ISSN: 1692-7257 - Volumen 1 – Número 45 - 2025

RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

Digital Object Identifier: 10.24054/rcta.v1i45.3091

Prototipo de sistema automatizado para lectura y control de medidores de energía inteligentes

Prototype of an automated system for reading and control of smart meters

Ing. Ronald Stivel Melo Cárdenas[1], Msc. Luis Alfredo Rodríguez Umaña[1]
Msc. Javier Eduardo Martínez Baquero[1], Msc. Nelson Baquero Álvarez[1]

[1] ***Universidad de los Llanos,*** *Facultad de Ciencias Básicas e Ingeniería, Ingeniería Electrónica, Villavicencio, Meta, Colombia.*

Correspondencia: *jmartinez@unillanos.edu.co*

Recibido: *13 septiembre 2024.* ***Aceptado:*** *17 diciembre 2024.* ***Publicado:*** *01 enero 2025.*

Cómo citar: *R. S. Melo Cárdenas, L. A. Rodríguez Umaña, J. E. Martinez Baquero, y N. Baquero Álvarez, «Prototipo de sistema automatizado para lectura y control de medidores de energía inteligentes», RCTA, vol. 1, n.º 45, pp. 57–65, ene. 2025. Recuperado de https://ojs.unipamplona.edu.co/index.php/rcta/article/view/3091*

Resumen: El presente artículo expone la implementación de un prototipo de sistema automatizado para la lectura y control de variables en medidores de energía eléctrica, siendo una respuesta fundamental ante los desafíos contemporáneos y futuros en el sector de distribución de energía. Este enfoque no sólo aborda las demandas actuales del panorama energético, caracterizadas por la necesidad de eficiencia y gestión inteligente, sino que también se proyecta hacia el futuro, anticipando y enfrentando los desafíos emergentes en la expansión y modernización de las infraestructuras eléctricas, optimizando la lectura remota y el control preciso de variables. Los resultados muestran una oportunidad significativa para mejorar la eficiencia operativa y reducir costos en el sector eléctrico, siendo la adopción de protocolos de comunicación como DLMS/COSEM y Modbus, respaldada por la implementación de sistemas telemétricos en plataformas como Raspberry Pi u Orange Pi con programación en Python.

Palabras clave: medidores inteligentes, modbus, redes eléctricas inteligentes, sistema automatizado.

Abstract: This article presents the implementation of a prototype of an automated system for the reading and control of variables in electric energy meters, being a fundamental response to contemporary and future challenges in the energy distribution sector. This approach not only addresses the current demands of the energy landscape, characterized by the need for efficiency and intelligent management, but also projects into the future, anticipating and facing the emerging challenges in the expansion and modernization of electrical infrastructures, optimizing remote reading and accurate control of variables. The results show a significant opportunity to improve operational efficiency and reduce costs in the electricity sector, being the adoption of communication protocols such as DLMS/COSEM and Modbus, supported by the implementation of telemetric systems on platforms such as Raspberry Pi or Orange Pi with Python programming.

Keywords: smart meters, modbus, smart grids, automated system.

1. INTRODUCCIÓN

En un mundo cada vez más enfocado en la eficiencia energética y la gestión sostenible de recursos, los medidores inteligentes se han convertido en elementos fundamentales para la supervisión y control de la energía eléctrica [1], [2]. Estos dispositivos, utilizados en redes de distribución eléctrica, permiten una recopilación y análisis precisos de los datos de consumo, lo que a su vez fomenta un uso más responsable de la energía [3].
Para su funcionamiento, estos medidores inteligentes dependen de protocolos de comunicación específicos, como el DLMS/COSEM o Modbus [4], los cuales controlan la interacción y el intercambio de información con otros dispositivos y sistemas. En este contexto, se ha vuelto esencial comprender las características, ventajas y aplicaciones de estos protocolos para garantizar la operación eficiente y segura de la información en las redes eléctricas [5], [6], [7], [8]. El paradigma de más rápido crecimiento en la distribución y gestión de servicios públicos, en el sector de la energía, es la Red Inteligente (SG, Smart Grid por sus siglas en inglés) [9], [10], [11]. De manera básica, una red inteligente representa una red de servicios públicos inteligente que incorpora capacidades de comunicación bidireccional y computo distribuido a través de los diferentes nodos de la red para mejorar el control, la eficiencia, la confiabilidad y la seguridad [12]. Las SG siendo estructuras de comunicación basadas en IoT son susceptibles a ataques cibernéticos debido a su compleja arquitectura y sistemas de comunicación críticos [13]. Esta vulnerabilidad exige una investigación minuciosa en la industria, el gobierno y la academia para reforzar la seguridad de las mismas. Las redes inteligentes aprovechan la tecnología de la información para la entrega eficiente de energía, enfrentando amenazas de seguridad significativas por las debilidades en la tecnología de comunicación [14].

Comprender las motivaciones detrás de los ataques a las redes inteligentes, que ahora incluyen métodos tanto físicos como cibernéticos para manipular datos de consumo energético, es crucial para mejorar su seguridad operacional [15]. A medida que aumentan los requisitos de energía, surgen nuevos desafíos en la gestión de la energía y el control de la demanda. Por lo tanto, las SG se desarrollan junto con tecnologías de electrónica de potencia, sensores y medición, lo que permite que la red adquiera mayor inteligencia a través de la gestión de la comunicación y el control inteligente [16]. Los medidores inteligentes, que son componentes fundamentales en las redes inteligentes, dependen de protocolos estandarizados para su gestión y control [17].

Dentro de los protocolos más usados está DLMS/COSEM el cual es un protocolo de capa de aplicación abierto y gratuito que se utiliza para una amplia gama de aplicaciones de medición, incluyendo la lectura remota de medidores, el control remoto y los servicios de valor añadido. Otro protocolo de amplio uso es Modbus, el cual establece reglas y formatos comunes para la transmisión de información entre un dispositivo maestro y uno o varios dispositivos esclavos. El protocolo Modbus se destaca por su simplicidad y facilidad de implementación, lo que lo convierte en una opción popular en una amplia gama de aplicaciones industriales, como el control de procesos, la automatización de edificios, el monitoreo de energía y más [18].

El artículo se estructura en cuatro apartados, la presente introducción que hace referencia al estado del arte. La segunda sección que expone los métodos y materiales en tres subsecciones: requerimientos, selección de lenguaje de programación y hardware, y diseño de firmware. La tercera sección presenta los resultados y su análisis y finalmente la cuarta sección presenta las conclusiones alcanzadas.

2. MATERIALES Y MÉTODOS

Para el desarrollo del sistema automatizado de lectura y control de variables en el medidor de energía eléctrica inteligente, se tomó como referencia la metodología de modelo en V, el cual es un procedimiento uniforme para el desarrollo tanto de productos TIC como de software [19], ya que el proceso de desarrollo que se propone debe tener como propósito fundamental un producto de calidad que cumpla con las especificaciones y responda a las expectativas y necesidades del cliente y/o usuario final [20], [21], [22], [23], [24], [25].

En la Fig. 1 se puede ver la descripción de la metodología. En el proceso de desarrollo, se establecen diferentes fases que son esenciales para garantizar la calidad y funcionalidad del sistema.

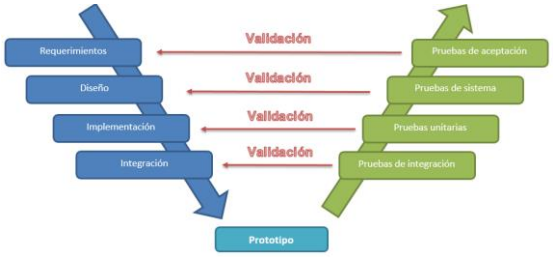

Fig. 1. Metodología a utilizar en el desarrollo del prototipo
***Fuente:** elaboración propia*

2.1. Requerimientos

En esta fase del proyecto, se procede a la definición y establecimiento detallado de los requisitos fundamentales que guiarán el desarrollo del prototipo, los cuales se clasifican en cuatro categorías: los de usuarios, funcionales, de calidad y de implementación. Los Requerimientos de Usuarios se centran en las necesidades y preferencias específicas de los usuarios finales del sistema, asegurando una experiencia intuitiva y satisfactoria. Los Requerimientos Funcionales detallan las funciones y operaciones específicas que el sistema debe realizar para cumplir con su propósito principal. Por su parte, los requerimientos de calidad establecen criterios y estándares que garantizarán la eficiencia, confiabilidad y seguridad del sistema. Finalmente, los Requerimientos de Implementación abordan consideraciones prácticas y técnicas esenciales para la exitosa ejecución del proyecto. Este proceso de definición de requisitos sienta las bases para un desarrollo estructurado y orientado a resultados, alineado con las expectativas y necesidades clave identificadas. Los requerimientos establecidos son:

Compatibilidad con Medidores Inteligentes Trifásicos de Medición Indirecta (RE1): El sistema debe ser compatible con diferentes modelos de medidores inteligentes trifásicos que operen mediante medición indirecta, garantizando una amplia aplicabilidad del proyecto.

Interfaz de Usuario Intuitiva (RE2): Se requiere una interfaz de usuario intuitiva y fácil de manejar para que los usuarios puedan configurar y operar el sistema sin dificultades. Este requisito abarca aspectos funcionales y de calidad, asegurando que el sistema sea accesible para todos los usuarios, incluyendo aquellos con poca experiencia técnica.

Implementación Efectiva de Protocolos de Comunicación (RE3): El sistema debe implementar de manera efectiva los protocolos de comunicación DLMS/COSEM o Modbus para asegurar la interoperabilidad y la eficiencia en la transmisión de datos entre los dispositivos del sistema.

Capacidad para Realizar Lecturas Remotas y Controles Específicos en Tiempo Real (RE4): Es fundamental que el sistema permita realizar lecturas remotas y ejercer controles específicos sobre los medidores en tiempo real, facilitando una gestión energético más eficiente y reactiva.

Utilización de Hardware Especifico para la Implementación de Telemetría (RE5): El sistema debe ser capaz de operar con hardware especifico como Raspberry Pi, Orange Pi, ESP32, o microcontroladores Atmega 2560 para la implementación de funciones de telemetría, lo cual es crucial para la recopilación y transmisión de datos.

Desarrollo de Módulos de Control (RE6): Se deben desarrollar módulos de control dedicados para gestionar funciones criticas como la conexión, conexión y activación de bancos de condensadores, mejorando así la eficiencia y la capacidad de respuesta del sistema.

Integración del Protocolo MQTT (RE7): La integración del protocolo MQTT para la transmisión de datos a la plataforma de gestión de información es esencial para asegurar una comunicación eficiente y segura entre los dispositivos y el sistema central.

Mecanismos de Seguridad Robustos (RE8): El sistema debe incorporar mecanismos de seguridad robustos para proteger la integridad y privacidad de los datos transmitidos, un aspecto critico en el contexto actual de preocupaciones sobre la seguridad cibernética.

Realización de Pruebas Exhaustivas (RE9): Antes de su implementación definitiva, el sistema debe ser sometido a pruebas exhaustivas para evaluar su funcionalidad y rendimiento, asegurando que cumpla con todos los requisitos establecidos y que esté libre de defectos.

2.2. Selección de lenguaje de programación y hardware

En esta fase crítica del desarrollo del proyecto se emprende la tarea de definir de manera integral la arquitectura del sistema de telemetría, abarcando tanto el aspecto hardware como la elección del lenguaje de programación. La confección detallada del firmware emerge como un componente esencial para establecer una comunicación efectiva con el medidor y facilitar la transmisión precisa de datos a la plataforma AOM ENERTEC.

La selección del lenguaje de programación fue objeto de un meticuloso proceso de preselección que evaluó tres opciones destacadas: Python, C# y C++. A continuación, se proporciona una descripción completa de las características más resaltantes de cada lenguaje, abordando aspectos como el paradigma de programación, sintaxis, rendimiento, manejo de memoria, versatilidad, así como la robustez de su comunidad y ecosistema:

Paradigma de Programación: Python es conocido por su flexibilidad, admitiendo paradigmas orientado a objetos, imperativo y funcional. C++, por otro lado, se enfoca principalmente en el paradigma orientado a objetos e imperativo. C# se adhiere estrictamente al paradigma orientado a objetos, lo que refleja su diseño y utilización.

Sintaxis: Python destaca por su sintaxis clara y concisa, lo que facilita su lectura y escritura. C++ presenta una sintaxis más compleja y detallada, lo que puede representar una curva de aprendizaje más pronunciada. C#, con una sintaxis similar a Java y C++, ofrece una estructura orientada a objetos bien definida.

Rendimiento: Python generalmente opera a una velocidad más lenta comparado con C++, debido a su naturaleza interpretada. C++ es altamente reconocido por su alto rendimiento, ya que se compila directamente a código máquina. C# ofrece un rendimiento sólido, aunque puede ser ligeramente más lento que C++ debido a su ejecución en el entorno de tiempo de ejecución de .NET.

Tipado: Python utiliza tipado dinámico, lo que permite mayor flexibilidad durante el tiempo de ejecución. Tanto C++ como C# emplean un sistema de tipado estático, lo que requiere que los tipos de datos se declaren explícitamente.

Manejo de Memoria: Python y C# gestionan la memoria automáticamente, liberando a los desarrolladores de la gestión manual de memoria. C++, en contraste, requiere que el programador maneje la memoria explícitamente, lo que permite un control más fino, pero aumenta la complejidad.

Versatilidad: Python es ampliamente utilizado en campos como el desarrollo web, análisis de datos e inteligencia artificial. C++ se destaca en el desarrollo de sistemas embebidos y aplicaciones de alto rendimiento. C# es frecuentemente elegido para el desarrollo de aplicaciones Windows, respaldado por el fuerte soporte de Microsoft.

Comunidad y Ecosistema: Los tres lenguajes disfrutan de comunidades grandes y activas. Python y C++ tienen un amplio espectro de aplicaciones y bibliotecas disponibles. C# cuenta con el respaldo de Microsoft y una vasta gama de recursos para desarrolladores.

Aplicaciones Típicas: Python se utiliza en desarrollo web y análisis de datos; C++ en sistemas embebidos, juegos y sistemas de tiempo real; y C# en aplicaciones Windows.

Plataformas Soportadas: Python y C++ son multiplataforma, aunque C++ es a menudo preferido para sistemas nativos. C# se utiliza principalmente en entornos Windows, aunque .NET Core ha extendido su aplicabilidad a otras plataformas.

Simultáneamente, la elección del hardware fue de igual trascendencia, contemplando cuatro tarjetas de desarrollo: Orange Pi, Raspberry Pi, ESP32 y Arduino Mega. A continuación, se presenta una clasificación detallada que aborda aspectos críticos como procesamiento, almacenamiento, conectividad, respaldo de la comunidad de desarrollo y asequibilidad en función de los costos asociados:

Arquitectura: Orange Pi utiliza ARM Cortex-A7, Raspberry Pi utiliza ARM Cortex-A72, ESP32 opera con Xtensa LX6, y ATMega 2560 usa AVR.

Velocidad del Procesador: Orange Pi alcanza 1.6 GHz, Raspberry Pi llega a 1.5 GHz, ESP32 opera hasta 240 MHz, y ATMega 2560 funciona a 16 MHz.

RAM: Orange Pi tiene 1 GB DDR3, Raspberry Pi ofrece versiones con 2 GB, 4 GB o 8 GB LPDDR4, ESP32 dispone de 520 KB SRAM, y ATMega 2560 cuenta con 8 KB SRAM (de los cuales 2.5 KB están disponibles para el usuario).

Almacenamiento: Orange Pi y Raspberry Pi disponen de ranura para tarjeta microSD, ESP32 puede tener hasta 16 MB Flash, y ATMega 2560 incluye 256 KB Flash y 4 KB EEPROM.

Conectividad: Orange Pi provee 10/100 Ethernet y WiFi, Raspberry Pi cuenta con Gigabit Ethernet

y WiFi, ESP32 incluy e WiFi y Bluetooth, y ATMega 2560 no tiene Ethernet, con varios modelos ofreciendo opciones con/sin WiFi.

Puertos USB: Orange Pi ofrece 3 USB 2.0, Raspberry Pi tiene 2 USB 3.0 y 2 USB 2.0, ESP32 cuenta con 1 USB 1.1 (OTG) y 1 USB 2.0, y ATMega 2560 dispone de 4 USB 2.0.

Pines GPIO: Orange Pi y Raspberry Pi tienen 40pines GPIO, ESP32 cuenta con 36, y ATMega 2560 ofrece 54 pines GPIO, de los cuales 14 son PWM.

Sistema Operativo: Orange Pi y Raspberry Pi son compatibles con Linux (Raspberry Pi también soporta Windows 10 IoT Core), ESP32 trabaja con Free RTOS entre otros, y ATMega 2560 no tiene un sistema operativo incorporado, siendo programado a través de un IDE.

Comunidad y Soporte: Orange Pi tiene una comunidad y soporte moderados, mientras que Raspberry Pi, ESP32 y ATMega 2560 gozan de una amplia comunidad y soporte.

Precio Aproximado: Orange Pi y Raspberry Pi tienen precios moderados, que varían en el caso de Raspberry Pi según el modelo; ESP32 y ATMega 2560 son considerados asequibles.

Estas decisiones estratégicas en la arquitectura del sistema sientan las bases para un desarrollo coherente y eficiente. La sinergia entre el lenguaje de programación y el hardware seleccionados desempeñará un papel fundamental en el éxito de la implementación de la telemetría destinada a la gestión de medidores inteligentes trifásicos de medición indirecta

2.3. Diseño de firmware

En la fase de diseño del firmware, se llevó a cabo un análisis detallado del flujo de información, desde el instante en que es recolectada por el medidor hasta su envío a la plataforma AOM de la empresa Enertec Latinoamérica S.A.S. Este proceso se visualiza de manera clara en el diagrama de flujo presentado en la Fig. 2. El diseño de este flujo busca garantizar una comunicación eficiente, precisa y segura, asegurando que los datos recopilados por el medidor sean transmitidos de manera óptima a la plataforma correspondiente. La Fig. 2 sirve como una representación gráfica que ilustra las etapas y conexiones esenciales en este flujo de información, facilitando una comprensión visual del sistema que se busca implementar.

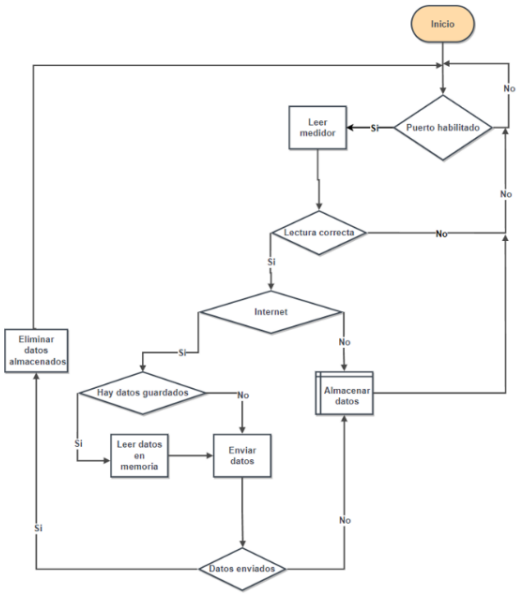

Fig. 2. *Diagrama de flujo de algoritmo propuesto para la solución.*
Fuente: *elaboración propia*

Con el objetivo de conferir un comportamiento sistemático al firmware, se decidió adoptar la utilización de máquinas de estados. Este enfoque aporta un valor significativo al proporcionar una estructura lógica y visual que facilita la gestión de estados y transiciones en el sistema. En el ámbito de la programación y desarrollo de firmware, la máquina de estado modela el comportamiento del sistema al dividirlo en estados discretos y definir las transiciones entre ellos, generando así un código más organizado y comprensible.

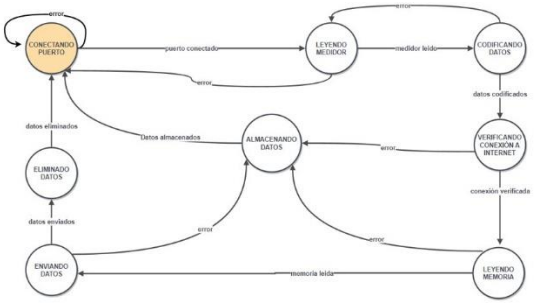

Fig. 3. *Diagrama de máquina de estados propuesto para la solución.*
Fuente: *elaboración propia*

La principal virtud de las máquinas de estado reside en su habilidad para representar de manera eficaz situaciones complejas y dinámicas. En el proyecto de telemetría, cada estado se corresponde con una etapa específica del proceso, tales como la conexión al medidor, la lectura de datos, la codificación de información, entre otros, como se puede observar en

la Fig. 3 Las transiciones entre estados reflejan cómo el sistema responde a eventos o condiciones particulares.

3. RESULTADOS

El protocolo DLMS/COSEM se escogió como la opción más apropiada para llevar a cabo el proceso de lectura y control del medidor de energía eléctrica inteligente. Este protocolo de capa de aplicación ofrece un enfoque abierto y gratuito, estableciendo estándares para diversas aplicaciones de medición, incluida la lectura remota y el control a distancia. Su flexibilidad y capacidad para describir interfaces para diversos objetos, como voltaje y corriente, además de que es un protocolo bidireccional y de implementación “end to end”, junto con su alta seguridad de acceso, protección de canales de comunicación y encriptación de datos que se transmiten durante todas las etapas, esto lo convirtió en una elección sólida para la gestión eficiente de este tipo de medidores inteligentes, destacando en estos aspectos frente al protocolo Modbus.

La estandarización de DLMS/COSEM, respaldada por la Asociación de Usuarios de DLMS (DLMS UA) y las normas IEC 62056, aseguró la interoperabilidad y la adopción en el contexto tecnológico de ENERTEC Latinoamérica S.A.S. La implementación de este protocolo facilito la lectura precisa y un control efectivo, contribuyendo así a una gestión energética más avanzada y a la mejora operativa de los servicios proporcionados por la empresa.

Luego de evaluar las características de las diferentes placas de desarrollo descritas en la sección de materiales y métodos se determinó que la Raspberry Pi, respaldada por un conversor USB a DB9 basado en el Chip CH-340 el cual se seleccionó debido a bajo costo, era la elección adecuada para facilitar el control y monitoreo remoto del medidor de energía eléctrica inteligente. La asignación de puntos según los criterios de selección se resume en la Tabla I.

Cada criterio se calificó en una escala de 1 a 5, donde 1 representa un rendimiento bajo y 5 un rendimiento alto. La Raspberry Pi, destacó especialmente en potencia de cómputo, facilidad de uso, conectividad, compatibilidad, E/S y puertos, comunidad y soporte, así como el costo, lo que contribuyó a su elección como la opción más equilibrada y eficiente para el proyecto.

Tabla 1: *Comparación de placas de desarrollo*

Criterio	OPi	RasPi	ESP32	ATMega 2560
Potencia de Cómputo	3	5	2	1
Facilidad de Uso	3	5	4	4
Conectividad	4	5	4	2
Compatibilidad	3	5	4	3
E/S y Puertos	4	5	4	5
Comunidad y Soporte	3	5	4	4
Costo	4	3	5	5
Versatilidad de Proyectos	3	5	4	4
Total	27	38	31	28

Fuente: *elaboración propia*

En la selección del lenguaje de programación, se optó por Python debido a los múltiples beneficios que ofrece para su desarrollo. La evaluación de Python se llevó a cabo junto con los otros lenguajes preseleccionados, utilizando criterios predefinidos donde la puntuación como en la selección del hardware oscila entre 1 (más baja) y 5 (más alta). La elección de Python se fundamentó en su puntuación total de 30 como se observa en la Tabla II, destacándose como la opción más idónea en comparación con C++ y C#˙ Este resultado resalta las características distintivas de Python, como su simplicidad y legibilidad de código, lo que facilita tanto el desarrollo rápido como la comprensión del programa. Además, la extensa disponibilidad de bibliotecas y frameworks en Python proporciona herramientas valiosas para la manipulación de datos y la implementación de protocolos de comunicación, como DLMS/COSEM.

Tabla 2: *Comparación de lenguajes de programación*

Criterio	Phyton	C++	C#
Facilidad de aprendizaje	4	3	4
Rendimiento	2	5	4
Manejo de memoria	4	2	4
Versatilidad	5	4	4
Comunidad y Ecosistema	5	5	4
Aplicaciones típicas	5	4	4
Plataformas soportadas	5	4	4
Total	**30**	**27**	**28**

Fuente: *elaboración propia*

La implementación exitosa del prototipo diseñado para cumplir con requisitos específicos que posibilitan la telemetría esencial en el control y monitoreo de medidores de energía eléctrica inteligentes, es un avance destacado en la convergencia de hardware, firmware y

comunicación efectiva. Este artículo examina la combinación estratégica de la Raspberry Pi como plataforma central, el conversor USB a DB9 como interfaz de conectividad clave y el lenguaje de programación Python en el desarrollo del firmware. Se aborda de manera detallada la robustez y eficiencia de esta combinación en términos de transmisión de datos y la interacción sinérgica con el medidor inteligente.

La Fig. 4 complementa este análisis al visualizar de manera gráfica el flujo de información en el proyecto, proporcionando una representación clara de la armoniosa integración de componentes clave en la implementación del sistema.

Este estudio destaca los logros, desafíos superados y contribuciones significativas de la solución propuesta, brindando una perspectiva valiosa para futuras investigaciones en el ámbito de la telemetría y gestión de medidores eléctricos inteligentes.

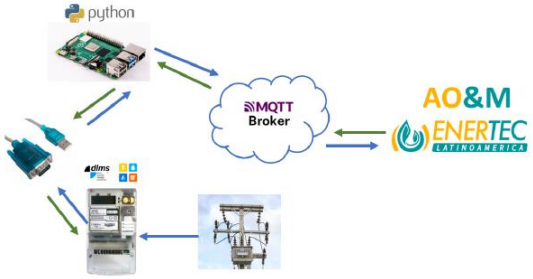

***Fig. 4**. Diagrama de flujo de la información en el proyecto*
***Fuente:** elaboración propia*

El proyecto fue implementado exitosamente en dos redes de media tensión, a continuación, en la Fig 5 se muestra la implementación física del prototipo.

***Fig. 5**. Prototipo funcional instalado en Mitú (Vaupés)*
***Fuente:** elaboración propia*

Luego de llevar a cabo una evaluación del sistema automatizado, abordando aspectos clave como la lectura de datos, la capacidad de respuesta en el control y la accesibilidad remota, el sistema demostró un rendimiento confiable en la recopilación de datos del medidor inteligente, con tiempos de respuesta óptimos para las operaciones de control. Además, la capacidad de acceder y monitorear el sistema de forma remota ha sido validada con éxito, confirmando la eficacia del prototipo en entornos de telemetría. A continuación, en la Fig. 6 y Fig. 7 se pueden ver imágenes del dashboard de la plataforma AO&M a través del cual se hace monitoreo.

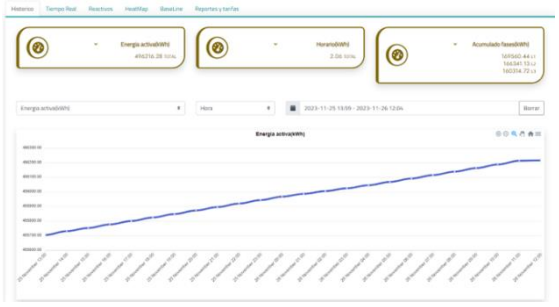

***Fig. 6.** Muestro por hora del sistema instalado*
***Fuente:** elaboración propia*

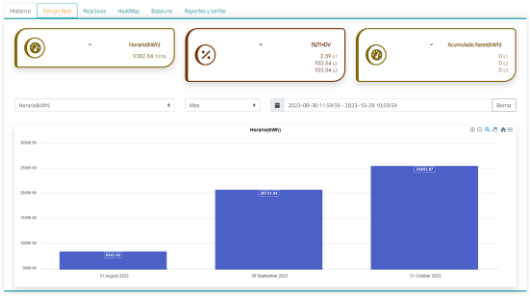

***Fig. 7.** Consumo en periodo de 3 meses*
***Fuente:** elaboración propia*

4. CONCLUSIONES

La adopción de protocolos de comunicación como DLMS/COSEM y Modbus, respaldada por la implementación de sistemas telemétricos en plataformas como Raspberry Pi u Orange Pi con programación en Python, ha demostrado ser un enfoque efectivo y versátil.

Este contexto de automatización no sólo responde a una necesidad coyuntural, sino que impulsa la evolución continua del sector eléctrico hacia practicas más eficientes, sostenibles y alineadas con las demandas cambiantes de la sociedad y la tecnología.

Las limitaciones asociadas con la gestión manual de medidores inteligentes, como la ineficiencia, los costos operativos y los errores humanos, subrayan la

necesidad critica de soluciones automatizadas. El prototipo exitoso desarrollado ofrece una respuesta a estos desafíos, proporcionando una base sólida para la modernización de la infraestructura eléctrica en Colombia.

La implementación de tecnologías de telemetría en una fase inicial en Colombia presenta oportunidades significativas para mejorar la eficiencia operativa y reducir costos en el sector eléctrico. Además, la capacidad de monitoreo remoto, la recopilación de datos precisa y la adaptabilidad del sistema a cambios futuros contribuyen al desarrollo sostenible y a una mejora integral de los servicios energéticos en el país. Este proyecto sienta las bases para futuras innovaciones en la gestión energética, respaldando la evolución y la modernización continua del sector eléctrico colombiano.

RECONOCIMIENTO

Los autores agradecen a la Universidad de los Llanos y Enertec Latinoamérica S.A.S. por su apoyo en el desarrollo de este proyecto.

REFERENCIAS

[1] P. Sospiro, L. Amarnath, V. Di Nardo, G. Talluri, and F. H. Gandoman, “Smart grid in china, EU, and the US: State of implementation,” Energies, vol. 14, no. 18, pp. 1–15, 2021, doi: 10.3390/en14185637.

[2] W. M. Giral Ramírez, H. J. Celedón Flórez, E. Galvis Restrepo, and A. T. Zona Ortiz, “Redes inteligentes en el sistema eléctrico colombiano: Revisión de tema,” Tecnura, vol. 21, no. 53, pp. 119–137, 2017, doi: 10.14483/22487638.12396.

[3] Y. Kabalci, “A survey on smart metering and smart grid communication,” Renew. Sustain. Energy Rev., vol. 57, pp. 302–318, 2016, doi: https://doi.org/10.1016/j.rser.2015.12.11.

[4] A. Sahu and A. Goulart, “Implementation of a C-UNB Module for NS-3 and Validation for DLMS-COSEM Application Layer Protocol,” in 2019 IEEE ComSoc International Communications Quality and Reliability Workshop (CQR), 2019, pp. 1–6. doi: 10.1109/CQR.2019.8880075.

[5] P. Matoušek, “Analysis of DLMS Protocol,” 2017. [Online]. Available: https://www.fit.vut.cz/research/publication-file/11616/TR-DLMS.pdf

[6] L. J. Weith, DLMS / COSEM protocol security evaluation. Eindhoven: Eindhoven University Technology, 2014. [Online]. Available: https://pure.tue.nl/ws/portalfiles/portal/46962657/773263-1.pdf

[7] H. Mendes, I. Medeiros, and N. Neves, “Validating and Securing DLMS/COSEM Implementations with the ValiDLMS Framework,” in 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), 2018, pp. 179–184. doi: 10.1109/DSN-W.2018.00060.

[8] D. L. Jiménez, J. A. Rea, P. R. Muñoz, G. E. Vizuete, L. J. Latacunga, and C. A. Iza, “Diseño y Construcción de un Medidor de Energía Eléctrica Domiciliar,” Rev. Técnica “energía,” vol. 20, no. 1, pp. 82–92, 2023, doi: 10.37116/revistaenergia.v20.n1.2023.573.

[9] J. J. Moreno Escobar, O. Morales Matamoros, R. Tejeida Padilla, I. Lina Reyes, and H. Quintana Espinosa, “A Comprehensive Review on Smart Grids: Challenges and Opportunities,” Sensors, vol. 21, no. 21. 2021. doi: 10.3390/s21216978.

[10] C. Wietfeld, A. A. Cardenas, H.-H. Chen, P. Popovski, and V. W. S. Wong, “Smart Grids,” IEEE Wirel. Commun., vol. 24, no. 2, pp. 8–9, 2017, doi: 10.1109/MWC.2017.7909091.

[11] M. Farmanbar, K. Parham, Ø. Arild, and C. Rong, “A Widespread Review of Smart Grids Towards Smart Cities,” Energies, vol. 12, no. 23. 2019. doi: 10.3390/en12234484.

[12] D. D. Vyas and H. N. Pandya, “Advance Metering Infrastructure and DLMS/COSEM Standards for Smart Grid,” Int. J. Eng. Res. Technol., vol. 1, no. 10, pp. 1–5, 2012, [Online]. Available: https://www.ijert.org/published-issue-archive

[13] Z. El Mrabet, N. Kaabouch, H. El Ghazi, and H. El Ghazi, “Cyber-security in smart grid: Survey and challenges,” Comput. Electr. Eng., vol. 67, pp. 469–482, 2018, doi: https://doi.org/10.1016/j.compeleceng.2018.01.015.

[14] M. Z. Gunduz and R. Das, “Cyber-security on smart grid: Threats and potential solutions,” Comput. Networks, vol. 169, p. 107094, 2020, doi: https://doi.org/10.1016/j.comnet.2019.107094.

[15] C. Barreto and A. A. Cárdenas, “Impact of the Market Infrastructure on the Security of Smart Grids,” IEEE Trans. Ind. Informatics, vol. 15, no. 7, pp. 4342–4351, 2019, doi: 10.1109/TII.2018.2886292.

[16] D. B. Avancini, J. J. P. C. Rodrigues, S. G. B. Martins, R. A. L. Rabêlo, J. Al-Muhtadi, and P. Solic, “Energy meters evolution in smart

grids: A review," J. Clean. Prod., vol. 217, pp. 702–715, 2019, doi: https://doi.org/10.1016/j.jclepro.2019.01.229.

[17] J. Zheng, D. W. Gao, and L. Lin, "Smart Meters in Smart Grid: An Overview," in 2013 IEEE Green Technologies Conference (GreenTech), 2013, pp. 57–64. doi: 10.1109/GreenTech.2013.17.

[18] Y. Fang, X. Han, and B. Han, "Research and Implementation of Collision Detection Based on Modbus Protocol," vol. 6, no. 1, pp. 91–96, 2013, [Online]. Available: http://www.jestr.org/index.php?option=com_content&view=article&id=28&Itemid=68

[19] E. García Sánchez, O. Vite Chávez, M. Á. Navarrete Sánchez, and M. Á. García Sánchez, "Metodología para el desarrollo de software multimedia educativo MEDESME," Rev. Investig. Educ. 23, vol. 23, no. Julio-Diciembre, pp. 217–226, 2016.

[20] A. F. Díaz, B. Prieto, J. J. Escobar, and T. Lampert, "Vampire: A smart energy meter for synchronous monitoring in a distributed computer system," J. Parallel Distrib. Comput., vol. 184, p. 104794, 2024, doi: https://doi.org/10.1016/j.jpdc.2023.104794.

[21] J. G. Fierro Mendoza, J. A. Asato España, J. B. Molina Castro, J. G. Delgado Núñez, and E. Noriega Vaca, "PROPUESTA METODOLÓGICA PARA VALIDAR LA FUNCIONALIDAD DE SOFTWARE EN SISTEMAS EMBEBIDOS," Pist. Educ., vol. 38, no. 122, pp. 156–177, 2016, [Online]. Available: https://pistaseducativas.celaya.tecnm.mx/index.php/pistas/article/view/689

[22] S. M. Velásquez Restrepo, J. D. Vahos Montoya, M. E. Gómez Adasme, E. J. Restrepo Zapata, A. A. Pino Martinez, and S. Londoño Marín, "Una revisión comparativa de la literatura acerca de metodologías tradicionales y modernas de desarrollo de software," Rev. CINTEX, vol. 24, no. 2, pp. 13–23, 2019, doi: 10.33131/24222208.334.

[23] M. V. Estrada-Velasco, J. A. Núñez-Villacis, P. R. Saltos-Chávez, and W. C. Cunuhay-Cuchipe, "Revisión Sistemática de la Metodología Scrum para el Desarrollo de Software," Rev. Científica Dominio las Ciencias, vol. 7, no. 4, pp. 434–447, 2021, doi: 10.23857/dc.v7i4.2429.

[24] B. Molina Montero, H. Vite Ceballos, and J. Dávila Cuesta, "Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software," Espirales. Rev. Multidiscip. Investig. científica., vol. 2, no. 17, 2018, doi: 10.31876/re.v2i17.269.

[25] J. P. Zumba Gamboa and C. León Arreaga, Cecibel Alexandra., "Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software.," INNOVA Res. J., vol. 3, no. 10, pp. 20–33, 2018, [Online]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=6777227