
    async sendCommand(puerto) {
        try {
            puerto.write(this.command1, (err) => {
                if (err) {
                    logError(`Error al enviar comando: ${err}`);
                } else {
                    console.log('Comando enviado correctamente');
                }
            });
        } catch (error) {
            logError(`Error en sendCommand: ${error.message}`);
        }
    }
    ////////////////////////////////////////////////////////////////
    async parser_salicru(data) {
        const response = data.toString('utf8').trim();
        const values = response.split(' ');
        // Definimos el nombre de los campos a analizar
        const nombreCampos = [
            'InputVoltage', 'InputFrequency', 'OutputVoltage', '
                OutputFrequency', 'OutputCurrent',
            'OutputLoad', 'PositiveBusVoltage', 'NegativeBusVoltage', '
                PBatteryVoltage', 'N-BatteryVoltage', 'Temperature', 'More'
        ];
        // Campos a omitir
        const fieldToOmit = ['OutputFrequency', 'PositiveBusVoltage', '
            NegativeBusVoltage', 'N-BatteryVoltage'];

```

```
60              // Construimos el objeto con el nombre de los campos y sus valores
61              const objectResultx = {};
62              values.forEach((value, indice) => {
63                  const nameField = nombreCampos[indice] || `Campo${indice + 1}`;
64
65                  if (!fieldToOmit.includes(nameField)) {
66                      objectResultx[nameField] = value.trim(); // Con trim(),
                            eliminamos los espacios en blanco
67                  }
68              });
69              // Extraer el valor del campo "More"
70              const moreValue = objectResultx['More'];
71              return (moreValue);
72          }
73          ////////////////////////////////////////////////////////////
74          async parseMoreValue(moreValue) {
75              const parsedData = { // Cada bit corresponde a un campo
76                  OperatingMode: moreValue.slice(0, 2), // 00(standy), 01(
                        lineInteractive) y 10(Online)
77                  UtilityFail: moreValue.slice(2, 3), // 0(Utility OK) y 1(Fail)
78                  BatteryLow: moreValue.slice(3, 4), // 0(Battery OK) y 1(Bateria
                         baja)
79                  BypassBoostActive: moreValue.slice(4, 5), // 0(Boost mode) y 1(
                        Bypass mode)
80                  UPSFailed: moreValue.slice(5, 6), // 0(OK) y 1(UPS Failed)
81                  EPO: moreValue.slice(6, 7),
82                  TestInProgress: moreValue.slice(7, 8),
83                  ShutdownActive: moreValue.slice(8, 9),
84                  BatSilence: moreValue.slice(9, 10),
85                  BatTestFail: moreValue.slice(10, 11),
86                  BatTestOk: moreValue.slice(11, 12)
87              };
88              return parsedData;
89          }
90          ////////////////////////////////////////////////
91          async parseito(data) {
92              let masvalor = await this.parser_salicru(data);
93              let moreData = await this.parseMoreValue(masvalor);
94              // Agregar objeto al objeto resultante
95              const objectFull = Object.assign({}, objectResultx, moreData);
96              console.log(objectFull);
97
98              // Devuelve el objeto con el nombre de los campos y sus respectivos
                     valores
99              return objectResultx;
100         }
101         //-------------------------------------------------------//
102         // LECTURA RS232
103         //-------------------------------------------------------//
104         async LeerSub_232RS(portName, baudRate, dataBits, stopBits, parity) {
105
106             return new Promise((resolve, reject) => {
107                 let datos = Buffer.from([]);
108                 let lastBuffer;
109                 let result;
110                 const port_rs = new SerialPort({
111                     path: portName,
112                     baudRate,
113                     dataBits,
114                     stopBits,
115                     parity
116                 });
117                 port_rs.on('open', () => {
118                     console.log('Conexion con el puerto serial Salicru');
```

```
119                     this.sendCommand(port_rs);
120                 });
121                 port_rs.on('error', (err) => {
122                     console.error('Error en el puerto serial:', err);
123                     if (err.message.includes('Cannot open')) {
124                         console.error('Error: Configuracion incorrecta del
                               puerto serial');
125                     }
126                     reject(err); // Rechazar la promesa en caso de error
127                 });
128                 port_rs.on('data', async (dato) => {
129                     datos = Buffer.concat([datos, dato]);
130                     lastBuffer = datos;
131 
132                     if (dato.includes(Buffer.from('Nak'))) {
133                         console.log('Nak recibido');
134                     }
135                     if (lastBuffer.byteLength > 3 && lastBuffer.includes('\r'))
                           {
136                         try {
137                             result = await this.parseito(lastBuffer);
138                             // Manejar el resultado
139                             for (const field in result) {
140                                 if (result.hasOwnProperty(field)) {
141                                     const valor = result[field];
142                                     console.log(`${field}: ${valor}`);
143                                 }
144                             }
145                             port_rs.close(); // Cerrar la conexion
146                             resolve(result); // Resolver la promesa con el
                                   resultado
147                         } catch (error) {
148                             console.log('Error Lectura del Salicru !!!');
149                             reject(error); // Rechazar la promesa en caso de
                                   error
150                         }
151                     }
152                 });
153             });
154         }
155         //------------------------------------------------------------//
156         // LECTURA 485RTU
157         //------------------------------------------------------------//
158         async LeerSub_485RTU(subsist, Address, Length, cliente, memoryMap,
                parse, Radix, Gain, registro, narrays, esclavo, Description,
                solaxFields) {
159             return new Promise((resolve, reject) => {
160                 cliente.readInputRegisters(Address, Length, (error, data) => {
161                     if (error) {
162                         console.log("Hay problemas de comunicacion");
163                         this.Dato5 = -4;
164                         reject(this.Dato5); // Rechazar la promesa con this.
                               Dato5
165                         return;
166                     }
167                     if (typeof data === 'undefined') {
168                         console.log('El objeto esta indefinido');
169                         this.Dato5 = this.Dato5 - 1;
170                         reject(this.Dato5); // Rechazar la promesa con this.
                               Dato5
171                         return;
172                     }
173                     if (parse) {
174                         let Lectura = this.Parsea_Dato(data, Radix, Gain);
```

```
175                     solaxFields[Description] = Lectura;
176                     console.log("Dato: " + Description + " " + Lectura);
177                     if (parseInt(registro) == narrays.length - 1) {
178                         reject(-4); // Rechazar la promesa con -4
179                     } else {
180                         resolve(this.Dato5); // Resolver la promesa con
                                this.Dato5
181                     }
182                 } else {
183                     resolve(this.Dato5); // Resolver si no hay necesidad de
                            parseo
184                 }
185             });
186         }).catch(err => {
187             if (err.name === 'SerialPortError' && err.message === 'Port is
                    not open') {
188                 console.log('Se perdio la conexion con el puerto serial.');
189             } else {
190                 console.error('Se produjo un error inesperado:', err);
191             }
192             throw err; // Re-lanzar el error para que pueda ser manejado en
                    el codigo que llama
193         });
194     }
195     //-------------------------------------------------------------//
196     undefined(dato) {
197         if (typeof dato === 'undefined') {
198             console.log('El objeto esta indefinido');
199 
200         } else {
201             console.log('El objeto esta definido');
202         }
203     }
204     //-------------------------------------------------------------//
205     // LECTURA 485TCP
206     //-------------------------------------------------------------//
207     async LeerSub_485TCP(subsist, Address, Length, cliente, memoryMap,
            parse, Radix, Gain, registro, narrays, esclavo, Description,
            solaxFields) {
208         if (this.Openn) {
209             this.Openn = false;
210             try {
211                 await cliente.connectTCP(this.IP, { port: 502 });
212                 console.log('1A Conectado el Socket-TCP');
213             } catch (error) {
214                 this.Dato4 = -1;
215                 console.error(error + "Error code: " + this.Dato4);
216                 throw error; // Lanzar el error para ser capturado en la
                        llamada de la funcion
217             }
218         }
219         return new Promise((resolve, reject) => {
220             cliente.readInputRegisters(Address, Length, (error, data) => {
221                 if (error) {
222                     console.error(error + "Hay problemas de comunicacion");
223                     this.Dato4 -= 2;
224                     reject(error);
225                 } else {
226                     if (parse) {
227                         let dat = this.Parsea_Dato(data, Radix, Gain);
228                         solaxFields[Description] = dat;
229                         console.log(dat);
230                     }
231                     resolve(this.Dato4);
```

```
232                     }
233                 });
234             });
235         }
236         //-------------------------------------------------------------//
237         obtenerFechaHoraActual() {
238             const fechaHora = new Date().toLocaleString('es-ES', { timeZone: '
                    Europe/Madrid' });
239             return fechaHora;
240         }
241         //-------------------------------------------------------------//
242         // ACUMULAR DATOS EN ARRAY
243         //-------------------------------------------------------------//
244         lecturaDeDatos(sensor, canal, valor) {
245             // Verificar si el sensor existe en medidas_arr
246             if (this.medidas_arr.hasOwnProperty(sensor)) {
247                 // Obtener el array correspondiente al sensor
248                 var arr = this.medidas_arr[sensor];
249                 // Verificar si el canal existe en el array
250                 if (arr[canal] !== undefined) {
251                     // Actualizar el valor del canal existente
252                     arr[canal] = valor;
253                 } else {
254                     // Agregar un nuevo canal con su valor al array
255                     arr.push({ [canal]: valor });
256                     return arr;
257                 }
258             } else {
259                 // Agregar un nuevo sensor con un canal y su valor al objeto
                        medidas_arr
260                 medidas_arr[sensor] = [{ [canal]: valor }];
261             }
262         }
263         //-------------------------------------------------------------//
264         //OBTENER HORA ACTUAL
265         //-------------------------------------------------------------//
266         obtenerHoraActual() {
267             const hora = new Date().toLocaleTimeString('es-ES');
268             return hora;
269         }
270         //-------------------------------------------------------------//
271         //OBTENER FECHA ACTUAL
272         //-------------------------------------------------------------//
273         obtenerFechaActual() {
274             const fecha = new Date().toLocaleDateString('es-ES');
275             return fecha;
276         }
277         //-------------------------------------------------------------//
278         // PASEAR DATOS
279         //-------------------------------------------------------------//
280         Parsea_Dato(dato, radix, Gain) {
281             var lectura;
282             var offset = 0;
283             let localTimestamp = Date.now();
284             //Verificar si hay que parsear
285             //Calcular el offset del elemento dentro del buffer "data"
286             //Recordando que las funciones buffer manejan solo 8 bits
287             switch (radix) { //Radix: Ver fichero TiposDeDatos.js
288                 ////////////////// big endian ////////////////////////
289                 case 'Uint8_BE':
290                     lectura = data.readUInt8(offset + 1);
291                     break;
292                 case 'Int8_BE':
293                     lectura = data.readInt8(offset + 1); s
```

```
294                     break;
295                 case 'Uint16_BE':
296                     let datos = dato;
297                     lectura = datos.data[0] * Gain;
298                     break;
299                 case 'Int16_BE':
300                     let datos1 = dato;
301                     console.log(datos1.data[0] * Gain);
302                     lectura = datos1.data[0] * Gain;
303                     break;
304                 case 'Uint32_BE':
305                     lectura = Buffer.from([
306                         (data >> 24) & 0xff,
307                         (data >> 16) & 0xff,
308                         (data >> 8) & 0xff,
309                         data & 0xff,
310                     ]);
311                     break;
312                 case 'Int32_BE':
313                     lectura = Buffer.from([
314                         (data >> 24) & 0xff,
315                         (data >> 16) & 0xff,
316                         (data >> 8) & 0xff,
317                         data & 0xff,
318                     ]);
319                     break;
320                 case 'Float32_BE':
321                     const floatBuffer = Buffer.alloc(4);
322                     floatBuffer.writeFloatBE(data);
323                     lectura = floatBuffer;
324                     break;
325                 case 'Inverse_Uint32_BE':
326                     lectura = Buffer.from([data[offset + 2], data[offset + 3],
                            data[offset], data[offset + 1]]).readUInt32BE(0);
327                     break;
328                 case 'Inverse_Int32_BE':
329                     lectura = Buffer.from([data[offset + 2], data[offset + 3],
                            data[offset], data[offset + 1]]).readInt32BE(0);
330                     break;
331                 case 'Inverse_Float32_BE':
332                     lectura = Buffer.from([data[offset + 2], data[offset + 3],
                            data[offset], data[offset + 1]]).readFloatBE(0);
333                     break;
334                 ////////////////// little endian ////////////////////////
335                 case 'Uint8_LE':
336                     lectura = data.readUInt8(offset);
337                     break;
338                 case 'Int8_LE':
339                     lectura = data.readInt8(offset);
340                     break;
341                 case 'Uint16_LE':
342                     lectura = data.readUInt16LE(offset);
343                     break;
344                 case 'Int16_LE':
345                     lectura = data.readInt16LE(offset);
346                     break;
347                 case 'Uint32_LE':
348                     lectura = data.readUInt32LE(offset);
349                     break;
350                 case 'Int32_LE':
351                     lectura = data.readInt32LE(offset);
352                     break;
353                 case 'Float32_LE':
354                     lectura = data.readFloatLE(offset);
```

```
                    break;
                case 'Inverse_Uint32_LE':
                    lectura = data.from([data[offset + 1], data[offset], data[
                        offset + 3], data[offset + 2]]).readUInt32LE(0);
                    break;
                case 'Inverse_Int32_LE':
                    lectura = data.from([data[offset + 1], data[offset], data[
                        offset + 3], data[offset + 2]]).readInt32LE(0);
                    break;
                case 'Inverse_Float32_LE':
                    lectura = data.from([data[offset + 1], data[offset], data[
                        offset + 3], data[offset + 2]]).readFloatLE(0);
                    break;
                default:
                    console.log(" >>> ERROR DE TIPO RADIX !!!");
                    //Alamacenar en "retArray" con el formato de abajo
                    retArray.push({ Description: "Hola", value: lectura,
                        timestamp: localTimestamp });
            }
            return lectura;
        }
    }
    // Exportar a otros modulos ....
    module.exports = LeerSubsistema_485;
```

A.2.4. IndexSubsistemas_485.js

```
// ------------------Clase IndexSubsistema_485----------------------
const SALICRU_MEMORY_MAP = require("./Subsistemas/SALICRU").MemoryMap;
const ADM_4280_I = require("./Subsistemas/ADM-4280-C").MemoryMap;
const EOS_ARRAY_MEMORY_MAP = require("./Subsistemas/EOS_ARRAY_STD").
    MemoryMap;
const SOLAX = require("./Subsistemas/SOLAX_FIN").MemoryMap;
const ADM_4280_V = require("./Subsistemas/ADM-4280-V").MemoryMap;

module.exports = Subsist_485 = [
    {
        MemoryMap: SOLAX,
        ID: 1,
        name: "SOLAX",
        protocol: "MODBUS_TCP",
        reg_type: "hex"

    },
    {
        MemoryMap: ADM_4280_V,
        ID: 2,
        name: "ADM-4280-V",
        protocol: "MODBUS_RTU",
        reg_type: "hex",
        Gain: 0.01
    },
    {
        MemoryMap: EOS_ARRAY_MEMORY_MAP,
        ID: 3,
        name: "EOS-ARRAY",
        protocol: "MODBUS_RTU",
        reg_type: "hex"
    },
    {
        MemoryMap: ADM_4280_I,
        ID: 5,
```

```
            name: "ADM-4280-I",
            protocol: "MODBUS_RTU",
            reg_type: "hex",
            Gain: 0.01
        },
        {
            MemoryMap: SALICRU_MEMORY_MAP,
            ID: 6,
            name: "SALICRU",
            protocol: "RS232",
            reg_type: "ascii"
        }
    ]
    /**
     * @description Funcion que calcula el numero de registros en un conjunto
         de direcciones del mampa de memoria
     * @param {Readonly Object} MEMORY_MAP
     * @return {number} Number of holding registers to be read
     */
    function calcLength(MEMORY_MAP) {
        return MEMORY_MAP[MEMORY_MAP.length - 1].Address - MEMORY_MAP[0].
            Address + MEMORY_MAP[MEMORY_MAP.length - 1].Length; //Direccion
            final - Direccion inicial + logitud ultimo dato -1
    }
    global.calcLength = calcLength;
```

A.2.5. Subsistemas(Mapas de memoria)

- **ADAM(Hidráulica)**

```
            var exports = module.exports = {};
            const tipo = require("../utiles/Tipo_datos/TiposDeDatos").
                Radix;
            exports.MemoryMap = Object.freeze([
                [

                    { RegType: "InputRegister", Description: "
                        Tanque_Nivel_Agua", Address: 0x000, Length: 1,
                        Radix: tipo.BE.Uint16, Gain: 1.0, parse: true,
                        ginflux: false, influx: { measurement: 'ADM-4280-C
                        ', tags: ['Subsistema'], tags_values: ['
                        SensoresHidraulica'], field: "Tanque_Nivel_Agua" }
                         },
                    { RegType: "InputRegister", Description: "
                        Bomba1_Caudal", Address: 0x001, Length: 1, Radix:
                        tipo.BE.Uint16, Gain: 1.0, parse: true, ginflux:
                        false, influx: { measurement: 'ADM-4280-C', tags:
                        ['Subsistema'], tags_values: ['SensoresHidraulica'
                        ], field: "Bomba1_Caudal" } },
                    { RegType: "InputRegister", Description: "
                        Tanque_Temp_Agua", Address: 0x002, Length: 1,
                        Radix: tipo.BE.Int16, Gain: 1.0, parse: true,
                        ginflux: true, influx: { measurement: 'ADM-4280-C'
                        , tags: ['Subsistema'], tags_values: ['
                        SensoresHidraulica'], field: "Tanque_Temp_Agua" }
                        },
                    { RegType: "InputRegister", Description: "
                        Bomba_Presio_alto_ch3", Address: 0x003, Length: 1,
                         Radix: tipo.BE.Int16, Gain: 1.0, parse: true,
                        ginflux: true, influx: { measurement: 'ADM-4280-C'
                        , tags: ['Subsistema'], tags_values: ['
                        SensoresHidraulica'], field: "
                        Bomba_Presio_alto_ch3" } },
```

```
10                  { RegType: "InputRegister", Description: "
                        Bomba_Presio_bajo_ch4", Address: 0x004, Length: 1,
                         Radix: tipo.BE.Int16, Gain: 1.0, parse: true,
                        ginflux: true, influx: { measurement: 'ADM-4280-C'
                        , tags: ['Subsistema'], tags_values: ['
                        SensoresHidraulica'], field: "
                        Bomba_Presio_bajo_ch4" } },
11                  { RegType: "InputRegister", Description: "
                        nivel_pozo_ch5", Address: 0x005, Length: 1, Radix:
                         tipo.BE.Int16, Gain: 1.0, parse: true, ginflux:
                        true, influx: { measurement: 'ADM-4280-C', tags: [
                        'Subsistema'], tags_values: ['SensoresHidraulica'
                        ], field: "nivel_pozo_ch5" } },
12                  { RegType: "InputRegister", Description: "
                        Valor_Canal_6", Address: 0x006, Length: 1, Radix:
                        tipo.BE.Int16, Gain: 1.0, parse: true, ginflux:
                        true, influx: { measurement: 'ADM-4280-C', tags: [
                        'Subsistema'], tags_values: ['SensoresHidraulica'
                        ], field: "Valor_Canal_6" } },
13                  { RegType: "InputRegister", Description: "
                        Valor_Canal_7", Address: 0x007, Length: 1, Radix:
                        tipo.BE.Int16, Gain: 1.0, parse: true, ginflux:
                        true, influx: { measurement: 'ADM-4280-C', tags: [
                        'Subsistema'], tags_values: ['SensoresHidraulica'
                        ], field: "Valor_Canal_7" } }
14              ]
15          ]);
```

- **ADAM(Fotovoltaica)**

```
1       var exports = module.exports = {};
2       const tipo = require("../utiles/Tipo_datos/TiposDeDatos").Radix;
3       exports.MemoryMap = Object.freeze([
4           [
5
6               { RegType: "InputRegister", Description: "
                    irradiancia_estacion", Address: 0x000, Length: 1,
                    Radix: tipo.BE.Uint16, Gain: 0.001, parse: true,
                    ginflux: false, influx: { measurement: 'ADM-4280',
                    tags: ['Subsistema'], tags_values: ['Sensores'], field
                    : "irradiancia_estacion" } },
7               { RegType: "InputRegister", Description: "
                    temperatura_amb_estacion", Address: 0x001, Length: 1,
                    Radix: tipo.BE.Uint16, Gain: 1.0, parse: true, ginflux
                    : false, influx: { measurement: 'ADM-4280', tags: ['
                    Subsistema'], tags_values: ['Sensores'], field: "
                    temperatura_amb_estacion" } },
8               { RegType: "InputRegister", Description: "
                    presion_estacion", Address: 0x002, Length: 1, Radix:
                    tipo.BE.Int16, Gain: 1.0, parse: true, ginflux: true,
                    influx: { measurement: 'ADM-4280', tags: ['Subsistema'
                    ], tags_values: ['Sensores'], field: "presion_estacion
                    " } },
9               { RegType: "InputRegister", Description: "
                    humedad_estacion", Address: 0x003, Length: 1, Radix:
                    tipo.BE.Int16, Gain: 1.0, parse: true, ginflux: true,
                    influx: { measurement: 'ADM-4280', tags: ['Subsistema'
                    ], tags_values: ['Sensores'], field: "humedad_estacion
                    " } },
10              { RegType: "InputRegister", Description: "
                    wind_direc_estacion", Address: 0x004, Length: 1, Radix
                    : tipo.BE.Int16, Gain: 1.0, parse: true, ginflux: true
                    , influx: { measurement: 'ADM-4280', tags: ['
```

```
                    Subsistema'], tags_values: ['Sensores'], field: "
                    wind_direc_estacion" } },
                { RegType: "InputRegister", Description: "
                    wind_speed_estacion", Address: 0x005, Length: 1, Radix
                    : tipo.BE.Int16, Gain: 1.0, parse: true, ginflux: true
                    , influx: { measurement: 'ADM-4280', tags: ['
                    Subsistema'], tags_values: ['Sensores'], field: "
                    wind_speed_estacion" } }
            ]
        ]);
```

- **EOS Array**

```
        var exports = module.exports = {};
        const tipo = require("../utiles/Tipo_datos/TiposDeDatos").Radix;
        exports.MemoryMap = Object.freeze([
            [

            { RegType: "InputRegister", Description: "irradiancia_sensor"
                , Address: 0x30A, Length: 1, Radix: tipo.BE.Uint16, Gain:
                1.0, parse: true, ginflux: true, influx: { measurement: '
                EOS ARRAY', tags: ['Subsistema'], tags_values: ['PV_I_1'],
                 field: "irradiancia_sensor" } },
            { RegType: "InputRegister", Description: "tension1_panel",
                Address: 0x309, Length: 1, Radix: tipo.BE.Uint16, Gain:
                1.0, parse: true, ginflux: true, influx: { measurement: '
                EOS ARRAY', tags: ['Subsistema'], tags_values: ['PV_V_1'],
                 field: "tension1_panel" } },
                { RegType: "InputRegister", Description: "
                    corriente1_panel", Address: 0x30B, Length: 1, Radix:
                    tipo.BE.Uint16, Gain: 1.0, parse: true, ginflux: true,
                     influx: { measurement: 'EOS ARRAY', tags: ['
                    Subsistema'], tags_values: ['PV_W_1'], field: "
                    corriente1_panel" } },
                { RegType: "InputRegister", Description: "tension2_panel"
                    , Address: 0x30C, Length: 1, Radix: tipo.BE.Uint16,
                    Gain: 1.0, parse: true, ginflux: true, influx: {
                    measurement: 'EOS ARRAY', tags: ['Subsistema'],
                    tags_values: ['PV_Ef_1'], field: "tension2_panel" } },
                { RegType: "InputRegister", Description: "
                    corriente2_panel", Address: 0x30D, Length: 1, Radix:
                    tipo.BE.Uint16, Gain: 1.0, parse: true, ginflux: true,
                     influx: { measurement: 'EOS ARRAY', tags: ['
                    Subsistema'], tags_values: ['PV_E_1'], field: "
                    corriente2_panel" } },
                { RegType: "InputRegister", Description: "temp_panel",
                    Address: 0x311, Length: 1, Radix: tipo.BE.Uint16, Gain
                    : 1.0, parse: true, ginflux: true, influx: {
                    measurement: 'EOS ARRAY', tags: ['Subsistema'],
                    tags_values: ['PV_V_2'], field: "temp_panel" } },
                { RegType: "InputRegister", Description: "temp_ambiente",
                     Address: 0x312, Length: 1, Radix: tipo.BE.Uint16,
                    Gain: 1.0, parse: true, ginflux: true, influx: {
                    measurement: 'EOS ARRAY', tags: ['Subsistema'],
                    tags_values: ['PV_I_2'], field: "temp_ambiente" } },
                { RegType: "InputRegister", Description: "canal_7",
                    Address: 0x313, Length: 1, Radix: tipo.BE.Uint16, Gain
                    : 1.0, parse: true, ginflux: true, influx: {
                    measurement: 'EOS ARRAY', tags: ['Subsistema'],
                    tags_values: ['PV_W_2'], field: "canal_7" } },
            ]
        ]);
```

- **Solax**

```
var exports = module.exports = {};
const tipo = require("../utiles/Tipo_datos/TiposDeDatos").Radix;
exports.MemoryMap = Object.freeze([
[
 /// AC_IN GRID O AUX GEN
 { RegType: "InputRegister", Description: "GridVoltage", Address: 0x000, Length: 1, Radix: tipo.BE.Uint16, Gain: 0.1, parse: true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Red In'], field: "GridVoltage" } },
 { RegType: "InputRegister", Description: "GridCurrent", Address: 0x001, Length: 1, Radix: tipo.BE.Int16, Gain: 0.1, parse: true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Red In'], field: "GridCurrent" } },
 { RegType: "InputRegister", Description: "GridPower", Address: 0x002, Length: 1, Radix: tipo.BE.Int16, Gain: 1, parse: true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Red In'], field: "GridPower" } },
 { RegType: "InputRegister", Description: "GridFrequency", Address: 0x007, Length: 1, Radix: tipo.BE.Uint16, Gain: 0.01, parse: true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Red In'], field: "GridFrequency" } },
 // PV System
 { RegType: "InputRegister", Description: "PvVoltage1", Address: 0x003, Length: 1, Radix: tipo.BE.Uint16, Gain: 0.1, parse: true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Campo_Solar'], field: "PvVoltage1" } },
 { RegType: "InputRegister", Description: "PvCurrent1", Address: 0x005, Length: 1, Radix: tipo.BE.Uint16, Gain: 1, parse: true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Campo_Solar'], field: "PvCurrent1" } },
 { RegType: "InputRegister", Description: "PvVoltage2", Address: 0x004, Length: 1, Radix: tipo.BE.Uint16, Gain: 0.1, parse: true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Campo_Solar'], field: "PvVoltage2" } },
 { RegType: "InputRegister", Description: "PvCurrent2", Address: 0x006, Length: 1, Radix: tipo.BE.Uint16, Gain: 1, parse: true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Campo_Solar'], field: "PvCurrent2" } },
 { RegType: "InputRegister", Description: "Powerdc1", Address: 0x00A, Length: 1, Radix: tipo.BE.Uint16, Gain: 1, parse: true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Campo_Solar'], field: "Powerdc1" } },
 { RegType: "InputRegister", Description: "Powerdc2", Address: 0x00B, Length: 1, Radix: tipo.BE.Uint16, Gain: 1, parse: true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Campo_Solar'], field: "Powerdc2" } },
 // AC_OUT INVERTER
 { RegType: "InputRegister", Description: "OffGridVoltageX1", Address: 0x04C, Length: 1, Radix: tipo.BE.Uint16, Gain: 0.1, parse: true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['EPS'], field: "OffGridVoltageX1" } },
```

```
19          { RegType: "InputRegister", Description: "OffGridCurrentX1",
                Address: 0x04D, Length: 1, Radix: tipo.BE.Uint16, Gain: 0.1,
                parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                SOLAX', tags: ['Subsistema'], tags_values: ['EPS'], field: "
                OffGridCurrentX1" } },
20          { RegType: "InputRegister", Description: "OffGridPowerX1",
                Address: 0x04E, Length: 1, Radix: tipo.BE.Uint16, Gain: 1,
                parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                SOLAX', tags: ['Subsistema'], tags_values: ['EPS'], field: "
                OffGridPowerX1" } },
21          { RegType: "InputRegister", Description: "OffGridFrequencyX1",
                Address: 0x04F, Length: 1, Radix: tipo.BE.Uint16, Gain: 0.01,
                 parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                 SOLAX', tags: ['Subsistema'], tags_values: ['EPS'], field: "
                OffGridFrequencyX1" } },
22          { RegType: "InputRegister", Description: "RunMode", Address: 0
                x009, Length: 1, Radix: tipo.BE.Uint16, Gain: 1, parse: true,
                 ginflux: false, influx: { measurement: 'INVERSOR SOLAX',
                tags: ['Subsistema'], tags_values: ['Cfg SOLAX'], field: "
                RunMode" } },
23          // // // Merter 1
24          { RegType: "InputRegister", Description: "feedin_power", Address
                : 0x046, Length: 2, Radix: tipo.BE.Uint16, Gain: 1, parse:
                true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX',
                 tags: ['Subsistema'], tags_values: ['AC_GRID_Meter'], field:
                 "feedin_power" } },
25          { RegType: "InputRegister", Description: "
                GridReactivePower_Total_Meter1", Address: 0x0C0, Length: 1,
                Radix: tipo.BE.Uint16, Gain: 1, parse: true, ginflux: true,
                influx: { measurement: 'INVERSOR SOLAX', tags: ['Subsistema'
                ], tags_values: ['AC_GRID_Meter'], field: "
                GridReactivePower_Total_Meter1" } },
26          { RegType: "InputRegister", Description: "
                GridPowerFactor_Total_Meter1", Address: 0x0C4, Length: 1,
                Radix: tipo.BE.Uint16, Gain: 1, parse: true, ginflux: true,
                influx: { measurement: 'INVERSOR SOLAX', tags: ['Subsistema'
                ], tags_values: ['AC_GRID_Meter'], field: "
                GridPowerFactor_Total_Meter1" } },
27          // // // Merter 2
28          { RegType: "InputRegister", Description: "feeding_power_Meter2",
                 Address: 0x0A8, Length: 2, Radix: tipo.BE.Uint16, Gain: 1,
                parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                SOLAX', tags: ['Subsistema'], tags_values: ['AC_GRID_Meter'],
                 field: "feeding_power_Meter2" } },
29          { RegType: "InputRegister", Description: "wActivePowerReal",
                Address: 0x0106, Length: 2, Radix: tipo.BE.Uint16, Gain: 1,
                parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                SOLAX', tags: ['Subsistema'], tags_values: ['AC_INVERTER'],
                field: "wActivePowerReal" } },
30          { RegType: "InputRegister", Description: "wReactivePowerReal",
                Address: 0x0108, Length: 2, Radix: tipo.BE.Uint16, Gain: 1,
                parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                SOLAX', tags: ['Subsistema'], tags_values: ['AC_INVERTER'],
                field: "wReactivePowerReal" } },
31          { RegType: "InputRegister", Description: "Etoday_togrid",
                Address: 0x050, Length: 1, Radix: tipo.BE.Uint16, Gain: 0.1,
                parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                SOLAX', tags: ['Subsistema'], tags_values: ['Red In'], field:
                 "Etoday_togrid" } },
32          { RegType: "InputRegister", Description: "Etotal_togrid",
                Address: 0x052, Length: 2, Radix: tipo.BE.Uint16, Gain: 0.1,
                parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                SOLAX', tags: ['Subsistema'], tags_values: ['Red In'], field:
                 "Etotal_togrid" } },
```

```
33          { RegType: "InputRegister", Description: "temperatura_interna",
                Address: 0x00C, Length: 1, Radix: tipo.BE.Uint16, Gain: 1,
                parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                SOLAX', tags: ['Subsistema'], tags_values: ['Inversor_Solax'
                ], field: "temperatura_interna" } },
34          { RegType: "InputRegister", Description: "Tempera_Radiador",
                Address: 0x008, Length: 1, Radix: tipo.BE.Int16, Gain: 1,
                parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                SOLAX', tags: ['Subsistema'], tags_values: ['Inversor_SOLAX'
                ], field: "Tempera_Radiador" } },
35          //----------------------------BATERIA-------------------------
36          { RegType: "InputRegister", Description: "BatVoltage_Charge1",
              Address: 0x014, Length: 1, Radix: tipo.BE.Int16, Gain: 0.1,
              offset: 0, parse: true, ginflux: true, influx: { measurement:
              'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Bateria
              '], field: "BatVoltage_Charge1" } },
37          { RegType: "InputRegister", Description: "BatCurrent_Charge1",
              Address: 0x015, Length: 1, Radix: tipo.BE.Int16, Gain: 0.1,
              offset: 0, parse: true, ginflux: true, influx: { measurement:
              'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Bateria
              '], field: "BatCurrent_Charge1" } },
38          { RegType: "InputRegister", Description: "Batpower_Charge1",
              Address: 0x016, Length: 1, Radix: tipo.BE.Int16, Gain: 1.0,
              offset: 0, parse: true, ginflux: true, influx: { measurement:
              'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Bateria
              '], field: "Batpower_Charge1" } },
39          { RegType: "InputRegister", Description: "BMS_Connect_state",
              Address: 0x017, Length: 1, Radix: tipo.BE.Uint16, Gain: 1.0,
              offset:0, parse: true, ginflux: true, influx: { measurement: '
              INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['
              Estado_SOLAX'], field: "BMS_Connect_state" } }, // (
              Desconectado, Conectado)
40          { RegType: "InputRegister", Description: "TemperatureBat",
              Address: 0x018, Length: 1, Radix: tipo.BE.Uint16, Gain: 1.0,
              offset: 0, parse: true, ginflux: true, influx: { measurement:
              'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Bateria
              '], field: "TemperatureBat" } },
41          { RegType: "InputRegister", Description: "estado_carga_bat",
              Address: 0x019, Length: 1, Radix: tipo.BE.Uint16, Gain: 1.0,
              offset: 0, parse: true, ginflux: true, influx: { measurement:
              'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Bateria
              '], field: "estado_carga_bat" } },
42          { RegType: "InputRegister", Description: "BatteryCapacity",
              Address: 0x01C, Length: 1, Radix: tipo.BE.Uint16, Gain: 0.01,
              parse: true, ginflux: true, influx: { measurement: 'INVERSOR
              SOLAX', tags: ['Subsistema'], tags_values: ['Bateria'], field:
               "BatteryCapacity" } },
43          { RegType: "InputRegister", Description: "
              OutputEnergy_Charge_today", Address: 0x020, Length: 1, Radix:
              tipo.BE.Uint16, Gain: 0.1, parse: true, ginflux: true, influx:
               { measurement: 'INVERSOR SOLAX', tags: ['Subsistema'],
              tags_values: ['Bateria'], field: "OutputEnergy_Charge_today" }
               },
44          { RegType: "InputRegister", Description: "
              InputputEnergy_Charge_today", Address: 0x023, Length: 1, Radix
              : tipo.BE.Uint16, Gain: 0.1, parse: true, ginflux: true,
              influx: { measurement: 'INVERSOR SOLAX', tags: ['Subsistema'],
               tags_values: ['Bateria'], field: "InputputEnergy_Charge_today
              " } },
45          { RegType: "HoldingRegister", Description: "wBattery1_Type",
              Address: 0x008D, Length: 1, Radix: tipo.BE.Uint16, Gain: 1,
              parse: true, ginflux: true, influx: { measurement: 'INVERSOR
              SOLAX', tags: ['Subsistema'], tags_values: ['Bateria'], field:
               "wBattery1_Type" } },
```

```
46          // Faults -->
47          { RegType: "InputRegister", Description: "Mgr_Fault_Msg", Address
                : 0x043, Length: 1, Radix: tipo.BE.Uint16, Gain: 1, parse:
                true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX',
                tags: ['Subsistema'], tags_values: ['Inversor_Solax'], field:
                "Mgr_Fault_Msg" } },

                // CUAL ES EL FAULT?
48          { RegType: "InputRegister", Description: "Inv_Fault_Message_LSB",
                 Address: 0x040, Length: 2, Radix: tipo.BE.Uint16, Gain: 1,
                parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                SOLAX', tags: ['Subsistema'], tags_values: ['Inverter_SOLAX'],
                 field: "Inv_Fault_Message_LSB" } },
49                          //FALLO GRAVE EN DC/AC
50          { RegType: "InputRegister", Description: "PCSMajorFault", Address
                : 0x03E, Length: 1, Radix: tipo.BE.Uint16, Gain: 1, parse:
                true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX',
                tags: ['Subsistema'], tags_values: ['Inversor_SOLAX DC-AC'],
                field: "PCSMajorFault" } },
51                          // TEMPERATURA INVERTER
52          { RegType: "InputRegister", Description: "TemperFaultValue",
                Address: 0x00C, Length: 1, Radix: tipo.BE.Uint16, Gain: 1,
                parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                SOLAX', tags: ['Subsistema'], tags_values: ['Inversor_Solax'],
                 field: "TemperFaultValue" } },
53          { RegType: "InputRegister", Description: "Temperature", Address:
                0x008, Length: 1, Radix: tipo.BE.Int16, Gain: 1, parse: true,
                ginflux: true, influx: { measurement: 'INVERSOR SOLAX', tags:
                ['Subsistema'], tags_values: ['Inversor_SOLAX'], field: "
                Temperature" } },
54          { RegType: "InputRegister", Description: "RunMode", Address: 0
                x009, Length: 1, Radix: tipo.BE.Uint16, Gain: 1, parse: true,
                ginflux: false, influx: { measurement: 'INVERSOR SOLAX', tags:
                 ['Subsistema'], tags_values: ['Cfg SOLAX'], field: "RunMode"
                } },
55          { RegType: "InputRegister", Description: "BatteryMajor_Fault",
                Address: 0x03F, Length: 1, Radix: tipo.BE.Uint16, Gain: 1,
                parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                SOLAX', tags: ['Subsistema'], tags_values: ['Bateria_SOLAX'],
                field: "BatteryMajor_Fault" } },
56                          // CODIGOS DE FALLOS BATERIA
57          { RegType: "InputRegister", Description: "
                Bat_BMS_FaultMessage_LSB", Address: 0x044, Length: 2, Radix:
                tipo.BE.Uint16, Gain: 1, parse: true, ginflux: true, influx: {
                 measurement: 'INVERSOR SOLAX', tags: ['Subsistema'],
                tags_values: ['Bateria_SOLAX'], field: "
                Bat_BMS_FaultMessage_LSB" } },
58          { RegType: "InputRegister", Description: "wBatteryVoltFaultVal",
                Address: 0x0069, Length: 1, Radix: tipo.BE.Uint16, Gain: 0.1,
                offset: 0, parse: true, ginflux: true, influx: { measurement:
                'INVERSOR SOLAX', tags: ['Subsistema'], tags_values: ['Bateria
                '], field: "wBatteryVoltFaultVal" } },
59          { RegType: "InputRegister", Description: "Pv1VoltFault", Address:
                 0x00D, Length: 1, Radix: tipo.BE.Uint16, Gain: 1, parse: true
                , ginflux: true, influx: { measurement: 'INVERSOR SOLAX', tags
                : ['Subsistema'], tags_values: ['Inversor_Solax_FV'], field: "
                Pv1VoltFault" } },
60          { RegType: "InputRegister", Description: "Pv2VoltFault", Address:
                 0x00E, Length: 1, Radix: tipo.BE.Uint16, Gain: 1, parse: true
                , ginflux: true, influx: { measurement: 'INVERSOR SOLAX', tags
                : ['Subsistema'], tags_values: ['Inversor_Solax_Fv'], field: "
                Pv2VoltFault" } },
61                          // FALLO AISLAMIENTO DC
```

```
62          { RegType: "InputRegister", Description: "GfciFaultValue",
                Address: 0x00F, Length: 1, Radix: tipo.BE.Uint16, Gain: 1,
                parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                SOLAX', tags: ['Subsistema'], tags_values: ['Inversor_Solax_Fv
                '], field: "GfciFaultValue" } },
63                              // FALLO TENSION BATERIAS
64          { RegType: "InputRegister", Description: "GridVoltFaultValue",
                Address: 0x010, Length: 1, Radix: tipo.BE.Uint16, Gain: 0.1,
                parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                SOLAX', tags: ['Subsistema'], tags_values: ['
                Inversor_Solax_Grid'], field: "GridVoltFaultValue" } },
65                              // FALLO CORRIENTE FV
66          { RegType: "InputRegister", Description: "DciFaultValue", Address
                : 0x012, Length: 1, Radix: tipo.BE.Uint16, Gain: 1, parse:
                true, ginflux: true, influx: { measurement: 'INVERSOR SOLAX',
                tags: ['Subsistema'], tags_values: ['Inversor_Solax_PV'],
                field: "DciFaultValue" } },
67          { RegType: "InputRegister", Description: "GridFreqFaultValueT",
                Address: 0x011, Length: 1, Radix: tipo.BE.Uint16, Gain: 0.01,
                parse: true, ginflux: true, influx: { measurement: 'INVERSOR
                SOLAX', tags: ['Subsistema'], tags_values: ['
                Inversor_Solax_Grid'], field: "GridFreqFaultValueT" } }
68          ]
69          ]);
```

- **Salicru**

```
1           var exports = module.exports = {};
2           const tipo = require("../utiles/Tipo_datos/TiposDeDatos").Radix;
3           exports.MemoryMap = Object.freeze([
4               [
5                   { RegType: "RS232Register", Description: "Status",
                        command: "QGS\r", Length: 1, Radix: tipo.BE.Uint16,
                        Gain: 1.0, parse: true, ginflux: false, influx: {
                        measurement: 'SALICRU', tags: ['Subsistema'],
                        tags_values: ['Sai_SALICRU'], field: "General Status
                        Parameters" } },
6                   { RegType: "RS232Register", Description: "Faults",
                        command: "QFS\r", Length: 1, Radix: tipo.BE.Uint16,
                        Gain: 1.0, parse: true, ginflux: false, influx: {
                        measurement: 'SALICRU', tags: ['Subsistema'],
                        tags_values: ['Sai_SALICRU'], field: "Fault Status" }
                         },
7                   { RegType: "RS232Register", Description: "Battery",
                        Address: QBTAH, Length: 1, Radix: tipo.BE.Uint16,
                        Gain: 1.0, parse: true, ginflux: false, influx: {
                        measurement: 'SALICRU', tags: ['Subsistema'],
                        tags_values: ['Sai_SALICRU'], field: "Battery
                        Information" } }
8               ]
9           ]);
```

A.2.6. utiles.js

```
////////////////////////////////////////////////////////////////
// FICHERO: utiles.js
////////////////////////////////////////////////////////////////
const fs = require('fs');
const va = require("./GlobalClass");
const { logError, logError_base } = require('./logger_pino'); //LOGGER PINO
//const path = require('path');
////////////////////////////////////////////////////////////////
// Clase de Utilidades para la Aplicacion ...
class Utiles {
    static nombreArchivo = ''; // Variables globales dentro de la Clase ....
    static rutaArchivo = '';
    static async retardo(num_ms) {
        const startTime = Date.now();
        while (Date.now() - startTime < num_ms) {
            // Esperar hasta que se cumpla el tiempo especificado
        }
    }
    ////////////////////////////////////////////////////////////////
    // Generar valores aleatorios entre un valor minimo y maximo
    static getRandomValue(min, max) {
        return (Math.random() * (max - min) + min).toFixed(3);
    }
    ////////////////////////////////////////////////////////////////
    static obtenerFechaActual() {
        const zonaHoraria = 'Europe/Madrid';
        const opcionesFecha = { timeZone: zonaHoraria };
        const fechaString = new Date().toLocaleString('en-US', opcionesFecha);
        //console.log("FECHA zona ---> ", fechaFormateada);
        return fechaString;
    }
    ////////////////////////////////////////////////////////////////
    // Obtener la fecha y hora actual en la zona horaria de Madrid con Time Zone
    static obtenerFecha_Madrid(tipo) {
        const currentDate = new Date();
        const madridTimeZoneOffset = 120; // Offset en minutos para la zona horaria de Madrid
        const fechaMadrid = new Date(currentDate.getTime() + madridTimeZoneOffset * 60000);
        if (tipo == 1) return fechaMadrid.toISOString()
        else return (fechaMadrid);
    }
////////////////////////////////////////////////////////////////
    static verificarSubcadena(texto, subcadena) {
        if (texto.includes(subcadena)) {
            return true;
        } else {
            return false;
        }
    }
    ////////////////////////////////////////////////////////////////
    static obtenerNombre() {
        const fechaActual = new Date();
        const dia = fechaActual.getDate();
        const mes = fechaActual.getMonth() + 1; // Los meses empiezan en 0
        const anio = fechaActual.getFullYear();
        const nombrefile = `${dia}-${mes < 10 ? '0' + mes : mes}-${anio}_datos_IOM.json`;
        return nombrefile;
```

```
57          }
58          ////////////////////////////////////////////////////////////////
59          static obtenerDiaDesdeNombre(nombre) {
60              const partes = nombre.split('_');
61              const fechaPartes = partes[0].split('-');
62              return parseInt(fechaPartes[0]);
63          }
64          ////////////////////////////////////////////////////////////////
65          static verificarCambioDeDia(nom_file) {
66              const fechaActual = new Date();
67              const diaActual = fechaActual.getDate();
68              let diafile = this.obtenerDiaDesdeNombre(nom_file);
69              if (diaActual !== diafile) {
70                  Utiles.nombreArchivo = this.obtenerNombre();
71                  Utiles.rutaArchivo = "./Storage/" + Utiles.nombreArchivo;
72              }
73              // Calcular el tiempo restante hasta las 23:59:59
74              const tiempoHastaUltimoSegundo = new Date(
75                  fechaActual.getFullYear(),
76                  fechaActual.getMonth(),
77                  fechaActual.getDate(),
78                  23,
79                  59,
80                  59
81              ) - fechaActual;
82              setTimeout(this.verificarCambioDeDia, tiempoHastaUltimoSegundo);
83              //setTimeout(() => this.verificarCambioDeDia(Utiles.nombreArchivo),
                    tiempoHastaUltimoSegundo);
84          }
85          ////////////////////////////////////////////////////////////////
86          // Funcion Escribir los Datos Leidos de cada subsistema en un Fichero
                JSON ...
87          static async obtener_ruta_cambio_dia() {
88              Utiles.nombreArchivo = this.obtenerNombre();
89              //const rutaArchivo = path.join(__dirname, nombreArchivo);
90              Utiles.rutaArchivo = "./Storage/" + Utiles.nombreArchivo;
91
92              console.log("Nombre Fichero JSON ---> ", Utiles.nombreArchivo);
93              console.log("Path + Nombre Fichero JSON ---> ", Utiles.rutaArchivo)
                    ;
94              // Iniciar la verificacion del cambio de dia
95              this.verificarCambioDeDia(Utiles.nombreArchivo);
96          }
97          ////////////////////////////////////////////////////////////////
98          // Funcion Escribir los Datos Leidos de cada subsistema en un Fichero
                JSON ...
99          static async loopEscrituturaJSON() {
100             try {
101                 var datosW
102                 const currentDate = new Date();
103                 const madridTimeZoneOffset = 120; // Offset en minutos para la
                        zona horaria de Madrid
104                 const fechaMadrid = new Date(
105                     currentDate.getTime() + madridTimeZoneOffset * 60000
106                 );
107                 const data = {
108                     fecha: fechaMadrid.toISOString(),
109                     medidas: va.medidas_arr,
110                 };
111                 datosW = data;
112
113                 await this.obtener_ruta_cambio_dia();
114                 // Convertir el objeto a formato JSON
115                 const jsonData = JSON.stringify(datosW, null, 2);
```

```
116                 // Verificar si existe el Fichero *. JSON
117                 const fileExists = fs.existsSync(Utiles.rutaArchivo);
118                 if (!fileExists) {
119                     // Si el archivo no existe, escribir el JSON inicial
120                     fs.writeFileSync(Utiles.rutaArchivo, `[${jsonData}]`);
121                 } else {
122                     // Si el archivo existe, leer su contenido y agregar el
                        nuevo JSON
123                     const existingData = fs.readFileSync(
124                         //"./Storage/datos_IOM_485.json",
125                         Utiles.rutaArchivo,
126                         "utf8"
127                     );
128                     const modifiedData = `${existingData.slice(0, -1)},\n${
                        jsonData}]`;
129                     fs.writeFileSync(Utiles.rutaArchivo, modifiedData);
130                 }
131                 console.log("ESCRITURA REALIZADA");
132                 console.log(datosW);
133                 // Cerrar el archivo
134                 fs.closeSync(fs.openSync(Utiles.rutaArchivo, 'r'));
135             } catch (error) {
136                 console.error(`Error de loopEscrituturaJSON ???`, error.message
                    );
137                 logError(0, error);
138                 throw error;
139             }
140         }
141         ////////////////////////////////////////////////////////////////
142         static async calc_soc_salicru(val_tension) {
143             // Datos de la tabla tension SOC de la bateria de Salicru
144             const tension = [52.8, 51.2, 49.6, 48.0, 46.4, 44.8, 43.2, 41.6,
                    40.0, 38.4, 36.8];
145             const soc = [100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 0];
146 
147             // Suma de los valores de las tensiones y del SOC
148             let sumTension = 0;
149             let sumSOC = 0;
150 
151             for (let i = 0; i < tension.length; i++) {
152                 sumTension += tension[i];
153                 sumSOC += soc[i];
154             }
155             // Calculo de los promedios
156             const avgTension = sumTension / tension.length;
157             const avgSOC = sumSOC / soc.length;
158 
159             // Calculo de los coeficientes de la ecuacion lineal
160             let numerator = 0;
161             let denominator = 0;
162 
163             for (let i = 0; i < tension.length; i++) {
164                 numerator += (tension[i] - avgTension) * (soc[i] - avgSOC);
165                 denominator += Math.pow(tension[i] - avgTension, 2);
166             }
167             const slope = numerator / denominator;
168             const intercept = avgSOC - slope * avgTension;
169             // Ecuacion lineal: SOC = pendiente * Tension + interseccion
170             console.log(`Ecuacion lineal: SOC = ${slope.toFixed(2)} * Tension +
                     ${intercept.toFixed(2)}`);
171             let SOC_bateria = val_tension * slope + intercept;
172              // Verificacion de que el valor SOC este dentro de los limites
                    esperados
173             //  if (SOC_bateria > 100) SOC_bateria = 100;
```

```
174             //  if (SOC_bateria < 0) SOC_bateria = 0;
175             return (SOC_bateria);
176         }
177     } // Fin de Clase Utiles ...
178     module.exports = Utiles;
```

A.2.7. Storage(Influx_Util.js)

```
1     ////////////////////////////////////////////////
2     // FICHERO: Influx_Util.js
3     ////////////////////////////////////////////////
4     const axios = require('axios');
5     const Influx = require('influx');
6     const va = require('./GlobalClass');
7     const { logError, logError_base } = require('./logger_pino'); //LOGGER PINO
8     // Configuracion de la conexion a InfluxDB
9     const influx = new Influx.InfluxDB({
10        host: va.IP_local,
11        port: 8086,
12        database: va.sourceDatabase,
13        username: '',
14        password: '',
15
16        // Definicion del esquema de la BBDD-INFLUX
17        schema: [
18            {
19                measurement: 'inv_solax',
20                fields: {
21                    /// AC_IN GRID O AUX GEN
22                    GridVoltage: Influx.FieldType.FLOAT,
23                    GridCurrent: Influx.FieldType.FLOAT,
24                    GridPower: Influx.FieldType.FLOAT,
25                    GridFrequency: Influx.FieldType.FLOAT,
26                    // PV System...  DC-IN
27                    PvVoltage1: Influx.FieldType.FLOAT,
28                    PvCurrent1: Influx.FieldType.FLOAT,
29                    PvVoltage2: Influx.FieldType.FLOAT,
30                    PvCurrent2: Influx.FieldType.FLOAT,
31                    Powerdc1: Influx.FieldType.FLOAT,
32                    Powerdc2: Influx.FieldType.FLOAT,
33                    /// AC_OUT INVERTER
34                    OffGridVoltageX1: Influx.FieldType.FLOAT,
35                    OffGridCurrentX1: Influx.FieldType.FLOAT,
36                    OffGridPowerX1: Influx.FieldType.FLOAT,
37                    OffGridFrequencyX1: Influx.FieldType.FLOAT,
38                    RunMode: Influx.FieldType.INTEGER,
39                    // Meter1
40                    feedin_power: Influx.FieldType.FLOAT,
41                    GridReactivePower_Total_Meter1: Influx.FieldType.FLOAT,
42                    GridPowerFactor_Total_Meter1: Influx.FieldType.FLOAT,
43                    // Meter2
44                    feeding_power_Meter2: Influx.FieldType.FLOAT,
45                    //NO Meter --> Desde la entrada RED/GEN
46                    wActivePowerReal: Influx.FieldType.FLOAT,
47                    wReactivePowerReal: Influx.FieldType.FLOAT,
48                    Etoday_togrid: Influx.FieldType.FLOAT,  //0x0050 Energia
                          Diaria a RED
49                    Etotal_togrid: Influx.FieldType.FLOAT,  //0x0052/53 Energia
                          total a RED
50                    temperatura_interna: Influx.FieldType.FLOAT,
51                    Tempera_Radiador: Influx.FieldType.FLOAT,
52                },
```

```
53                  tags: ['tag1', 'tag2'],
54              },
55              {
56                  measurement: 'bateria', // Bateria SOLAX
57                  fields: {
58                      BatVoltage_Charge1: Influx.FieldType.FLOAT,
59                      BatCurrent_Charge1: Influx.FieldType.FLOAT,
60                      Batpower_Charge1: Influx.FieldType.FLOAT,
61                      BMS_Connect_state: Influx.FieldType.FLOAT,
62                      TemperatureBat: Influx.FieldType.FLOAT,
63                      BatteryCapacity: Influx.FieldType.FLOAT,
64                      OutputEnergy_Charge_today: Influx.FieldType.FLOAT,
65                      InputputEnergy_Charge_today: Influx.FieldType.FLOAT,
66                      wBattery1_Type: Influx.FieldType.STRING,
67                  },
68                  tags: ['tag1', 'tag2'],
69              },
70              {
71                  measurement: 'eos_array',
72                  fields: {
73                      irradiancia_sensor: Influx.FieldType.FLOAT,
74                      tension1_panel: Influx.FieldType.FLOAT,
75                      corriente1_panel: Influx.FieldType.FLOAT,
76                      tension2_panel: Influx.FieldType.FLOAT,
77                      corriente2_panel: Influx.FieldType.FLOAT,
78                      temp_panel: Influx.FieldType.FLOAT,
79                      temp_ambiente: Influx.FieldType.FLOAT,
80                      canal_7: Influx.FieldType.FLOAT,
81                  },
82                  tags: ['tag1', 'tag2'],
83              },
84              {
85                  measurement: 'salicru',
86                  fields: {
87                      ups_status: Influx.FieldType.INTEGER,
88                      voltage_bateria: Influx.FieldType.FLOAT,
89                      output_voltage: Influx.FieldType.FLOAT,
90                      output_current: Influx.FieldType.FLOAT,
91                      soc_battery: Influx.FieldType.INTEGER,  // SOC de la
                            bateria del SAI-Salicru
92                      comentario: Influx.FieldType.STRING,
93                  },
94                  tags: ['tag1', 'tag2'],
95              },
96              {
97                  measurement: 'hidraulica',  // Conversor AD-4280-I
98                  fields: {
99                      Tanque_Nivel_Agua: Influx.FieldType.FLOAT,
100                     Bomba1_Caudal: Influx.FieldType.FLOAT,    // Litros/seg
101                     Tanque_Temp_Agua: Influx.FieldType.FLOAT,
102                     Bomba_Presio_alto_ch3: Influx.FieldType.FLOAT,  // Presion
                            nivel alto de la Bomba
103                     Bomba_Presio_bajo_ch4: Influx.FieldType.FLOAT, // Presion
                            nivel BAJOalto de la Bomba
104                     nivel_pozo_ch5: Influx.FieldType.FLOAT,
105                     Valor_Canal_6: Influx.FieldType.FLOAT,
106                     Valor_Canal_7: Influx.FieldType.FLOAT,
107                 },
108                 tags: ['tag1', 'tag2'],
109             },
110             {
111                 measurement: 'clima_estacion',  // Estacion propia en planta.
112                 fields: {
113                     irradiancia_estacion: Influx.FieldType.FLOAT,
```

```
114                     temperatura_amb_estacion: Influx.FieldType.FLOAT,
115                     presion_estacion: Influx.FieldType.FLOAT,
116                     humedad_estacion: Influx.FieldType.FLOAT,
117                     wind_speed_estacion: Influx.FieldType.FLOAT,
118                 },
119                 tags: ['tag1', 'tag2'],
120             },
121             {
122                 measurement: 'carga',  // Shelly o Medidor Chint
123                 fields: {
124                     corriente1: Influx.FieldType.FLOAT, // Solo para caso
                        Monofasico
125                     corriente2: Influx.FieldType.FLOAT, // Solo para caso
                        trifasico
126                     corriente3: Influx.FieldType.FLOAT, // Solo para caso
                        trifasico
127                     tension: Influx.FieldType.FLOAT,
128                     energia_s1: Influx.FieldType.FLOAT, // Solo para Shelly 4PM
129                     energia_s2: Influx.FieldType.FLOAT,
130                     energia_s3: Influx.FieldType.FLOAT,
131                     energia_s4: Influx.FieldType.FLOAT,
132                     energia_histo: Influx.FieldType.FLOAT,
133                 },
134                 tags: ['tag1', 'tag2'],
135             },
136             {
137                 measurement: 'fallos_solax_planta',
138                 fields: {
139                     Mgr_Fault_Msg: Influx.FieldType.INTEGER,// 0x0043 Existe un
                        Fallo en el Inversor, ver tabla 2.5
140                     Inv_Fault_Message_LSB: Influx.FieldType.INTEGER,// 0X0040
                        Fallo del inversor LSB
141                     Inv_Fault_Message_MSB: Influx.FieldType.INTEGER,// 0X0041
                        Fallo del inversor MSB
142                     PCSMajorFault: Influx.FieldType.INTEGER,// 0x003E Hay un
                        Fallo en etapa DC/AC
143                     TemperFaultValue: Influx.FieldType.INTEGER,// 0X000C
                        Temperatura entorno Inversor
144                     Temperature: Influx.FieldType.INTEGER,// 0X0008 Temperatura
                        radiador Inversor
145                     RunMode: Influx.FieldType.INTEGER,// 0X0009 Modo de
                        funcionamiento, donde exite fallo general, ver tabla
                        2.2
146                     BatteryMajor_Fault: Influx.FieldType.INTEGER,// 0x003F
                        Fallo BUS DC Battery
147                     Bat_BMS_FaultMessage_LSB: Influx.FieldType.INTEGER,//0x0044
                        Codificacion tabla 2.6
148                     Bat_BMS_FaultMessage_MSB: Influx.FieldType.INTEGER,//0x0045
                        Codificacion tabla 2.7
149                     wBatteryVoltFaultVal: Influx.FieldType.INTEGER,//0x0069
                        Fallo de Vdc Battery
150                     Pv1VoltFault: Influx.FieldType.INTEGER,// 0x000D Fallo
                        tension FV1
151                     Pv2VoltFault: Influx.FieldType.INTEGER,// 0x000E Fallo
                        tension FV2
152                     GfciFaultValue: Influx.FieldType.INTEGER, //0X000F
                        AislamientoDC_Faul_mA
153                     GridVoltFaultValue: Influx.FieldType.INTEGER,//0x0010 Fallo
                        de tension de Red
154                     DciFaultValue: Influx.FieldType.INTEGER,// 0x0012 Fallo de
                        la corriente en FV
155                     GridFreqFaultValueT: Influx.FieldType.INTEGER,// 0x0011
                        Fallo de Frecuencia de Red
156                     //////
```

```
157                     fault_bomba: Influx.FieldType.INTEGER,  // Si bombea o no (
                            caudalimetro)
158                     fault_carga_aux: Influx.FieldType.INTEGER, // Si hay
                            tension en salida y corriente
159                     fault_eos_array: Influx.FieldType.INTEGER, // Si hay fallo
                            de comunicacion o dato anomalo
160                     fault_ad_4280V: Influx.FieldType.INTEGER,  // Si hay fallo
                            de comunicacion o dato anomalo
161                     fault_ad_4280I: Influx.FieldType.INTEGER,  // Si hay fallo
                            de comunicacion o dato anomalo
162                 },
163                 tags: ['tag1', 'tag2'],
164             },
165         ]
166     });
167     ////////////////////////////////////////////////
168     class InfluxUtil {
169         // Funcion para leer un campo desde InfluxDB
170         static async leerCampoInflux(campo, serie) {
171             try {
172                 // Consulta para obtener el ultimo registro de un measurement
173                 const query = `SELECT ${campo} FROM "${serie}" ORDER BY time
                        DESC LIMIT 1`;
174                 //const query1 = `SELECT limpPV_alm_ref FROM "alarma_limpPV"
                        ORDER BY time DESC LIMIT 1`
175                 const url = `http://localhost:8086/query?q=${encodeURIComponent
                        (query)}&db=BBDD_IOM23_TEST`;
176                 //console.log("URL --->", url)
177                 const response = await axios.get(url);
178                 const result = response.data.results[0].series[0].values[0];
179                 //console.log("QUERY --->", result)
180                 return result[1];
181             } catch (error) {
182                 console.error(`Error al leer el campo ${campo} de la serie ${
                        serie}:`, error.message);
183                 logError(0, error);
184                 throw error;
185             }
186         }
187         ////////////////////////////////////////////////
188         static async leerCamposEntreFechas(camposQuery, serie, startDate,
                endDate) {
189             try {
190                 const query = `SELECT ${camposQuery} FROM "${serie}" WHERE time
                        >= '${startDate}' AND time <= '${endDate}' limit 3`;
191                 const result = await influx.query(query);
192 
193                 const processedResult = result.map(row => {
194                     const { time, ...rest } = row;
195                     return rest;
196                 });
197                 console.log('Resultado de la consulta:');
198                 if (processedResult.length > 0) {
199                     const csvData = processedResult.map(row => Object.values(
                            row).join(','));
200                     const csvContent = Object.keys(processedResult[0]).join(',
                            ') + '\n' + csvData.join('\n');
201                     console.log("JSON ---> ", JSON.stringify(processedResult,
                            null, 2));
202                 } else {
203                     console.log("NO SE HA ENCONTRADO NADA !!!");
204                 }
205             } catch (error) {
206                 console.error('Error al ejecutar la consulta:', error);
```

```
207             }
208         }
209         ////////////////////////////////////////////////
210         static async Leer_Influx1(rbbdd, serie, campo) {
211             try {
212                 const query = `SELECT ${campo} FROM "${serie}" ORDER BY time
                        DESC LIMIT 1`;
213                 const url = 'http://localhost:8086/query?q=' +
                        encodeURIComponent(query) + `&db=${rbbdd}`;
214                 const response = await axios.get(url);
215                 const result = response.data.results[0].series[0].values[0][1];
216                 return result;
217             } catch (error) {
218                 console.error('Error:', error.message);
219             }
220         }
221     }
222     // Funcion para realizar una consulta a InfluxDB
223     async function queryInflux(query) {
224         try {
225             const result = await influx.query(query);
226             return result;
227         } catch (error) {
228             throw new Error(`Error al ejecutar la consulta: ${error}`);
229         }
230     }
231     // Funcion para leer datos de una serie en InfluxDB
232     async function leerSerieInflux(database, serie) {
233         const query = `SELECT * FROM "${serie}" ORDER BY time DESC LIMIT 1`;
234         const result = await queryInflux(query);
235         return result;
236     }
237     // Registrar_Alerta('PR', 'ACTIVO', 'ALTA, 'Revisar PR de todo el sistema,
            influx)
238     async function Registrar_Alerta_influx(tipo_alarma, estado, seve, nota) {
239         var dataPoints = [];
240         const fechaAct = new Date();
241         const fechaStr = fechaAct.toISOString();
242 
243         var point = {
244             measurement: 'alertas',
245             tags: {
246                 tag1: 'alerta1x',
247                 tag2: 'alerta2x',
248             },
249             fields: {
250                 planta: 'PBS001ZA',
251                 fecha: fechaStr,
252                 tipo_alarma: tipo_alarma,
253                 estado: estado,
254                 severidad: seve,
255                 notas: nota,
256             },
257         };
258         dataPoints.push(point);
259         // Escribir los puntos de datos en InfluxDB
260         influx.writePoints(dataPoints)
261             .then(() => {
262                 console.log(" >> Datos Grabados de la Alerta");
263             })
264             .catch((error) => {
265                 console.error('Error al escribir datos de Alerta en InfluxDB:',
                        error);
266             })
```

```
267     }
268     ////////////////////////////////////////////////
269     function grabar_scada_influx_IOM(subsistema, datos_subsistema) {
270         return new Promise(async (resolve, reject) => {
271           const currentDate = new Date();
272           const madridTimeZoneOffset = 120; // Offset en minutos para la zona
                horaria de Madrid
273           const fechaMadrid = new Date(currentDate.getTime() +
                madridTimeZoneOffset * 60000);
274           var dataPoint = []; // Arreglo para almacenar los objetos de punto de
                 datos
275           try {
276             if (subsistema == 'SOLAX_ALL') {
277               // Iterar sobre cada objeto en datos_subsistema --> medidas_arr
278               for (const key in datos_subsistema) {
279                 const medicion = datos_subsistema[key][0];
280                 const fields = medicion.fields;
281                 // Crear un objeto de punto de datos y agregarlo al arreglo
282                 var puntoDatos = {
283                   measurement: medicion.measurement.toLowerCase(),
284                   fields,
285                   tags: {
286                     tag1: "subsistema",
287                     tag2: medicion.measurement.toLowerCase(),
288                   },
289                   timestamp: fechaMadrid,
290                 };
291                 dataPoint.push(puntoDatos);
292               }
293             }
294             // Realizar la escritura en la base de datos INFLUX version 1.X
295             console.log(">> Valor de puntos a grabar ---> ", dataPoint);
296             try {
297               await influx.writePoints(dataPoint);
298               console.log('Datos grabados correctamente en InfluxDB.');
299               resolve();
300             } catch (error) {
301               console.error('>>> Error al escribir en InfluxDB --->', error);
302               reject(error);
303             }
304           } catch (error) {
305             console.error('Error de la funcion al grabar en INFLUX:', error);
306             reject(error);
307           }
308         });
309     }
310     // Exportar las funciones y modulos que deseas hacer accesibles
311     module.exports = {
312         InfluxUtil,
313         queryInflux,
314         leerSerieInflux,
315         Registrar_Alerta_influx,
316         grabar_scada_influx_IOM,
317         influx, // Tambien se puede exportar el modulo InfluxDB directamente si
                se necesita acceder a otras funcionalidades
318     };
```

A.2.8. Registro de Errores

- **logger_pino.js**

```
////////////////////////////////////////////////
// FICHERO: logger_pino.js
////////////////////////////////////////////////
const pino = require('pino');
const { errorTable } = require('./errorTable_pino');
////////////////////////////////////////////////
// Obtener la fecha y hora actual en la zona horaria de Madrid con Time Zone
function obtenerFechaTZ_Madrid(tipo) {
    const currentDate = new Date();
    const madridTimeZoneOffset = 120; // Offset en minutos para la zona horaria de Madrid
    const fechaMadrid = new Date(currentDate.getTime() + madridTimeZoneOffset * 60000);
    if (tipo == 1) return fechaMadrid.toISOString()
    else return (fechaMadrid);
}
////////////////////////////////////////////////
// Configuracion de la libreria pino (logger)
const logger = pino({
    level: 'error',
    timestamp: () => `,"time":"${obtenerFechaTZ_Madrid(1)}"`,
}, pino.destination('./logger/log_err.json'));
////////////////////////////////////////////////
function logError(errorCode, error) {
    const error_ID = errorTable[errorCode] || 'Error desconocido';
    var errorNum = error.errno;
    logger.error(`[${errorCode}] [${error_ID}] [${errorNum}] ${error.message}`);
}
////////////////////////////////////////////////
function logError_base(mensa) {
    logger.error(`${mensa}`);
}
////////////////////////////////////////////////
// exportar este modulo
module.exports = {
    logError,
    logError_base
};
```

- **errorTable_pino.js**

```
////////////////////////////////////////////////
// FICHERO: errorTable_pino.js
////////////////////////////////////////////////
const errorTable = {
    'Comunicacion RS232' : {
        dispositivo : 'SALICRU',
        errores : [
            {
                error : 'IDEPSA001',
                comentario : 'Error al conectarse mediante RS232'
            },
            {
                error : 'IDEPSA002',
                comentario : 'Error de NaK detectado en el buffer de recepcion de datos'
            },
```

```
                    {
                        error : 'IDEPSA003',
                        comentario : 'Configuracion del dispositivo
                            erronea(baudRate/parity/...)'
                    }

                ]
            },
            'Comunicacion RS485' : {
                eos_array : {
                    dispositivo : 'EOS',
                    errores : [
                        {
                            error : 'IDEPEOS001',
                            comentario : 'Error al conectarse mediante
                                RS485'
                        },
                        {
                            error : 'IDEPEOS002',
                            comentario : 'Configuracion del dispositivo
                                erronea(baudRate/parity/...)'
                        }
                    ]
                },
                adm_tension : {
                    dispositivo : 'ADM_V',
                    errores : [
                        {
                            error : 'IDEPADM_V001',
                            comentario : 'Error al conectarse mediante
                                RS485'
                        },
                        {
                            error : 'IDEPADM_V002',
                            comentario : 'Configuracion del dispositivo
                                erronea(baudRate/parity/...)'
                        }
                    ]
                },
                adm_current : {
                    dispositivo : 'ADM_I',
                    errores : [
                        {
                            error : 'IDEPADM_I001',
                            comentario : 'Error al conectarse mediante
                                RS485'
                        },
                        {
                            error : 'IDEPADM_I002',
                            comentario : 'Configuracion del dispositivo
                                erronea(baudRate/parity/...)'
                        }
                    ]
                }
            },
            'Comunicacion TCP' : {
                solax : {
                    dispositivo : 'SOLAX',
                    errores : [
                        {
                            error : 'IDEPSO001',
                            comentario : 'Error al conectarse mediante
                                TCP'
                        }
```

```
                    ]
                },
                shelly : {
                    dispositivo : 'SHELLY',
                    errores : [
                        {
                            error : 'IDEPSH001',
                            comentario : 'Error al conectarse mediante
                                TCP'
                        }
                    ]
                }
            },
            'Programa Main' : {
                dispositivo : 'MAIN',
                errores : [
                    {
                        error : 'IDEM001',
                        comentario : 'Error en el programa principal'
                    }

                ]
            }
        };
        // Exportar a todos los modulos de la Aplicacion
        module.exports = {
            errorTable
        };
```

A.2.9. TiposDeDatos.js

```
var exports = module.exports = {};
exports.Radix = Object.freeze({
    BE: {            //Big Endiand
        Uint8: "Uint8_BE",
        Int8: "Int8_BE",
        Uint16: "Uint16_BE",
        Int16: "Int16_BE",
        Uint32: "Uint32_BE",
        Int64: "Int64_BE",
        Uint64: "Uint64_BE",
        Int32: "Int32_BE",
        Float32: "Float32_BE",
        Inverse_Uint32: "Uint32_BE_I",
        Inverse_Int32: "Int32_BE_I",
        Inverse_Float32: "Float32_BE_I"
    },
    LE: {        //Little Endiand
        Uint8: "Uint8_LE",
        Int8: "Int8_LE",
        Uint16: "Uint16_LE",
        Int16: "Int16_LE",
        Uint32: "Uint32_LE",
        Int64: "Int64_LE",
        Uint64: "Uint64_LE",
        Int32: "Int32_LE",
        Float32: "Float32_LE",
        Inverse_Uint32: "Uint32_LE_I",
        Inverse_Int32: "Int32_LE_I",
        Inverse_Float32: "Float32_LE_I"
    }
});
```

APÉNDICE B
Objetivos de Desarrollo Sostenible

El grado de relación del trabajo con los Objetivos de Desarrollo Sostenible(ODS) se muestra en la siguiente tabla B.1.

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**		X		
ODS 2. **Hambre cero.**		X		
ODS 3. **Salud y bienestar.**	X			
ODS 4. **Educación de calidad.**		X		
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**	X			
ODS 7. **Energía asequible y no contaminante.**	X			
ODS 8. **Trabajo decente y crecimiento económico.**	X			
ODS 9. **Industria, innovación e infraestructuras.**	X			
ODS 10. **Reducción de las desigualdades.**				X
ODS 11. **Ciudades y comunidades sostenibles.**	X			
ODS 12. **Producción y consumo responsables.**	X			
ODS 13. **Acción por el clima.**	X			
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**		X		
ODS 16. **Paz, justicia e instituciones sólidas.**		X		
ODS 17. **Alianzas para lograr objetivos.**		X		

Tabla B.1: Tabla de los Objetivos de Desarrollo Sostenible

Reflexión sobre la relación del TFG con los ODS involucrados en el proyecto.

- **Fin de la pobreza:** Este proyecto contribuye al ODS 1 al desarrollar tecnologías y sistemas que pueden reducir los costos energéticos, especialmente en comunidades vulnerables. Al optimizar la gestión energética y fomentar el uso de energías renovables, se puede mejorar el acceso a la energía asequible, ayudando a mitigar la pobreza energética en regiones desfavorecidas.
- **Hambre cero:** Aunque la relación no es directa, la eficiencia energética y el uso de energías renovables desarrollados en este proyecto pueden aplicarse en la agricultura sostenible, mejorando la productividad y reduciendo los costos energéticos en la producción de alimentos. Esto puede contribuir a una mayor seguridad alimentaria y a la reducción del hambre.
- **Salud y Bienestar:** El proyecto promueve un entorno más saludable al reducir las emisiones de gases de efecto invernadero mediante el uso de energías renovables. Además, al disminuir la dependencia de combustibles fósiles, se mejora la calidad del aire, lo que contribuye directamente a la salud pública y el bienestar general.
- **Educación de calidad:** Este trabajo también puede ser una herramienta educativa, proporcionando conocimientos sobre energías renovables y eficiencia energética. Los resultados y el conocimiento generado pueden ser utilizados para formar a futuras generaciones en temas de sostenibilidad y tecnología.
- **Agua limpia y Saneamiento:** Aunque el enfoque principal del proyecto no es el agua, la eficiencia energética lograda puede ser aplicada a tecnologías que tratan y distribuyen agua potable. Además, el uso de energías renovables en estos sistemas puede asegurar un suministro continuo y sostenible de agua limpia.
- **Energía Asequible y No Contaminante:** Este es uno de los ODS más directamente relacionados con el proyecto. El TFG se centra en la optimización del uso de energías renovables, promoviendo una transición hacia fuentes de energía más limpias y sostenibles. Al mejorar la eficiencia energética y fomentar la adopción de estas tecnologías, el proyecto contribuye significativamente a asegurar el acceso universal a energía asequible, fiable y moderna.
- **Trabajo Decente y Crecimiento Económico:** Al fomentar el desarrollo de nuevas tecnologías en el sector de la energía renovable, el proyecto apoya la creación de empleos verdes y promueve un crecimiento económico sostenible. La innovación en eficiencia energética puede abrir nuevas oportunidades laborales y de negocio, contribuyendo al crecimiento económico sin comprometer el medio ambiente.
- **Industria, Innovación e Infraestructura:** El TFG impulsa la innovación tecnológica y contribuye al desarrollo de infraestructuras sostenibles. Al centrarse en soluciones energéticas avanzadas y sostenibles, el proyecto apoya la modernización de la infraestructura energética, lo cual es fundamental para el desarrollo industrial sostenible.
- **Ciudades y Comunidades Sostenibles:** Al promover el uso de energías renovables y mejorar la eficiencia energética, el proyecto contribuye a la creación de ciudades y comunidades más sostenibles. La reducción de la huella de carbono y la implementación de soluciones energéticas limpias son clave para el desarrollo de entornos urbanos resilientes y sostenibles.

- **Producción y Consumo Responsable:** El TFG fomenta un uso más eficiente y responsable de los recursos energéticos. La integración de tecnologías limpias y eficientes en la producción y el consumo de energía ayuda a reducir el desperdicio de recursos y promueve prácticas de consumo sostenible.
- **Acción por el clima:** El proyecto tiene un impacto directo en la mitigación del cambio climático, al reducir la dependencia de combustibles fósiles y promover el uso de energías renovables. Al contribuir a la reducción de emisiones de gases de efecto invernadero, el TFG apoya las acciones globales para combatir el cambio climático.
- **Vida de ecosistemas terrestres:** Al reducir la contaminación y las emisiones a través del uso de energías renovables, el proyecto también contribuye a la protección y restauración de los ecosistemas terrestres. La disminución de la presión sobre los recursos naturales es clave para preservar la biodiversidad y mantener los servicios ecosistémicos.
- **Paz, Justicia e Instituciones Sólidas:** La relación con este ODS puede interpretarse en el contexto de la gobernanza energética. Al promover prácticas energéticas más sostenibles y justas, el proyecto puede contribuir a un marco regulatorio y político que favorezca la paz y la justicia social, apoyando a instituciones más sólidas y resilientes.
- **Alianzas para Lograr Objetivos:** Finalmente, el TFG destaca la importancia de la colaboración interdisciplinaria y las alianzas para avanzar en los ODS. La implementación de tecnologías energéticas sostenibles requiere la cooperación entre la ONG, la academia, el sector privado y la sociedad civil. El proyecto demuestra cómo la colaboración y las alianzas son esenciales para alcanzar los objetivos globales de sostenibilidad.