IMPLEMENTACIÓN BÁSICA DE UN SISTEMA SCADA CON RAPIDSCADA COMO ALTERNATIVA DE BAJO COSTO PARA ENTORNOS ACADÉMICOS

LISDEY FABIANA PEÑA BOHORQUEZ

ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

FACULTAD DE INGENIERÍA

UNIVERSIDAD DEL VALLE

SANTIAGO DE CALI

MARZO, 2026

IMPLEMENTACIÓN BÁSICA DE UN SISTEMA SCADA CON RAPIDSCADA COMO ALTERNATIVA DE BAJO COSTO PARA ENTORNOS ACADÉMICOS

LISDEY FABIANA PEÑA BOHORQUEZ

TRABAJO PRESENTADO PARA OPTAR POR EL TÍTULO DE INGENIERA ELECTRÓNICA

DIRECTOR
ASFUR BARANDICA LÓPEZ, M.SC

Grupo de Investigación Percepción y Sistemas Inteligentes

ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
FACULTAD DE INGENIERÍA
UNIVERSIDAD DEL VALLE
SANTIAGO DE CALI
MARZO, 2026

Tabla de contenido

LISTA DE TABLAS

LISTA DE ECUACIONES

LISTA DE EJEMPLOS

LISTA DE FIGURAS

Nota de Aceptación

Director

Jurado

Jurado

AGRADECIMIENTOS

El desarrollo de este trabajo de grado ha sido un proceso largo y exigente, marcado por retos académicos y personales, así como por momentos de aprendizaje y crecimiento. Su culminación no habría sido posible sin el apoyo y acompañamiento de las personas que, de distintas maneras, hicieron parte fundamental de este camino.

En primer lugar, agradezco profundamente a mi familia, en especial a mi madre y a su pareja, por su apoyo constante, su paciencia y por estar siempre presente en cada etapa de este proceso. Su confianza y respaldo fueron un pilar fundamental para continuar incluso en los momentos más difíciles. Agradezco también a mi padre y a mi hermana, por su acompañamiento permanente, su apoyo incondicional y por ser un soporte fundamental a lo largo de toda mi formación académica. De igual manera, expreso mi agradecimiento a mis tíos Genaro, Edilma, Adriana y Edwin, quienes con sus palabras de ánimo, apoyo y presencia contribuyeron de forma significativa a que pudiera avanzar y culminar este proyecto. El respaldo familiar recibido durante este proceso fue esencial para mantener la motivación y la constancia necesarias para alcanzar este logro académico.

De manera muy especial, agradezco a mi novio Steven, por su acompañamiento incondicional, por su apoyo constante y por recordarme, incluso en los momentos de mayor cansancio, las razones por las cuales inicié este camino. Asimismo, expreso mi agradecimiento a su familia, en particular a mi suegra Julia y a su abuela Mercedes, con quienes compartí gran parte de mi vida universitaria. Su acogida, apoyo y compañía fueron fundamentales en este proceso y contribuyeron de manera significativa a mi estabilidad personal y académica.

Expreso un agradecimiento especial a mi director de trabajo de grado, por la orientación académica brindada, por la oportunidad de desarrollar este proyecto y por su disposición permanente para acompañar, corregir y motivar a lo largo del proceso.

Finalmente, agradezco a todas las personas que, directa o indirectamente, contribuyeron a la realización de este trabajo y a la culminación de esta etapa académica.

Resumen

Este trabajo presenta la implementación de un sistema SCADA básico orientado a entornos educativos, desarrollado sobre la plataforma de código abierto RapidSCADA, para la supervisión y adquisición de datos mediante los protocolos Modbus TCP, MQTT y Modbus RTU. El desarrollo se estructuró en tres etapas: análisis funcional de la arquitectura de RapidSCADA, implementación de un entorno de simulación con Modbus TCP y MQTT, e integración física con dispositivos industriales y educativos empleando Modbus RTU.

El sistema se ejecutó sobre un entorno Windows y utilizó la base de datos SQLite integrada para el almacenamiento local de datos históricos. En la etapa de integración física se adquirieron señales de temperatura mediante termopares tipo K conectados al módulo ADAM-4018+, y se realizaron pruebas de escritura de señales analógicas de 0–20 mA sobre el módulo ADAM-4024. Adicionalmente, se incorporó un Arduino Nano como esclavo Modbus RTU, generando señales periódicas para contrastar el comportamiento entre hardware industrial y un nodo programable de bajo costo.

El desarrollo incluyó la configuración del sistema SCADA, la definición de líneas de comunicación, el mapeo de registros a canales internos, la creación de vistas en Webstation y la validación del flujo de datos en entornos simulados y físicos. Los resultados demuestran que RapidSCADA permite integrar de forma estable dispositivos industriales y educativos bajo una arquitectura SCADA unificada, constituyendo una alternativa replicable y de bajo costo para prácticas de laboratorio en ingeniería electrónica.

Palabras clave: *Automatización industrial, Modbus RTU, Modbus TCP, MQTT, RapidSCADA, Supervisión de procesos.*

Abstract

This work presents the implementation of a basic SCADA system oriented to educational environments, developed on the open-source platform RapidSCADA, for monitoring and data acquisition using the Modbus TCP, MQTT, and Modbus RTU protocols. The development was structured in three stages: functional analysis of the RapidSCADA architecture, implementation of a simulation environment using Modbus TCP and MQTT, and physical integration with industrial and educational devices using Modbus RTU.

The system was deployed on a Windows environment and used the integrated SQLite database for local storage of historical data. In the physical integration stage, temperature signals were acquired using K-type thermocouples connected to the ADAM-4018+ module, and analog output tests in the 0–20 mA range were performed on the ADAM-4024 module. Additionally, an Arduino Nano was incorporated as a Modbus RTU slave, generating periodic signals to compare the behavior between industrial hardware and a low-cost programmable node.

The development included SCADA system configuration, communication line definition, mapping of registers to internal channels, creation of monitoring views in Webstation, and validation of data flow in both simulated and physical environments. The results demonstrate that RapidSCADA enables stable integration of industrial and educational devices under a unified SCADA architecture, constituting a replicable and low-cost alternative for laboratory practices in electronic engineering.

***Keywords**: Industrial automation, Modbus RTU, Modbus TCP, MQTT, RapidSCADA, Process monitoring.*

1. Introducción

El control y la supervisión de procesos constituyen funciones habituales dentro de las aplicaciones de la ingeniería electrónica. En este contexto, los sistemas *SCADA* (*Supervisory Control and Data Acquisition*) representan un punto de encuentro entre la electrónica, la informática industrial y las comunicaciones. Su propósito no se limita a mostrar variables en una pantalla, también intervienen en la continuidad operativa, la seguridad funcional y el uso eficiente de los recursos de un sistema automatizado.

Desde la perspectiva de ingeniería, un *SCADA* cumple una función estratégica al centralizar la información proveniente de dispositivos de campo y habilitar decisiones (manuales o automáticas) apoyadas en datos en tiempo real. Colombo et al. [1] señalan que los sistemas de control actuales se inscriben en el paradigma de los sistemas ciberfísicos industriales (*ICPS*), en los cuales los procesos físicos y las infraestructuras digitales forman una unidad operativa capaz de adaptarse y optimizar su desempeño mediante el intercambio permanente de información.

En el ámbito educativo y de la investigación aplicada, los sistemas *SCADA* permiten comprender con mayor claridad la interacción entre sensores, actuadores, redes de comunicación industrial y aplicaciones de supervisión. Sin embargo, su uso en laboratorios universitarios suele estar condicionado por el costo de las licencias y del hardware especializado, así como por la complejidad de instalación y mantenimiento de las soluciones comerciales. Plataformas como *Ignition*[1], *Siemens WinCC*[2] o *AVEVA InTouch*[3] requieren inversiones que muchas instituciones no pueden asumir de manera sostenida, lo que limita la posibilidad de contar con entornos de supervisión completos durante la formación práctica.

Dentro de las alternativas de código abierto, *RapidSCADA* se ha consolidado como una herramienta robusta para la creación de sistemas *SCADA* e *IIoT* distribuidos. Su arquitectura modular, compuesta principalmente por los servicios ***Server***, ***Communicator*** y ***Webstation***, permite adquirir, procesar y visualizar datos empleando protocolos industriales como *Modbus RTU/TCP, OPC, MQTT,* entre otros. Además, su compatibilidad con bases de datos *SQL* y su distribución bajo licencia *Apache 2.0* facilitan su adopción con fines académicos y de investigación, sin incurrir en costos adicionales por licencias de uso [2].

En este trabajo de grado se plantea la implementación básica de un sistema *SCADA* utilizando *RapidSCADA*, orientado a la supervisión de procesos simulados e interacción con hardware físico mediante el protocolo *MQTT* y *Modbus* en sus variantes *RTU y TCP*. El proyecto busca demostrar la viabilidad técnica y formativa de utilizar software libre como alternativa de bajo costo para la enseñanza de la automatización industrial, fortaleciendo la comprensión práctica de los procesos de adquisición de datos, comunicación y visualización en tiempo real en un entorno académico controlado.

---

[1] *Inductive Automation*: https://inductiveautomation.com/pricing/ignition.
[2] *Siemens:* https://xcelerator.siemens.com/global/en/all-offerings/products/s/simatic-wincc-unified.html
[3] *Aveva:* https://www.aveva.com/es-es/products/intouch-hmi/

1.1 Contextualización del problema

Los sistemas *SCADA* constituyen hoy la base tecnológica de numerosos procesos en sectores como la generación y distribución de energía, el control de plantas industriales, los sistemas de agua potable, el transporte o las infraestructuras críticas. Su presencia en la industria hace que la comprensión de su funcionamiento sea un aspecto importante en la formación de ingenieros electrónicos y de profesionales vinculados al área de automatización.

En los programas de ingeniería, la enseñanza de estos sistemas suele abordarse desde los fundamentos teóricos del control y las comunicaciones industriales. No obstante, la transición desde los conceptos a la práctica se ve con frecuencia limitada por la falta de plataformas de supervisión disponibles en los laboratorios. La adquisición de licencias comerciales, la adecuación del *hardware* y la administración de los servidores *SCADA* representan costos que muchas instituciones no están en capacidad de asumir de manera continua, especialmente cuando se busca dotar varios laboratorios o permitir acceso simultáneo a múltiples estudiantes.

Como consecuencia, los estudiantes conocen el concepto de *SCADA*, pero no siempre logran experimentar el flujo completo de información: desde la lectura de una variable física en un módulo de adquisición de datos, pasando por la transmisión de esa información a través de un protocolo industrial, hasta su visualización y registro en una interfaz *HMI*. Esta brecha entre teoría y práctica limita la consolidación de competencias relacionadas con la integración de *hardware*, la configuración de protocolos y la interpretación de eventos y alarmas en tiempo real.

Diversas investigaciones han explorado alternativas para facilitar el trabajo práctico en laboratorios cuando no es posible disponer de plataformas comerciales. Minchala et al. [3] muestran que las arquitecturas *SCADA* abiertas permiten desarrollar ejercicios de monitoreo y control con estabilidad suficiente para actividades formativas, mientras que Mejía Rojas y Barbieri [4] evidencian la utilidad de integrar hardware de bajo costo mediante protocolos estándar. En conjunto, estos resultados indican que las soluciones abiertas reducen barreras económicas y permiten una aproximación más directa a los componentes internos de un sistema de supervisión.

En este contexto, *RapidSCADA* aparece como una alternativa adecuada para entornos académicos, dado que permite crear proyectos de supervisión completos sin requerir licencias propietarias y es compatible con protocolos de uso extendido en la industria, como *Modbus RTU/TCP/ASCII, OPC DA/AE/UA, MQTT, SNMP, SMTP, M-BUS*[4]. Su estructura modular y el uso de bases de datos *SQL* permiten reproducir el flujo típico de los sistemas de supervisión industrial dentro de un laboratorio universitario, facilitando la integración con microcontroladores y módulos de adquisición de bajo costo [2].

El panorama descrito pone de manifiesto la necesidad de contar con soluciones *SCADA* abiertas, funcionales y replicables que permitan a los estudiantes trabajar con escenarios cercanos a la realidad industrial, sin depender de plataformas comerciales. La ausencia de este tipo de herramientas limita tanto la adquisición de competencias técnicas como el

[4] *Connectivity and Integration in RapidSCADA* https://rapidscada.org/product/connectivity-and-integration/

desarrollo de proyectos aplicados en los cursos y trabajos de grado. En respuesta a esta necesidad, se propone la implementación de un sistema *SCADA* básico basado en *RapidSCADA*, orientado al monitoreo de variables simuladas y a la comunicación con hardware físico mediante *Modbus RTU/TCP* y *MQTT*, como alternativa formativa para fortalecer la enseñanza de la automatización y el control en entornos académicos.

1.2 Justificación académica

La formación en automatización, control y adquisición de datos ocupa un lugar importante en los programas de Ingeniería Electrónica e Ingeniería Eléctrica de la Universidad del Valle, donde varias asignaturas requieren que los estudiantes trabajen con señales reales, dispositivos de campo y herramientas de supervisión. En la Escuela se dispone de laboratorios especializados como el Laboratorio de Automática que está equipado con tarjetas de *National Instruments* y plataformas como *Matlab* y *LabVIEW*. Aunque estos espacios permiten desarrollar prácticas avanzadas, su uso está condicionado por el número limitado de equipos disponibles y por la necesidad de operar *software* bajo licenciamiento, lo que restringe su acceso simultáneo a varios grupos y la posibilidad de replicar actividades fuera del laboratorio.

En este contexto, contar con una herramienta de supervisión que pueda instalarse sin restricciones en los equipos institucionales y personales representa una oportunidad para ampliar las actividades prácticas asociadas a la adquisición de datos, la comunicación industrial y el diseño de interfaces *HMI*. La implementación de un sistema *SCADA* basado en *RapidSCADA* responde a esa necesidad, pues permite reproducir el flujo típico de supervisión utilizando hardware de bajo costo que ya forma parte de los ejercicios académicos de la Escuela, como microcontroladores y módulos de adquisición compatibles con Modbus.

De esta manera, el proyecto se justifica porque complementa las capacidades de los laboratorios existentes y ofrece un entorno de trabajo que los estudiantes pueden instalar, modificar y utilizar sin depender de licencias propietarias. Esto facilita su integración en asignaturas de formación como Sistemas Automáticos de Control, Electrónica, Medidas e Instrumentación, así como en proyectos de curso y trabajos de grado. Su adopción contribuye a fortalecer la articulación entre teoría y práctica, permitiendo que un mayor número de estudiantes experimente con escenarios de supervisión industrial acordes con su proceso formativo.

1.3 Objetivos

1.3.1 Objetivo General

Implementar un sistema *SCADA* básico utilizando el proyecto de código abierto *RapidSCADA*, para ser usado como alternativa de bajo costo en la supervisión y control de procesos en entornos educativos, integrando datos simulados y dispositivos físicos.

1.3.2 Objetivos Específicos

1. Documentar la estructura y funcionalidades básicas de *RapidSCADA*, enfocándose en sus módulos esenciales (***Server, Communicator, Webstation***) y protocolos de comunicación estándar para facilitar su adopción en entornos académicos.

2. Implementar una arquitectura *SCADA* básica en sistemas operativos *Windows* o *Linux*, utilizando datos simulados para validar la visualización de variables, gestión de alarmas y almacenamiento en bases de datos locales.
3. Desarrollar un módulo de comunicación básico que permita integrar dispositivos físicos mediante al menos dos protocolos abiertos, demostrando la interacción en tiempo real entre hardware y software *SCADA*.
4. Evaluar las ventajas y limitaciones de *RapidSCADA* frente a soluciones comerciales con el propósito de proyectar trabajos futuros a partir del esquema de *SCADA* básico.

1.4 Alcances y limitaciones del proyecto

El trabajo fue desarrollado en un entorno académico controlado y se orientó a la implementación básica de un sistema *SCADA* de bajo costo utilizando la plataforma *RapidSCADA*. Durante su ejecución se verificó la viabilidad técnica y formativa de esta herramienta para apoyar actividades de supervisión y control en un contexto educativo, integrando tanto variables simuladas como dispositivos físicos de bajo costo.

En términos técnicos, el alcance comprendió el diseño, configuración y validación funcional de los módulos principales de *RapidSCADA* (***Server, Communicator*** y ***Webstation***) operando en un esquema local de un solo servidor. Se estableció comunicación con dispositivos físicos conformados por los módulos *ADAM 4018+* y *ADAM 4024*, conectados mediante un bus *RS-485*, así como un microcontrolador *Arduino Nano* configurado como esclavo *Modbus RTU*. Adicionalmente, se verificó el funcionamiento del sistema para la gestión de alarmas básicas, el registro histórico en una base de datos *SQL* y la visualización de variables en una interfaz *HMI* accesible mediante navegador web.

Desde el punto de vista académico, el proyecto permitió disponer de una solución replicable que puede integrarse en actividades prácticas dentro de los laboratorios universitarios con recursos limitados. La implementación se limitó a una arquitectura local y a un número reducido de dispositivos; no obstante, los resultados obtenidos permitieron evaluar su comportamiento en escenarios reales de formación.

Es importante precisar que el alcance del proyecto se centró en la validación funcional del sistema, por lo que no se contempló la caracterización cuantitativa de su desempeño en términos de latencia ni de error en la transmisión de datos. La evaluación de este tipo de métricas requiere un diseño experimental específico y mecanismos de instrumentación temporal, los cuales no fueron considerados dentro de esta fase de implementación.

La Tabla 1.4.1 presenta los alcances y limitaciones identificados, organizados por categoría técnica y académica, con el fin de establecer de manera explícita el grado de cobertura de la solución desarrollada y las condiciones bajo las cuales fue evaluada. Esta síntesis permite visualizar el nivel de funcionalidad alcanzado durante la implementación, así como los aspectos que, por restricciones metodológicas o de infraestructura, no fueron considerados en esta etapa. Al estructurar los resultados en términos de arquitectura del sistema, comunicación, protocolos, base de datos, interfaz *HMI* y criterios de evaluación, se facilita la interpretación del desempeño obtenido y se define el marco operativo en el que funcionó el sistema. Asimismo, el análisis comparativo ofrece una referencia útil para actividades académicas posteriores que requieran comprender el alcance real del sistema y las condiciones bajo las cuales fue validado.

Tabla 1.4.1 Alcances y limitaciones del proyecto.
Fuente: Elaboración propia

Categoría	Aspecto	Descripción técnica y académica
Arquitectura del sistema	Alcance	Se implementó una arquitectura *SCADA* básica utilizando *RapidSCADA v6.4*, organizada en los módulos ***Server, Communicator*** y ***Webstation***, lo que permitió la adquisición, supervisión y visualización de datos en tiempo real dentro de un entorno local.
	Limitación	La arquitectura se validó únicamente en un entorno monousuario, sin pruebas en redes distribuidas ni acceso simultáneo desde múltiples clientes remotos.
Hardware y comunicación	Alcance	Se integraron los módulos *ADAM 4018+* y *ADAM 4024* como dispositivos esclavos *Modbus RTU* dentro de una misma línea *RS-485*, conectados mediante un enlace *USB-Serial*. Ambos dispositivos compartieron el mismo canal de comunicación serial en RapidSCADA y se configuraron con direcciones Modbus independientes (*4018+* en dirección 5 y *4024* en dirección 1). También se incorporó un Arduino Nano configurado como esclavo *Modbus RTU* como parte de las actividades de integración educativa.
	Limitación	La implementación se restringió a hardware de bajo costo y módulos industriales de pequeña escala, sin incluir *PLCs* ni equipos que operan con protocolos utilizados en entornos de automatización de planta, como *Ethernet/IP, Profibus* o *Profinet*.
Protocolos y software	Alcance	Se implementaron los protocolos *Modbus RTU, Modbus TCP* y *MQTT*. *Modbus TCP* se verificó mediante el simulador *ModRSsim2* y *Modbus RTU* con los módulos *ADAM* y un *Arduino Nano*. Para *MQTT* se utilizó el simulador *MQTT Explorer.*
	Limitación	Las pruebas con *Modbus RTU/TCP* se realizaron en un entorno local con pocos dispositivos, sin evaluar condiciones de carga ni redes distribuidas. *MQTT* se utilizó únicamente para recepción básica de datos en formato *JSON*, sin análisis de QoS ni múltiples clientes. Las pruebas con *OPC UA/DA* fueron exploratorias y no se documentaron. No se realizó caracterización cuantitativa de latencia ni de error de transmisión, debido a la ausencia de instrumentación temporal y de un diseño experimental específico.
Base de datos y registro histórico	Alcance	Se habilitó el almacenamiento local de variables en una base de datos *SQL*, permitiendo la generación de historiales y la verificación del comportamiento de los registros desde ***Webstation***.
	Limitación	No se emplearon bases de datos distribuidas ni servicios en la nube. La solución mantuvo los registros en un servidor local.
Interfaz HMI y visualización	Alcance	Se desarrollaron pantallas *HMI* utilizando vistas tipo *Table* y vistas de tipo *Schematic* en ***Webstation***, representando de forma gráfica las variables y comandos asociados a los módulos ADAM 4018+ y 4024, priorizando claridad y operatividad para fines educativos.
	Limitación	El diseño *HMI* se mantuvo en un nivel básico, sin animaciones ni elementos avanzados presentes en plataformas comerciales como *Ignition, WinCC* o *InTouch.*
Evaluación académica	Alcance	El sistema fue documentado y validado mediante actividades prácticas realizadas en un entorno académico, incluyendo sesiones demostrativas con estudiantes, lo que permitió evidenciar su utilidad como herramienta formativa.
	Limitación	No se aplicaron instrumentos de evaluación pedagógica formal ni métricas cuantitativas de aprendizaje, por lo que la validación corresponde únicamente a criterios técnicos y de aplicabilidad.
Seguridad y escalabilidad	Alcance	Se configuraron niveles básicos de usuario y autenticación en ***Webstation***, garantizando acceso controlado al sistema local.
	Limitación	No se implementaron mecanismos avanzados de ciberseguridad industrial, como cifrado (*TLS*), segmentación de red o esquemas de redundancia, ni monitoreo de eventos de seguridad.

1.5 Estructura del documento

El trabajo se organizó en capítulos que siguen una secuencia lógica desde la identificación del problema hasta la validación del sistema implementado y el análisis académico de sus resultados. Esta estructura facilita el desarrollo progresivo de los fundamentos teóricos, el diseño metodológico y la aplicación práctica de la propuesta.

- **Capítulo 1. Introducción:** Presentó la contextualización del problema, la justificación académica, los objetivos del proyecto, así como sus alcances, limitaciones y la estructura general del documento. Este capítulo estableció el marco que motiva la implementación de un sistema *SCADA* de libre acceso en entornos educativos.
- **Capítulo 2. Fundamentación teórica y marco conceptual de los sistema *SCADA*:** Expuso los conceptos fundamentales sobre los sistemas *SCADA*, su evolución y estructura funcional. Analiza los modelos de software comercial y de código abierto, profundiza en los protocolos *Modbus RTU/TCP* y MQTT como estándar de comunicación industrial adoptados y justifica la selección de *RapidSCADA* como plataforma base. Además, revisa antecedentes académicos relacionados con el uso de sistemas SCADA libres en contextos educativos.
- **Capítulo 3. Arquitectura funcional y plataforma tecnológica de *RapidSCADA*:** Describe la estructura interna de la plataforma, sus módulos principales (***Server, Communicator y Webstation***), el flujo de interacción entre ellos, el modelo de datos y las posibilidades de extensión mediante *scripting* en *C#.* También aborda los protocolos soportados relevantes para el proyecto, así como aspectos básicos de seguridad, redundancia y almacenamiento, para concluir con un análisis técnico de la herramienta.
- **Capítulo 4. Sistema *SCADA* básico simulado con *Modbus TCP* y *MQTT*:** Presenta el diseño e implementación de un proyecto *SCADA* simulado utilizando los protocolos *Modbus TCP y MQTT*. Se expuso la metodología seguida, la configuración del entorno y del dispositivo simulado, la creación de canales y plantillas, el desarrollo de vistas en ***Webstation*** y la habilitación de alarmas y registro histórico. Finalmente, se presentan los resultados de validación funcional obtenidos en este escenario.
- **Capítulo 5. Integración con hardware físico mediante *Modbus RTU*:** Detalla la conexión de *RapidSCADA* con dispositivos físicos a través de *Modbus RTU*. Se describen el hardware utilizado (módulos *ADAM 4018+/4024* y *Arduino Nano*), la topología *RS-485*, la configuración de líneas, dispositivos y canales en *RapidSCADA*, y las pruebas de lectura y escritura realizadas. Se presentan por separado los resultados del proyecto con módulos *ADAM* y del proyecto complementario con *Arduino*, y se discuten las observaciones obtenidas en ambos casos.
- **Capítulo 6. Conclusiones y recomendaciones:** Presenta las conclusiones generales del proyecto, organizadas en función de los objetivos planteados, y propone recomendaciones para la adopción de *RapidSCADA* en laboratorios de ingeniería. Asimismo, sugiere líneas de trabajo futuro relacionadas con la expansión hacia arquitecturas distribuidas, incorporación de nuevos protocolos o integración con plataformas *IoT*.

Finalmente, se incluyen las **referencias bibliográficas**, citadas en formato *IEEE*, y los **anexos**, en los que se recopilan configuraciones, diagramas, fragmentos de código, capturas de pantalla y documentación complementaria asociada al sistema desarrollado.

2. FUNDAMENTACIÓN TEÓRICA Y MARCO CONCEPTUAL DE LOS SISTEMA *SCADA*

2.1 Concepto y evolución de los sistemas *SCADA*

Los sistemas *SCADA* (*Supervisory Control and Data Acquisition*), conocidos en español como Supervisión, Control y Adquisición de Datos, constituyen una arquitectura tecnológica ampliamente adoptada en la automatización industrial. Su función principal es permitir que los operadores supervisen y actúen sobre procesos físicos que pueden estar distribuidos geográficamente, a partir de la adquisición de datos en tiempo real desde el entorno operativo. Esta capacidad de supervisión remota resulta relevante en sectores donde la continuidad, la eficiencia y la seguridad de los procesos son críticas, como ocurre en la energía, el agua, la industria química y la manufactura[5].

En términos generales, un sistema *SCADA* integra componentes de *hardware* y *software* orientados a capturar información desde dispositivos de campo, visualizar su comportamiento en interfaces gráficas, registrar el historial operativo y ejecutar respuestas automáticas ante condiciones anómalas. La evolución de la informática industrial ha llevado estos sistemas desde soluciones rígidas y fuertemente cableadas hacia plataformas abiertas y escalables, capaces de integrarse con redes industriales, sistemas empresariales y tecnologías asociadas al Internet Industrial de las Cosas (*IIoT*)[6].

Función general y componentes principales de un sistema SCADA

El *SCADA* actúa como un puente entre el entorno físico de la planta y los sistemas digitales de supervisión y control. Para cumplir con esto, se apoya en varios elementos funcionales que interactúan de manera jerárquica y coordinada, tal como se muestra en la Figura 2.1.1, donde se representa la arquitectura general y el flujo de información entre los diferentes niveles de un sistema *SCADA*:

- ***Sensores y actuadores:*** dispositivos que interactúan directamente con el proceso. Los sensores capturan variables como nivel, presión o temperatura, mientras que los actuadores ejecutan acciones físicas como el accionamiento de bombas o la apertura de válvulas[7].
- ***Controladores (PLC/RTU):*** reciben señales de campo, ejecutan la lógica de control y envían órdenes a los actuadores. Los *PLC* se emplean con mayor frecuencia en procesos locales, mientras que las *RTU* se aplican en sistemas más distribuidos.

[5] Fortinet - ¿Qué es SCADA y el sistema SCADA? https://www.fortinet.com/lat/resources/cyberglossary/scada-and-scada-systems

[6] ¿Qué es un Sistema SCADA? El Corazón de la Automatización Industrial https://es.ubidots.com/blog/what-is-scada/

[7] Sistemas SCADA ¿Qué es un sistema SCADA? https://upkeep.com/es/learning/scada-systems-explained/

Familias como *Siemens S7* son ejemplos habituales en combinación con diversas plataformas *SCADA*.

- ***Red de comunicaciones:*** conecta los componentes del sistema mediante enlaces seriales (*RS-232, RS-485*) o *Ethernet* industrial. Sobre esta infraestructura se transportan los datos utilizando protocolos estandarizados como *Modbus, DNP3* u *OPC UA*, permitiendo la interoperabilidad entre dispositivos de distintos fabricantes[7].
- ***Interfaz Humano-Máquina (HMI):*** facilita la visualización del estado del proceso, la confirmación de alarmas y la modificación de consignas. Aunque a menudo se asocia al concepto de *SCADA*, la *HMI* representa solo una parte de la solución global.
- ***Servidor SCADA y base de datos histórica:*** procesan, almacenan y distribuyen la información a las diferentes estaciones cliente. En instalaciones de menor tamaño estos componentes pueden residir en un mismo equipo; en sistemas de mayor escala se implementan arquitecturas distribuidas y esquemas de redundancia[5].

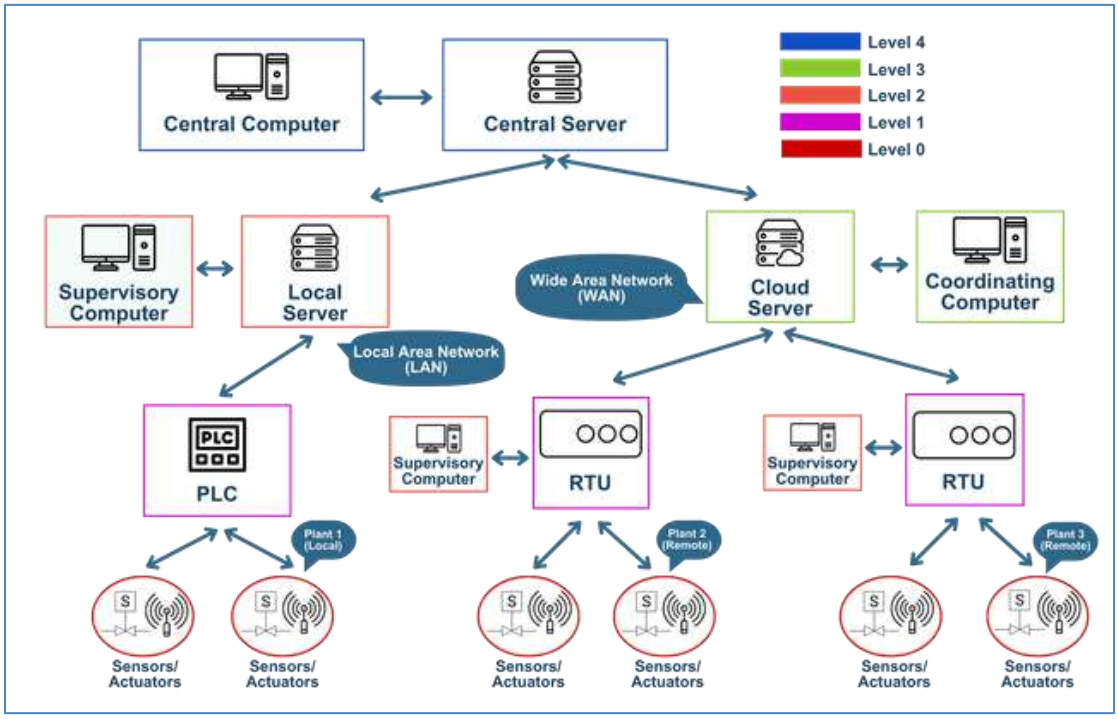


Figura 2.1.1 Arquitectura y diagrama de bloques de un sistema SCADA.
Fuente: Empowered Automation Solutions Inc.[8]

El funcionamiento conjunto de estos elementos permite disponer de una vista consolidada del proceso, detectar condiciones críticas y ejecutar acciones de control oportunas, incluso desde ubicaciones remotas.

[8] Empowered Automation Solutions Inc. https://www.empoweredautomation.com/scada-system

Evolución histórica de los sistemas SCADA

Desde la década de 1960, la evolución de los sistemas *SCADA* ha ido de la mano con el avance de la electrónica, la informática y las comunicaciones.

Entre los hitos más relevantes se encuentran:

- ***1960s – Automatización por relés:*** los primeros sistemas de control utilizaban paneles de relés electromecánicos, con capacidades limitadas pero suficientes para establecer esquemas iniciales de supervisión remota[9].
- ***1968 – Aparición del primer PLC (*Modicon 084):** la sustitución del cableado por lógica programable facilitó la reconfiguración de procesos y sentó las bases para el desarrollo posterior de los SCADA[10].
- ***1970s – Primeros SCADA centralizados:*** se introdujeron sistemas con computadoras centrales que monitoreaban y controlaban procesos desde una única estación, sin integración entre plantas[9].
- ***1979 – Nacimiento del protocolo Modbus:*** *Modicon* presentó *Modbus* como protocolo maestro-esclavo, que se consolidó como estándar de comunicación por su sencillez y compatibilidad con múltiples equipos[11].
- ***1980s – Arquitectura distribuida y HMI en PC:*** el uso de redes *LAN* y computadores personales permitió conectar varios *PLC* y desarrollar las primeras interfaces gráficas de operación[9].
- ***1990s – Protocolos abiertos y redes extendidas:*** la normalización de estándares como *DNP3, OPC* e *IEC 60870* favoreció la interoperabilidad entre fabricantes y posibilitó la supervisión a través de redes *WAN* [5].
- ***2000s – Conectividad web y ciberseguridad:*** la migración hacia infraestructuras *TCP/IP* habilitó el acceso a los *SCADA* mediante navegadores *web*, pero también hizo necesaria la incorporación de medidas específicas de ciberseguridad [6].
- ***2010s – IoT e Industria 4.0:*** los *SCADA* se conectan con sensores inteligentes, servicios en la nube y herramientas de análisis avanzado, formando parte de ecosistemas de fabricación inteligente[9].
- ***2020s – Inteligencia artificial y asistentes virtuales:*** se generaliza el uso de servidores virtuales, contenedores y plataformas abiertas, y comienzan a explorarse formas de interacción basadas en asistentes de voz y servicios cognitivos [6].

Esta transición desde sistemas cableados y cerrados hacia plataformas conectadas y flexibles ha permitido que los *SCADA* se utilicen tanto en entornos industriales complejos como en contextos académicos y de bajo costo. Es precisamente esta evolución la que

---

[9] Historia de los SCADAs: un repaso a la evolución desde sus inicios hasta la actualidad https://www.aydep.com/blog/2023/11/24/historia-de-los-scadas-un-repaso-a-la-evolucion-de-los-scadas-desde-sus-inicios-hasta-la-actualidad/

[10] ¿Cuál es la historia del PLC? https://upkeep.com/es/learning/history-of-the-plc/

[11] Protocolo de comunicaciones en serie https://e-robot-latam.com/modbus/

posibilita el desarrollo del proyecto formativo planteado en este trabajo de grado, en el que se recurre a herramientas abiertas para reproducir el comportamiento de un sistema *SCADA* sin depender de infraestructura propietaria.

2.2 Estructura funcional y principios de operación

Los sistemas *SCADA* se construyen a partir de una arquitectura modular que organiza las funciones de supervisión y control en diferentes niveles. Esta estructura permite distribuir responsabilidades, facilitar el mantenimiento del sistema y asegurar que cada componente cumpla un rol específico dentro del proceso de adquisición de datos y operación. De manera general, la arquitectura se concibe en tres capas funcionales referenciadas en la literatura técnica especializada [5, 6].

1. ***Capa de adquisición de datos:*** en este nivel se agrupan los dispositivos de campo: sensores, actuadores y los controladores industriales a los que se conectan, como *PLC* o *RTU*. Los sensores capturan información del proceso (por ejemplo, temperatura, presión o nivel), mientras que los actuadores ejecutan acciones físicas asociadas al control del sistema. Los controladores consolidan estas señales, ejecutan la lógica programada localmente y preparan los datos para ser transmitidos hacia el nivel superior del *SCADA*. La comunicación suele implementarse mediante protocolos industriales estandarizados[12].
2. ***Capa de procesamiento y almacenamiento:*** Corresponde al servidor *SCADA* y a las bases de datos donde se registran los valores históricos. En esta capa se realiza el procesamiento central: se integran las señales provenientes de los controladores, se aplican fórmulas o conversiones necesarias, se evalúan condiciones para activar alarmas, se generan eventos y se almacenan los registros para su análisis posterior. Esta capa constituye el núcleo lógico del sistema, ya que administra la información en tiempo real y coordina la distribución de datos hacia las interfaces de usuario.
3. ***Capa de supervisión:*** Incluye las interfaces *HMI* y demás aplicaciones de visualización mediante las cuales los operadores interactúan con el sistema. Desde estas pantallas se monitorea el comportamiento del proceso, se reconocen alarmas, se consultan tendencias históricas y, cuando es requerido, se envían comandos hacia los equipos de campo (como modificar consignas o habilitar el funcionamiento de dispositivos). Esta capa es la responsable de presentar la información de forma clara y accesible, lo que facilita la toma de decisiones informada.

Principios de operación del sistema SCADA

En la mayoría de las implementaciones, el sistema *SCADA* opera bajo un esquema cliente-servidor. Los controladores transmiten datos hacia el servidor, ya sea de manera periódica o en respuesta a consultas explícitas. El servidor procesa la información recibida, la

[12] Qué es el sistema SCADA y para qué sirve https://www.mgelectricidad.es/que-es-el-sistema-scada-y-para-que-sirve/

almacena, la compara con condiciones previamente definidas y, posteriormente, la redistribuye a las estaciones cliente o a las interfaces *HMI*. Cuando una variable sobrepasa un rango permitido o se identifica una condición crítica, el sistema genera una alarma que se comunica al operador mediante señales visuales o sonoras[13].

Este modelo de operación facilita la integración del *SCADA* con otros niveles de la pirámide de automatización industrial, como los sistemas *MES* (*Manufacturing Execution Systems*) y *ERP* (*Enterprise Resource Planning*), tal como lo señalan distintos autores en el contexto de dicha pirámide [5]. Gracias a esta estructura, el sistema no solo contribuye a la supervisión operativa en tiempo real, sino que también fortalece la trazabilidad del proceso y el análisis de su desempeño a lo largo del tiempo.

2.3 Modelos de software *SCADA*: comerciales y de código abierto

El software SCADA puede agruparse en dos modelos principales según su esquema de licenciamiento: soluciones comerciales y plataformas de código abierto. Cada enfoque ofrece beneficios y restricciones distintos, por lo que su elección depende de factores como el presupuesto, la escala del proyecto y el nivel de flexibilidad requerido. En la práctica educativa y en proyectos de pequeña escala, esta distinción determina en gran medida la accesibilidad de las herramientas y su posibilidad de adaptación.

2.3.1 *SCADA* comerciales

Las plataformas comerciales son desarrolladas por fabricantes consolidados de la industria de automatización y requieren la adquisición de licencias que, por lo general, se dimensionan según el número de etiquetas (*tags*), estaciones cliente o servidores habilitados. Su atractivo radica en que ofrecen herramientas avanzadas integradas, soporte oficial del fabricante y ecosistemas completos de historización, diagnósticos y desarrollo de interfaces gráficas. No obstante, este nivel de integración suele implicar costos elevados y una dependencia significativa del proveedor[14][15]. Entre las plataformas más representativas se encuentran:

- ***AVEVA InTouch (antes Wonderware):*** uno de los *HMI/SCADA* más utilizados en entornos industriales. Destaca por su biblioteca gráfica y su integración con el portafolio *AVEVA*.
- ***Ignition (Inductive Automation):*** solución moderna basada en arquitectura web, con un servidor central (*Gateway*) desde el cual se administran clientes y *tags* ilimitados. Su compatibilidad con múltiples protocolos lo hace atractivo en proyectos que requieren escalabilidad.

[13] Estructura de un sistema SCADA y sus componentes https://www.geprom.com/estructura-de-un-sistema-scada-y-sus-componentes/
[14] Sistemas de control de software libre https://www.incibe.es/incibe-cert/blog/sistemas-control-software-libre
[15] Ignition vs Wonderware - Parte 1 https://corsosystems.com/posts/ignition-vs-wonderware-part-1

- ***Siemens SIMATIC WinCC:*** integrado en *TIA* Portal, está orientado a sistemas que emplean *PLC Siemens* y es habitual en aplicaciones industriales donde se exige confiabilidad y continuidad operativa[16].

2.3.2 *SCADA* de código abierto

Las plataformas de código abierto eliminan los costos de licenciamiento y ofrecen al usuario la posibilidad de revisar, modificar y extender sus componentes. En los últimos años, han ganado espacio tanto en laboratorios académicos como en aplicaciones industriales de pequeña y mediana escala, gracias a su accesibilidad y adaptabilidad[1417]. Entre las más utilizadas se encuentran:

- ***RapidSCADA:*** plataforma desarrollada sobre *.NET*, compuesta por ***Server***, ***Communicator*** y ***Webstation***. Soporta *Modbus RTU/TCP* y *MQTT* de manera nativa, y dispone de módulos adicionales para ampliar sus capacidades. Su disponibilidad en *Windows* y *Linux*, junto con su licencia *Apache* 2.0, la convierten en una herramienta flexible para formación e investigación [7].
- ***ScadaBR:*** proyecto brasileño basado en *Java*, con compatibilidad para múltiples protocolos y bases de datos. Su enfoque educativo y comunidad activa lo han convertido en una opción frecuente en laboratorios universitarios[19].
- ***OpenSCADA:*** plataforma modular orientada a proyectos que requieren altos niveles de personalización, con soporte para protocolos como *DNP3* e *IEC 60870-5*[18].

2.3.3 Comparación técnica de plataformas *SCADA*

La comparación sintetizada en la Tabla 2.3.1 permite observar diferencias significativas entre ambos modelos. Las soluciones comerciales concentran capacidades avanzadas de soporte, estabilidad y herramientas integradas orientadas a entornos industriales, mientras que las plataformas abiertas destacan por su flexibilidad, accesibilidad y pertinencia en contextos académicos.

Las plataformas consideradas fueron seleccionadas por su relevancia en escenarios educativos y por la disponibilidad de documentación técnica accesible. *RapidSCADA*, *ScadaBR* y *OpenSCADA* representan soluciones de código abierto utilizadas en programas de ingeniería para prácticas de supervisión y adquisición de datos, mientras que *WinCC* e *Ignition* se incluyen como referentes de soluciones comerciales ampliamente utilizadas en el ámbito industrial. Esta selección permite establecer una comparación equilibrada entre requerimientos académicos y características operativas propias de sistemas *SCADA* industriales.

---

[16] Sistema unificado de *SIMATIC WinCC* https://www.siemens.com/es/es/productos/automatizacion/sistemas/simatic/hmi/wincc-unified.html
[17] *RapidSCADA* – Introduction https://rapidscada.net/docs/en/latest/software-overview/introduction/
[18] *OpenSCADA* – Página oficial http://oscada.org/en/main/

Tabla 2.3.1 Comparación entre plataformas *SCADA* comerciales y de código abierto
Fuente: Elaboración Propia

Plataforma	Licencia	Sistema operativo	Protocolos soportados	Soporte técnico	Aplicabilidad educativa
AVEVA InTouch[3]	Propietaria comercial (licencia de pago)	*Windows*	*OPC DA/UA, Modbus TCP, DNP3, BACnet*	Oficial	Baja
Ignition[1]	Propietaria comercial (licencia de pago)	*Windows/ Linux y macOS*	*OPC UA, MQTT, Modbus, BACnet*	Oficial	Moderada
WinCC (Siemens)[2]	Propietaria comercial (licencia de pago)	*Windows*	*S7, OPC DA/UA, Modbus TCP*	Oficial	Baja
RapidSCADA[17]	*Apache License 2.0*	*Windows/ Linux*	*Modbus RTU/TCP, MQTT, OPC UA/DA, SNMP*	Comunidad (activa)	Alta
ScadaBR[19]	GNU General Public License (GPL)	*Windows/ Linux*	*Modbus, OPC, BACnet, DNP3*	Comunidad (activa)	Alta
OpenSCADA[18]	GNU General Public License (GPL)	*Linux/ Sistemas tipo Unix*	*Modbus, DNP3, IEC 60870-5*	Comunidad (media)	Media

La clasificación presentada en la Tabla 2.3.1 se realizó con base en la documentación oficial y los repositorios de cada plataforma, diferenciando entre *software* propietario con licenciamiento comercial y *software* de código abierto bajo licencias reconocidas por la *Open Source Initiative,* tales como *Apache License 2.0* y *GNU General Public License*. Asimismo, la aplicabilidad educativa se evaluó considerando criterios como el esquema de licenciamiento, la facilidad de instalación, la disponibilidad de documentación técnica, la posibilidad de uso sin restricciones económicas y la compatibilidad con *hardware* de bajo costo.

2.4 Protocolo *Modbus*: estándar de comunicación industrial

En el ámbito de la automatización industrial, *Modbus* se ha consolidado como uno de los protocolos de comunicación más extendidos desde su introducción en 1979 por la empresa *Modicon*. Su estructura sencilla, su confiabilidad y su amplia compatibilidad con dispositivos de distintos fabricantes han favorecido su adopción como estándar en el intercambio de datos entre controladores, módulos de *E/S* y sistemas *SCADA*[20].

[19] *SCADABR* – Página oficial https://scadabr.org/
[20] *MODBUS*: El Protocolo de Comunicación que Sigue Liderando la Automatización Industrial https://www.etkho.com/modbus-el-protocolo-de-comunicacion-que-sigue-liderando-la-automatizacion-industrial/

Modbus opera bajo una arquitectura maestro-esclavo o cliente-servidor [5]. En este esquema, un dispositivo maestro (como un servidor *SCADA*) envía consultas a uno o varios dispositivos esclavos, por ejemplo *PLCs*, módulos de adquisición o sensores inteligentes, los cuales responden con los datos solicitados o con la confirmación de una escritura. Esta organización secuencial simplifica la interoperabilidad y facilita el diagnóstico de fallas en la red[21].

A lo largo de su evolución, *Modbus* se ha adaptado a diferentes medios físicos y topologías. En este proyecto se emplean dos de sus variantes más difundidas: *Modbus RTU*, orientado a comunicación serial, y *Modbus TCP,* basado en redes Ethernet.

2.4.1 *Modbus RTU*

Modbus RTU (*Remote Terminal Unit*) constituye la variante clásica del protocolo y opera sobre enlaces seriales como *RS-232* o, con mayor frecuencia, *RS-485*. Utiliza una codificación binaria compacta que optimiza el tamaño de las tramas y mejora la eficiencia del canal de comunicación. El funcionamiento de este protocolo se basa en un esquema maestro–esclavo, el cual se ilustra en la Figura 2.4.1.

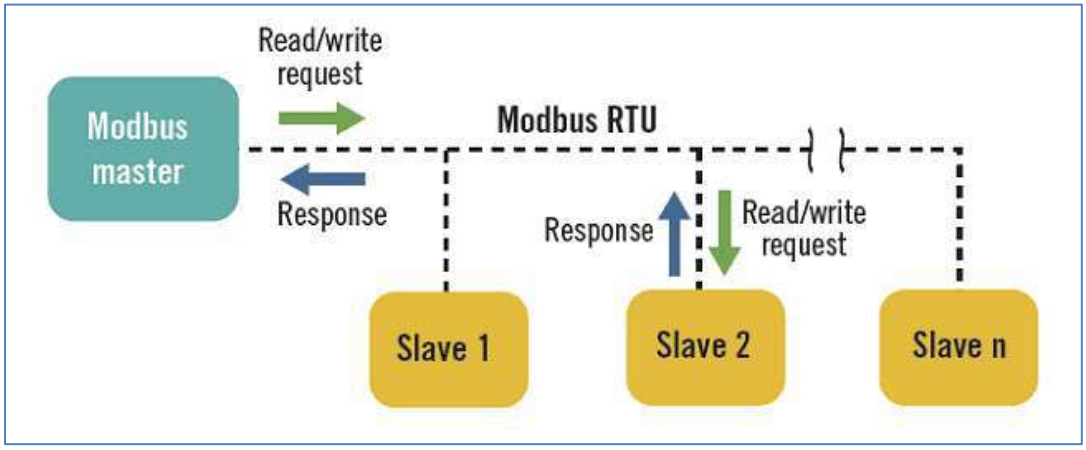


Figura 2.4.1 Esquema del modelo de comunicación maestro–esclavo en una red Modbus RTU.
Fuente: Altamirano, S. "Modbus RTU". Escuela Control Más, disponible en: https://blog.escuelacontrolmas.com/modbus-rtu/

En una red *Modbus RTU*, un maestro interroga secuencialmente a cada dispositivo esclavo conectado al bus. Cada esclavo posee una dirección única, lo que permite integrar múltiples equipos sobre la misma línea física. El estándar contempla hasta 247 direcciones esclavas [5, 6], aunque en la práctica este valor se ve limitado por la longitud del cableado, la calidad de la instalación y las características eléctricas del medio[22]. *RS-485* es el medio

[21] Implementación de *Modbus* en sistemas de automatización industrial https://aumaiberia.com/modbus-en-sistemas-de-automatizacion-industrial/
[22] Modbus RTU vs. Modbus TCP/IP: ¿Qué protocolo elegir? https://www.trugemtech.com/es/modbus-rtu-vs-modbus-tcp-ip/

predominante para Modbus *RTU* en entornos industriales por su capacidad diferencial, inmunidad al ruido, posibilidad de conexión multipunto y alcance de cientos de metros. Estas propiedades lo hacen adecuado para integrar módulos de adquisición distribuidos en campo, como los empleados en este proyecto (*ADAM 4018+ y ADAM 4024*).

2.4.2 *Modbus TCP*

Modbus TCP (también conocido como *Modbus TCP/IP*) adapta el protocolo original a infraestructuras *Ethernet*. En esta variante, las tramas Modbus se encapsulan dentro de paquetes *TCP*, lo que permite aprovechar redes existentes, *switches* industriales y servicios de enrutamiento *IP*. El esquema general de comunicación cliente–servidor y la integración de dispositivos *Modbus TCP* sobre *Ethernet* se ilustran en la Figura 2.4.2.

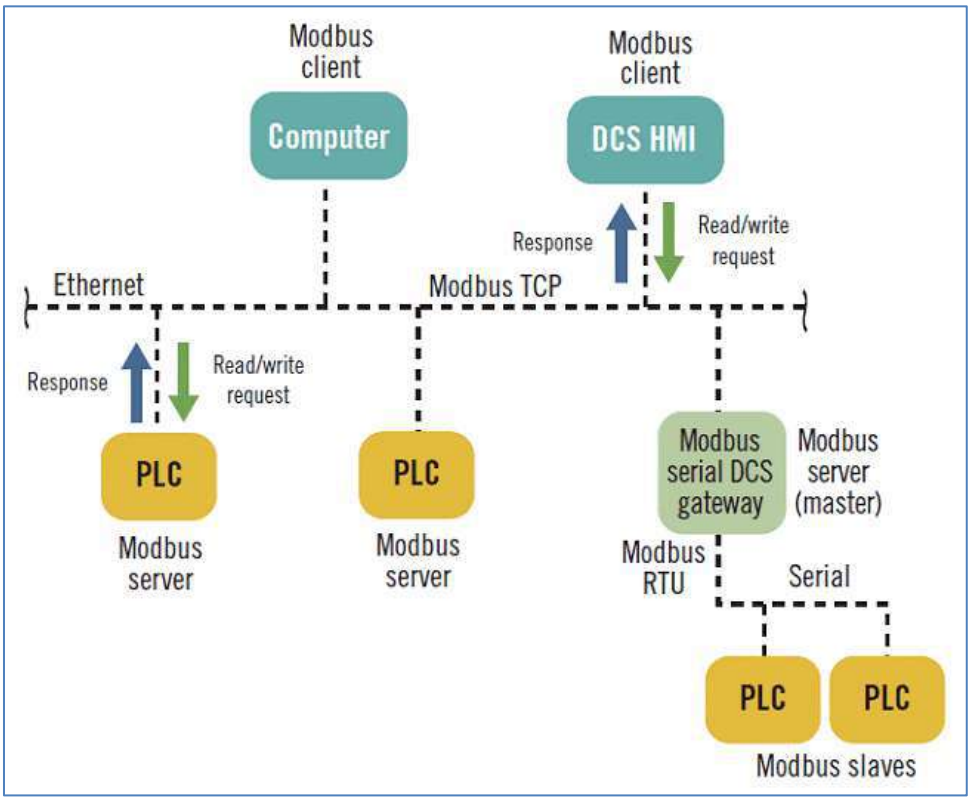


Figura 2.4.2 Esquema de comunicación Modbus TCP sobre Ethernet con integración hacia dispositivos Modbus RTU mediante pasarela.
Fuente: Chongqing Blue Jay Technology Co., Ltd. "Difference Between Modbus TCP and RTU". LinkedIn, 2023, disponible en: https://www.linkedin.com/pulse/difference-between-modbus-tcp-rtu-blue-jay-technology-co-ltd-zryoe/

El uso de *TCP* habilita comunicación *full-dúplex*, mecanismos propios de control de errores y mayores velocidades de transmisión en comparación con los enlaces seriales tradicionales. Adicionalmente, el direccionamiento basado en *IP* facilita la interoperabilidad con redes corporativas, *VPN* y entornos distribuidos, permitiendo integrar dispositivos Modbus dentro de infraestructuras de *TI* modernas[2122].

En el marco del presente trabajo de grado, *Modbus TCP* se emplea como una etapa inicial de validación del sistema *SCADA* mediante la simulación de variables, utilizando un *software* emulador que actúa como dispositivo esclavo. Esta estrategia permite verificar la arquitectura, la comunicación y el flujo de datos del sistema sin requerir inicialmente *hardware* físico, antes de proceder a la integración con dispositivos reales de campo, la cual se realiza posteriormente mediante *Modbus RTU*.

2.4.3 Comparación entre *Modbus RTU* y *Modbus TCP*

Para sintetizar las diferencias relevantes entre ambas variantes y facilitar su aplicación en escenarios educativos de bajo costo, se presenta la comparación de la Tabla 2.4.1.

Tabla 2.4.1 Comparación técnica entre *Modbus RTU y Modbus TCP*.
Fuente: Elaboración propia

Criterio	***Modbus RTU***	***Modbus TCP***
Medio físico	*RS-485* (bus multipunto)	*Ethernet (TCP/IP)*
Dirección / direccionamiento	Dirección de esclavo (1–247)	Dirección *IP* + puerto *TCP* (502)
Topología habitual	Bus compartido maestro–esclavo	Red conmutada en estrella (*switches*)
Velocidad / latencia	Menor velocidad, latencia estable	Mayor velocidad, latencia variable
Alcance típico	Cientos de metros por cable *RS-485*	Alcance de red *IP* existente (*LAN/VLAN/VPN*)
Nivel de infraestructura requerido	Mínimo (convertidor *USB–RS485*)	Requiere red IP funcional
Robustez frente a ruido	Alta (diferencial *RS-485*)	Dependiente de la calidad de la red
Casos de uso en este trabajo de grado	Integración con *ADAM 4018+, ADAM 4024, Arduino*	Simulación con *ModRSsim2* en entorno de laboratorio
Ventaja principal	Bajo costo y fácil integración con hardware de campo	Acceso remoto y escalabilidad en entornos *IP*
Limitación principal	Alcance y número de nodos limitados	Dependencia de la infraestructura de red

En el contexto académico de este proyecto, *Modbus* resulta especialmente adecuado porque combina una estructura de mensajes sencilla, abundante documentación y amplia disponibilidad en dispositivos de bajo costo. Su presencia tanto en módulos industriales comerciales (*ADAM 4018+/4024*) como en bibliotecas para microcontroladores de uso educativo (*Arduino*) permite reproducir escenarios cercanos a la práctica industrial sin requerir infraestructura compleja ni costosa. Estas características lo convierten en un protocolo idóneo para introducir a los estudiantes en la comunicación maestro-esclavo y en la integración de *hardware* con sistemas *SCADA*.

2.5 Protocolo *MQTT* en sistemas *SCADA*

El protocolo *Message Queuing Telemetry Transport (MQTT)* es un estándar de mensajería que opera bajo un modelo *publish/subscribe* gestionado por un intermediario conocido como *broker*. De acuerdo con su documentación oficial[23], los nodos publicadores envían mensajes a temas (*topics*) específicos, y los nodos suscriptores reciben esa información sin establecer comunicación directa entre sí. El protocolo utiliza un formato de mensaje sencillo y niveles de servicio (*Quality of Service*) que regulan la fiabilidad en la entrega. El modelo de comunicación *publish/subscribe* y el rol del *broker* en la distribución de mensajes se ilustran en la Figura 2.5.1.

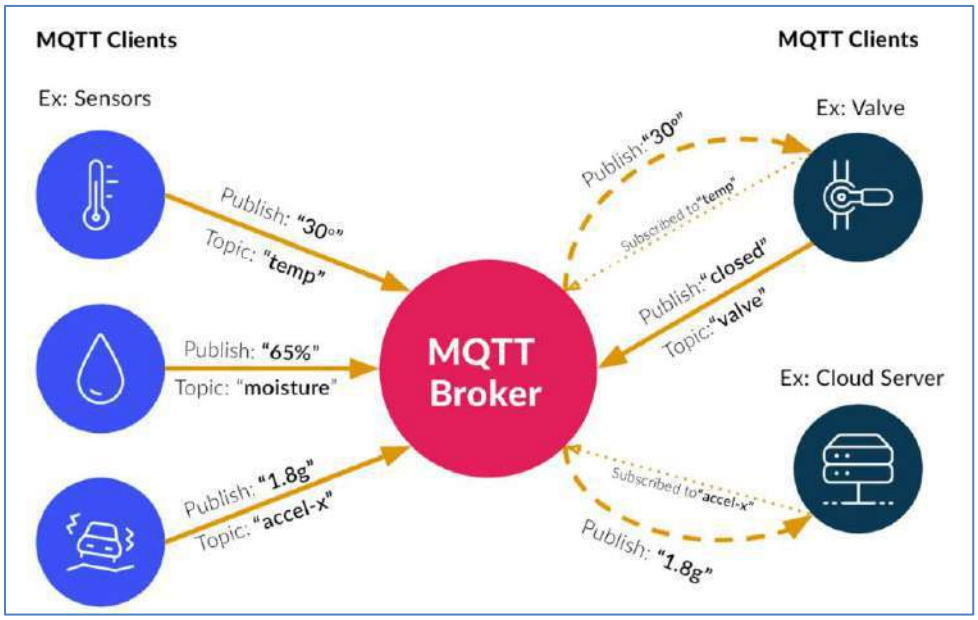


Figura 2.5.1 Esquema del modelo de comunicación publish/subscribe en el protocolo MQTT.
Fuente: Cavli Wireless. "What Is the MQTT Protocol?". Cavli Wireless Blog, disponible en: https://www.cavliwireless.com/blog/nerdiest-of-things/what-is-the-mqtt-protocol

Información técnica aplicada al ámbito *SCADA* muestra que *MQTT* puede emplearse para recibir datos provenientes de dispositivos externos siempre que exista un *broker* encargado de gestionar la distribución de los mensajes[24]. En este contexto, las plataformas de supervisión pueden suscribirse a temas relevantes y procesar los datos recibidos sin realizar consultas periódicas a cada dispositivo.

En este proyecto, *MQTT* se utilizó de manera experimental para verificar su interoperabilidad con *RapidSCADA*. Se configuró un *broker* local y se empleó un simulador para publicar mensajes en formato *JSON*. *RapidSCADA* recibió estos mensajes mediante

[23] *MQTT* – Página oficial https://mqtt.org/
[24] Benefits and Applications of MQTT in SCADA for Reliable Data Exchange https://www.cse-icon.com/mqtt-in-scada/

su controlador *MQTT* y los procesó mediante un *script JavaScript* que permitió asignar los valores a los canales internos correspondientes.

2.6 Fundamentación de la selección de *RapidSCADA* como plataforma de desarrollo

La selección de *RapidSCADA* como plataforma base para este trabajo de grado responde a una combinación de criterios técnicos, pedagógicos y operativos que la convierten en una alternativa especialmente adecuada para proyectos de bajo costo orientados a la enseñanza de la automatización industrial.

En primer lugar, su licenciamiento libre bajo *Apache 2.0* permite utilizarla, modificarla y redistribuirla sin restricciones económicas. La versión base incorpora funciones completas de adquisición de datos, supervisión, alarmas e historización, sin límites estrictos de variables. Aunque existen módulos complementarios de pago (como los orientados a reportes o integración *GIS*), no son necesarios para la implementación planteada en este trabajo [7].

En segundo lugar, *RapidSCADA* ofrece compatibilidad nativa con *MQTT* y *Modbus* en sus variantes *RTU, TCP y ASCII*, lo que facilita su integración con módulos de adquisición industriales y microcontroladores sin desarrollar controladores o librerías adicionales. Además, dispone de extensiones para protocolos como *SNMP* u *OPC UA/DA*, lo que amplía sus posibilidades de uso en configuraciones futuras [8].

Un tercer criterio relevante es que la plataforma proporciona funcionalidad *SCADA* completa desde su instalación base. Incluye herramientas de configuración gráfica para definir canales de datos, bases *SQL* para almacenamiento histórico, gestión de alarmas y administración de usuarios. Estas características reducen la dependencia de aplicaciones externas y simplifican el despliegue del sistema [9].

Otro aspecto clave es su curva de aprendizaje favorable para entornos académicos. Gran parte de la configuración se realiza mediante interfaces visuales y archivos estructurados, lo que permite a los estudiantes concentrarse en entender la arquitectura *SCADA*, la comunicación industrial y la integración con dispositivos reales, sin necesidad de desarrollar software desde cero. Su estructura modular y transparente la convierte en un recurso didáctico apropiado para cursos de redes industriales y supervisión [10].

RapidSCADA también destaca por su escalabilidad y ejecución multiplataforma. Puede funcionar en sistemas *Windows o Linux*, en computadores personales o en dispositivos embebidos como *Raspberry Pi*. Adicionalmente, mediante módulos como *RapidGate* es posible implementar arquitecturas distribuidas o replicadas, lo que habilita escenarios más avanzados sin abandonar la plataforma [11].

Finalmente, la herramienta cuenta con documentación oficial amplia, foros activos y ejemplos de configuración, elementos que resultaron fundamentales durante la integración de los módulos *ADAM 4018+/4024* y del *Arduino Nano* en las fases experimentales del proyecto. Este ecosistema de soporte ha sido particularmente útil para resolver dudas de implementación y validar comportamientos en tiempo real [12].

2.7 Antecedentes y estado del arte en entornos educativos

2.7.1 Antecedentes a nivel nacional

En Colombia, la incorporación de sistemas *SCADA* en el ámbito universitario ha tomado fuerza en las últimas décadas, como respuesta a la necesidad de acercar la formación en ingeniería a las herramientas utilizadas en la industria. El costo elevado de las licencias comerciales ha motivado la exploración de alternativas basadas en software libre y hardware de bajo costo.

Castrillón y Salazar, en la Universidad *EIA*, desarrollaron un sistema *SCADA* basado en tecnologías abiertas para la supervisión de plantas piloto de nivel y presión, demostrando que es posible implementar laboratorios robustos sin depender de soluciones propietarias [13]. En la Universidad Tecnológica de Bolívar se rediseñó un sistema de supervisión que originalmente utilizaba *WinCC* Flexible, limitado a una única estación de monitoreo. La solución propuesta migró hacia un *SCADA* web basado en *OPC*, permitiendo el control remoto de una planta de nivel a través de Internet y ampliando el alcance didáctico del laboratorio [14].

La Universidad de los Andes implementó un sistema *SCADA* de software libre para supervisar una microrred no convencional, registrando variables de generación fotovoltaica y condiciones de la red, y evidenciando la viabilidad de las soluciones abiertas en el seguimiento de sistemas energéticos [15]. De manera más reciente, el capítulo estudiantil de Instrumentación y Medidas del *IEEE* en la Universidad del Valle organizó el seminario-taller "*RapidSCADA*: Plataforma *Open Source* de Automatización Industrial", liderado por la autora de este trabajo de grado. En este espacio se introdujeron los conceptos básicos de la herramienta, se realizaron ejercicios prácticos de integración con dispositivos de campo y se evidenció el interés de los estudiantes por contar con una plataforma libre y replicable para sus proyectos académicos[25].

En conjunto, estos antecedentes muestran una transición progresiva desde el uso exclusivo de plataformas comerciales hacia la adopción de soluciones abiertas, que permiten construir laboratorios de automatización más accesibles y ajustados a las necesidades locales.

[25]Seminario: RapidSCADA – Plataforma *Open Source* de Automatización Industrial https://events.vtools.ieee.org/m/507123

2.7.2 Antecedentes a nivel continental

En América Latina y Norteamérica también se reportan experiencias significativas de uso de *SCADA* libres con fines educativos. En Ecuador, la Universidad Católica de Cuenca desarrolló un sistema *SCADA* con *Node-RED* para integrar diferentes *PLC* del laboratorio de automatización mediante una interfaz web unificada, lo que mejoró la supervisión de procesos y la interacción de los estudiantes con las prácticas [16]. En Perú se implementó un sistema *SCADA* educativo utilizando *RapidSCADA* como servidor de supervisión enlazado vía *OPC* a una microrred simulada en *Matlab-Simulink*, centrada en fuentes renovables y sincronización de red, demostrando su pertinencia para cursos de energías renovables [17].

En Brasil, la Universidad Federal Fluminense empleó *ScadaBR* para supervisar procesos químicos simulados, combinando modelos matemáticos con pantallas *HMI* personalizadas y fortaleciendo las competencias de los estudiantes en control de procesos [18]. Asimismo, en la Universidad Politécnica Salesiana (Ecuador) se integró *Node-RED* con una planta didáctica *FESTO MPS PA*, incorporando incluso comandos de voz mediante asistentes virtuales, lo que acerca a los estudiantes a tecnologías *IoT* en un entorno controlado [19].

En Estados Unidos, la Universidad de Alabama en *Huntsville* desarrolló un laboratorio de ciberseguridad industrial basado en plataformas *SCADA* abiertas, capaz de emular infraestructuras como plantas de agua o gasoductos y de recrear escenarios de ataque y defensa, lo que permite una formación práctica en la protección de sistemas de control [20].

Estas experiencias sugieren que las plataformas *SCADA* de código abierto son una opción válida para actividades formativas, al permitir implementar funciones fundamentales de supervisión dentro de los límites presupuestarios de muchas instituciones educativas.

2.7.3 Antecedentes a nivel global

En el ámbito internacional, la tendencia hacia el uso de plataformas *SCADA* abiertas también es evidente. En España, la Universidad de Almería implementó un *SCADA* basado en *software* libre para monitorear su laboratorio de energías renovables, integrando diferentes fuentes energéticas y ofreciendo acceso vía *web* a los estudiantes [21]. La Universidad Politécnica de Valencia diseñó un sistema de supervisión accesible desde navegador para gestionar una microrred híbrida, utilizando un *PLC OMRON* conectado a bases de datos *MySQL* y a una interfaz desarrollada en *Java/PHP*, demostrando la viabilidad de soluciones abiertas en aplicaciones de generación distribuida [22].

En la Universidad de Coímbra (Portugal) se empleó *RapidSCADA* como *HMI* en un laboratorio de ciberseguridad industrial enfocado en plantas solares, permitiendo estudiar vulnerabilidades y mecanismos de protección en un entorno controlado [23].

En África, la Universidad de Ashesi (Ghana) utilizó *ScadaBR* para implementar laboratorios SCADA básicos en una red local cerrada, con el objetivo de supervisar sensores y controlar

actuadores desde navegadores *web*. Esta experiencia confirmó que, incluso en contextos con recursos limitados, es posible reproducir infraestructuras de automatización funcionales mediante soluciones de código abierto [24].

2.8 Síntesis del marco conceptual

Este capítulo reunió los elementos teóricos necesarios para comprender el papel que desempeñan los sistemas *SCADA* dentro de la automatización industrial y su utilidad en contextos formativos. En primer lugar, se presentó la naturaleza de estos sistemas, su evolución histórica y los componentes que permiten integrar el mundo físico con las plataformas digitales de supervisión. También se explicó su estructura funcional por capas, la cual organiza las tareas de adquisición de datos, procesamiento y visualización.

El capítulo también abordó los distintos modelos de software *SCADA* (comerciales y de código abierto), destacando sus diferencias en licenciamiento, alcance técnico y pertinencia educativa. Este análisis permitió justificar la selección de *RapidSCADA* como herramienta de trabajo en este trabajo de grado, tanto por su licencia libre como por su arquitectura modular y su compatibilidad nativa con *MQTT* y las variantes *RTU y TCP* del protocolo *Modbus*.

Posteriormente, se describieron los protocolos empleados en el estudio, haciendo énfasis en *Modbus* por su presencia en módulos de adquisición de bajo costo y su utilidad en prácticas de laboratorio, e incorporando *MQTT* como alternativa ligera para el intercambio de mensajes en arquitecturas distribuidas. Finalmente, se revisaron experiencias reportadas en Colombia y en otros países, evidenciando una tendencia sostenida hacia el uso de plataformas *SCADA* abiertas en universidades, centros de investigación y entornos de formación técnica.

Los conceptos tratados a lo largo del capítulo constituyen la base que orienta el diseño e implementación del sistema desarrollado en este proyecto. A partir de estos fundamentos, el capítulo siguiente se centra en la arquitectura específica de *RapidSCADA* y en los aspectos técnicos que permiten llevar la teoría a una configuración práctica y operativa.

3. Arquitectura funcional y plataforma tecnológica de RapidSCADA

3.1 Introducción al capítulo

En este capítulo se presenta la arquitectura funcional y la plataforma tecnológica de *RapidSCADA*. El objetivo es comprender cómo está conformado este sistema y de qué manera sus componentes trabajan de forma conjunta para crear un entorno de supervisión y control confiable. Conocer su estructura resulta necesario para el desarrollo de la implementación propuesta en este trabajo de grado, ya que permite identificar la lógica interna sobre la cual se construyen los procesos de adquisición, procesamiento y visualización de datos.

RapidSCADA tiene un diseño modular que está compuesto por diversos servicios que trabajan de manera coordinada para realizar funciones específicas: adquisición de información desde equipos de campo, procesamiento interno de datos y visualización de resultados mediante una interfaz *web* interactiva [8]. Esta estructura flexible y multiplataforma permite su uso tanto en entornos industriales como en laboratorios académicos, donde se busca una alternativa libre y de bajo costo.

En las secciones siguientes se detalla la estructura general del sistema y se describen las aplicaciones que lo componen: ***Server, Communicator, Webstation, Administrator*** y ***Agent***. También se analiza el intercambio de información entre ellos y cómo se gestiona la base de datos que almacena los registros del proceso. Además, se aborda la capacidad de extensión mediante *scripts* en *C#,* los protocolos industriales compatibles y las medidas de seguridad, redundancia y respaldo que garantizan la confiabilidad del sistema.

El análisis técnico que se desarrolla en este capítulo sirve de base para las etapas posteriores del trabajo. Los conceptos expuestos apoyan la implementación del sistema básico descrita en el Capítulo 4 y para la integración con hardware físico mediante el protocolo *Modbus*, presentada en el Capítulo 5. De esta manera, se contribuye al propósito central de la investigación: justificar que *RapidSCADA* puede aplicarse como una solución educativa accesible para la enseñanza y práctica de la automatización industrial.

3.2 Estructura general del sistema

La estructura modular de RapidSCADA permite integrar distintos niveles de procesamiento y comunicación, lo que facilita su aplicación tanto en entornos de laboratorio como en instalaciones industriales con múltiples dispositivos conectados.

En su diseño, *RapidSCADA* adopta un modelo cliente-servidor multicapa, donde los componentes principales cumplen funciones específicas dentro del flujo de información. El sistema se organiza en torno a tres procesos fundamentales: adquisición de datos (Communicator), procesamiento y almacenamiento (Server), y presentación al usuario (Webstation). Cada una de estas etapas es ejecutada por un módulo independiente que interactúa con los demás mediante interfaces definidas y protocolos de comunicación estándar [8].

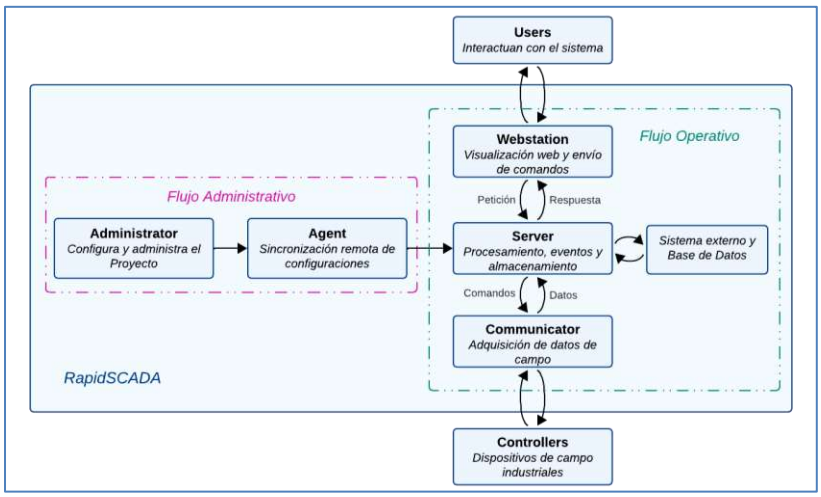


Figura 3.2.1 Flujo operativo y administrativo de una arquitectura básica de Rapid SCADA.
Fuente: Elaboración propia

El esquema de funcionamiento puede representarse de forma general como se muestra en la Figura 3.2.1, donde se integran dos tipos de flujo: un flujo administrativo, encargado de la configuración y sincronización remota del proyecto (***Administrator*** y ***Agent***), y un flujo operativo, responsable de la adquisición de datos, su procesamiento y la interacción con el usuario (***Communicator, Server*** y ***Webstation***). En el nivel más bajo se encuentran los dispositivos de campo, como sensores, actuadores o controladores lógicos programables (*PLCs*), que generan las variables del proceso.

La información recibida es transmitida al módulo ***Server***, que actúa como núcleo del sistema. En él se almacenan los datos históricos, se aplican cálculos o fórmulas definidas por el usuario y se generan los registros de eventos y alarmas. Además, el Server se puede comunicar con bases de datos externas, por ejemplo, *PostgreSQL o InfluxDB*, con el fin de conservar los valores históricos y facilitar su posterior análisis [10].

Los resultados procesados por el servidor se ponen a disposición del usuario a través del módulo ***Webstation***, el cual proporciona una interfaz gráfica accesible mediante navegador web. Desde allí, es posible visualizar tendencias, alarmas y reportes, así como enviar comandos hacia los dispositivos de campo. Finalmente, los módulos ***Administrator*** y ***Agent*** complementan el sistema al permitir la configuración del proyecto, la administración de usuarios y la sincronización remota de las configuraciones y servicios del sistema.

La arquitectura modular permite escalar el sistema de manera flexible según las necesidades de cada implementación. En su forma más sencilla, la plataforma puede instalarse en un único equipo que integra todos los componentes (***Server, Communicator, Webstation, Administrator*** y ***Agent***), lo que resulta apropiado para laboratorios o pruebas de desarrollo. No obstante, el mismo esquema puede expandirse hacia configuraciones distribuidas, donde varios servidores pueden intercambiar información entre sí o distribuir funciones según la estructura de cada implementación.

Además, el *software* contempla un modo de operación redundante, en el cual un servidor principal y otro de respaldo mantienen una sincronización automática de datos y servicios. Esta funcionalidad mejora la continuidad operativa del sistema ante posibles fallos o

interrupciones, alineándose con los criterios de confiabilidad habituales en entornos *SCADA* industriales.

En contextos académicos, esta arquitectura permite que los estudiantes observen de forma práctica la interacción entre niveles de campo, control y supervisión, al mismo tiempo que experimentan con protocolos industriales reales y herramientas de visualización propias de los sistemas *SCADA* modernos.

3.3 Módulos principales

La arquitectura de *RapidSCADA* se organiza en torno a cinco aplicaciones que trabajan de manera coordinada: ***Server, Communicator, Webstation, Administrator y Agent***. Cada uno de estos componentes desempeña un papel específico dentro del flujo de información, conformando un entorno integral para la adquisición, el procesamiento, el almacenamiento y la visualización de datos [10]. La disposición general de estas aplicaciones dentro de la plataforma se ilustra en la Figura 3.3.1.

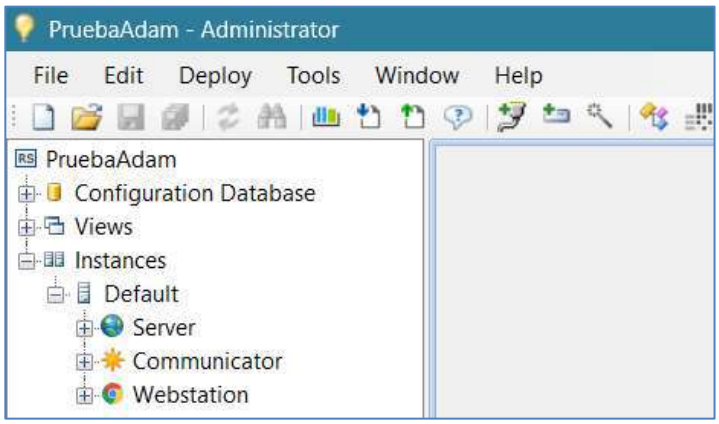


Figura 3.3.1 Aplicaciones del Rapid SCADA.
Fuente: Elaboración Propia

3.3.1 Server

El módulo ***Server*** es el núcleo operativo de *RapidSCADA*. Su función principal es recibir los datos provenientes de los dispositivos de campo a través del módulo ***Communicator***, procesarlos y generar los resultados que se le presentarán al usuario final. En esta etapa se aplican fórmulas matemáticas o *scripts* declaradas por el usuario, se registran los eventos y alarmas del sistema, y se almacenan los valores históricos en diferentes formatos, como archivos locales o bases de datos externas, según la configuración del sistema [8].

Además de gestionar la información proveniente del **Communicator**, el ***Server*** atiende las solicitudes enviadas por ***Webstation***, proporciona los datos procesados para su visualización y ejecuta los comandos que el usuario genera desde la interfaz *web*. También puede intercambiar información con sistemas externos o bases de datos adicionales cuando el proyecto lo requiere. De esta manera, el ***Server*** concentra la operación continua

del sistema y la disponibilidad de la información en tiempo cercano al tiempo real, aspectos relevantes en entornos industriales y de formación académica donde se requiere trazabilidad de los datos.

3.3.2 Communicator

El módulo ***Communicator*** establece el enlace entre *RapidSCADA* y los dispositivos de campo. Gestiona las líneas de comunicación y los controladores (*drivers*) necesarios para la adquisición de datos, soportando protocolos como Modbus *RTU/TCP, OPC, MQTT o SNMP* mediante los drivers correspondientes [8].

Su función consiste en consultar periódicamente a los dispositivos, recopilar los valores obtenidos y transmitirlos al ***Server*** para su procesamiento. El diseño del ***Communicator*** permite operar varias líneas de forma paralela, lo que facilita la integración de distintos equipos y protocolos en un mismo proyecto, característica especialmente útil en entornos académicos donde se requiere experimentar con arquitectura heterogénea.

3.3.3 Webstation

El módulo ***Webstation*** es la interfaz de usuario del sistema. A través de esta aplicación *web*, el operador puede visualizar datos procesados, revisar tendencias, consultar alarmas y generar comandos hacia los dispositivos conectados. La comunicación se realiza mediante un navegador estándar empleando *HTTP* o *HTTPS* según la configuración del servidor *web*, lo que permite acceder al sistema desde distintos equipos sin instalar *software* adicional [8].

Estas características lo convierten en una herramienta adecuada para entornos educativos, al ofrecer una visualización clara del comportamiento del sistema y permitir la interacción directa con el proceso supervisado.

3.3.4 Administrator

El módulo ***Administrator*** es la herramienta de desarrollo y configuración de proyectos en *RapidSCADA*. Desde su interfaz se definen los dispositivos, canales, líneas de comunicación, vistas, roles de usuario y parámetros de cálculo que conforman la base de configuración del sistema [8].

Aunque desde ***Administrator*** es posible supervisar el estado general de los servicios, como revisar registros o verificar la disponibilidad de módulos, la adquisición de datos y la consulta a los controladores no se realiza en este componente, sino en el ***Communicator***. Su función corresponde al flujo administrativo del sistema, orientado a crear, ajustar y mantener el proyecto *SCADA*. En el contexto académico, permite modificar parámetros y observar cómo estos cambios afectan el comportamiento del sistema, lo que facilita la comprensión estructural del *SCADA*.

3.3.5 Agent

El módulo ***Agent*** se utiliza para sincronizar y administrar de manera remota las configuraciones entre el ***Administrator*** y las instancias de *RapidSCADA* instaladas en distintos servidores. Este servicio permite transferir archivos del proyecto, obtener registros de *log* y verificar el estado de los servicios *SCADA*, facilitando tareas como despliegue de cambios o reinicio de módulos en entornos distribuidos [8].

Su uso resulta especialmente útil en laboratorios con varias estaciones o en implementaciones que buscan reproducir una arquitectura distribuida de control, ya que reduce la intervención manual y disminuye el riesgo de inconsistencias de configuración.

3.4 Flujo de interacción entre módulos

El flujo de operación de *RapidSCADA* se basa en la interacción coordinada entre sus módulos, donde cada componente cumple una función específica dentro del ciclo de adquisición, procesamiento y presentación de la información. La Figura 3.4.1 resume este recorrido, distinguiendo el flujo operativo (que involucra la comunicación con los dispositivos de campo y la visualización al usuario) del flujo administrativo, asociado a la gestión y transferencia de la configuración del proyecto.

En el flujo operativo, los dispositivos de campo generan las variables del proceso, las cuales son adquiridas por el ***Communicator*** mediante los protocolos configurados. Los datos obtenidos se transmiten al ***Server***, que los procesa aplicando las fórmulas definidas, registra los eventos y alarmas, y conserva los valores históricos según la configuración del proyecto. Esta información se pone a disposición de ***Webstation***, desde donde el operador puede consultar las variables, revisar alarmas y emitir comandos que el ***Server*** redirige a los dispositivos a través del ***Communicator***.

En paralelo, el flujo administrativo integra los módulos ***Administrator*** y ***Agent***. El primero permite crear y ajustar la configuración del proyecto, mientras que el segundo automatiza su transferencia hacia los servidores locales o remotos en caso de instalaciones distribuidas. Esta separación entre operación y administración contribuye a mantener la organización funcional del sistema y a facilitar su despliegue en distintos entornos académicos y de laboratorio.

La Figura 3.4.1 muestra de forma conjunta el recorrido de los datos desde el nivel de campo hasta la interfaz del usuario, así como el retorno de los comandos hacia los dispositivos, de acuerdo con la arquitectura básica implementada.

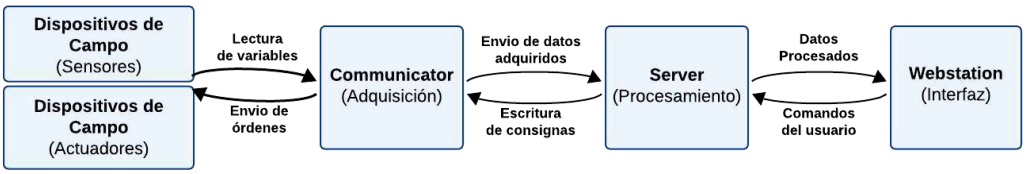


Figura 3.4.1 Esquema general del flujo de interacción entre los módulos de RapidSCADA.
Fuente: Elaboración Propia

3.5 Modelo de datos y base de configuración

La base de configuración de *RapidSCADA* define la estructura lógica del proyecto y las relaciones que permiten que los módulos funcionen de manera coordinada. Su objetivo no es almacenar valores del proceso, sino describir la forma en que el sistema adquiere, procesa y presenta la información. Las tablas primarias que conforman esta base de configuración, junto con su propósito técnico dentro del sistema, se resumen en la Tabla 3.5.1.

Tabla 3.5.1 Tablas primarias de la base de configuración en RapidSCADA.
Fuente: Elaboración Propia

Tabla	Descripción	Propósito técnico
Objects	Define las entidades físicas o lógicas supervisadas (plantas, áreas o subsistemas).	Permite estructurar jerárquicamente el proyecto *SCADA* y organizar los dispositivos por ubicación o función.
Communication lines	Contiene las líneas de comunicación que gestionará el módulo ***Communicator***.	Determina la conexión entre el sistema y los dispositivos de campo, estableciendo protocolo, puerto y frecuencia de sondeo.
Devices	Representa los equipos conectados a cada línea de comunicación.	Es el vínculo directo entre el *hardware* (*PLCs*, módulos, sensores, microcontroladores) y el sistema *SCADA*.
Channels	Registra las variables de proceso, tanto de entrada como de salida.	Constituye el núcleo de los datos del sistema; cada canal corresponde a una variable supervisada o controlada.
Limits	Define los valores mínimos y máximos permitidos para los canales.	Facilita la detección de condiciones anómalas y la generación de alarmas.
Views	Configura las vistas gráficas o paneles que se muestran en ***Webstation***.	Proporciona la representación visual de los datos de proceso y las interfaces de control.
Roles	Establece los roles de usuario disponibles.	Controla la jerarquía de permisos dentro del sistema.
Role inheritance	Define las relaciones jerárquicas entre roles.	Permite que un rol herede permisos de otro, simplificando la administración de usuarios.
Object rights	Asigna los derechos de acceso a objetos o vistas específicas.	Refuerza la seguridad del sistema al limitar la interacción según el perfil del usuario.
Users	Contiene los usuarios registrados en el sistema.	Controla la autenticación y acceso seguro a ***Webstation*** y ***Administrator***.

Desde la perspectiva de la ingeniería de automatización, la base de configuración actúa como un modelo de datos declarativo, en el cual cada tabla representa una entidad del sistema y sus dependencias funcionales. De acuerdo con la documentación oficial, esta

base se organiza en tablas primarias y tablas secundarias, diferenciadas según su impacto en la operación del sistema: las primeras determinan la arquitectura fundamental de comunicación y supervisión, mientras que las segundarias amplían las capacidades del sistema sin modificar su estructura principal [25].

Tabla 3.5.2 Tablas secundarias de la base de configuración en RapidSCADA.
Fuente: Elaboración propia

Tabla	Descripción	Importancia o relevancia
Archive kinds	Define los tipos de archivo histórico utilizados para el registro de datos.	Permite establecer estrategias de almacenamiento (por periodo o cambio de valor).
Archives	Contiene la configuración de los archivos históricos del sistema.	Determina la forma y frecuencia con que el ***Server*** guarda los valores de los canales.
Channel status	Enumera los estados posibles de un canal (por ejemplo, activo, error, desconectado).	Mejora la trazabilidad de la comunicación y el diagnóstico del sistema.
Channel types	Clasifica los canales según su función (analógico, digital, calculado).	Permite diferenciar el tipo de variable y su tratamiento dentro del ***Server***.
Data types	Especifica el formato de los datos utilizados por los canales.	Garantiza compatibilidad y coherencia en el manejo de los valores adquiridos.
Device types	Define los tipos de dispositivos que pueden configurarse.	Facilita la estandarización de controladores y la reutilización de configuraciones.
Formats	Determina la forma en que se presentan los valores numéricos.	Mejora la legibilidad de los datos en las vistas gráficas.
Quantities	Define las magnitudes físicas medidas o controladas (temperatura, presión, flujo).	Asegura la correcta interpretación de las variables de proceso.
Scripts	Contiene las fórmulas o expresiones en *C#* aplicadas a canales.	Permite personalizar el comportamiento del sistema mediante cálculos o condiciones lógicas.
Units	Define las unidades de medida asociadas a los canales.	Estandariza la presentación de datos en las vistas y reportes.
View types	Clasifica los tipos de vistas disponibles.	Permite estructurar distintos paneles de supervisión según el tipo de proceso o usuario.

De manera complementaria, la Tabla 3.5.2 presenta las tablas secundarias de la base de configuración, las cuales apoyan funciones específicas de operación, registro y visualización del sistema. Por ejemplo, un *script* puede utilizar valores definidos en la tabla *Channels*, mientras que los usuarios acceden a vistas configuradas en *Views* según los permisos establecidos. Esta estructura relacional proporciona flexibilidad y escalabilidad sin alterar el núcleo operativo del sistema.

El módulo ***Administrator*** es responsable de la creación, validación y despliegue (*Deploy*) de la base de configuración, verificando que las relaciones entre las tablas sean coherentes antes de su envío al sistema. Durante este proceso se generan archivos de configuración, en formatos como *XML* o *DAT*, que se transfieren automáticamente a los ***módulos Server, Communicator y Webstation***, los cuales los interpretan como instrucciones de funcionamiento. Este procedimiento contribuye a mantener la consistencia de la

información, facilita la ampliación del proyecto con nuevas líneas o dispositivos y permite replicar configuraciones sin modificar el código fuente [25].

En conjunto, la base de configuración actúa como el vínculo entre el diseño conceptual del proyecto y su implementación práctica, favoreciendo una integración coherente en entornos industriales y académicos.

3.6 Extensibilidad y scripting interno

El *scripting* interno de *RapidSCADA* permite ampliar el procesamiento que realiza el módulo ***Server*** mediante expresiones escritas en *C#*. Estas instrucciones complementan la base de configuración y facilitan la aplicación de cálculos o verificaciones adicionales sobre los valores adquiridos, sin necesidad de modificar el código fuente del sistema [26]. Dado que su ejecución se integra al ciclo normal de actualización del ***Server***, el efecto de estas reglas puede observarse de forma inmediata en el comportamiento del sistema.

La configuración del *scripting* se articula a través de dos elementos. Por un lado, la tabla *Channels*, donde cada canal puede habilitar una fórmula que modifica la forma en que su valor es interpretado o generado. Por otro, la tabla *Scripts*, que almacena funciones escritas en *C#* y disponibles para ser llamadas desde cualquier canal del proyecto. En ambos casos, el ***Server*** interpreta las expresiones sin requerir un proceso de compilación adicional, lo que permite realizar ajustes de forma sencilla durante el desarrollo. Las Figuras 3.6.1 y 3.6.2 muestran estas configuraciones tal como aparecen en el módulo ***Administrator***.

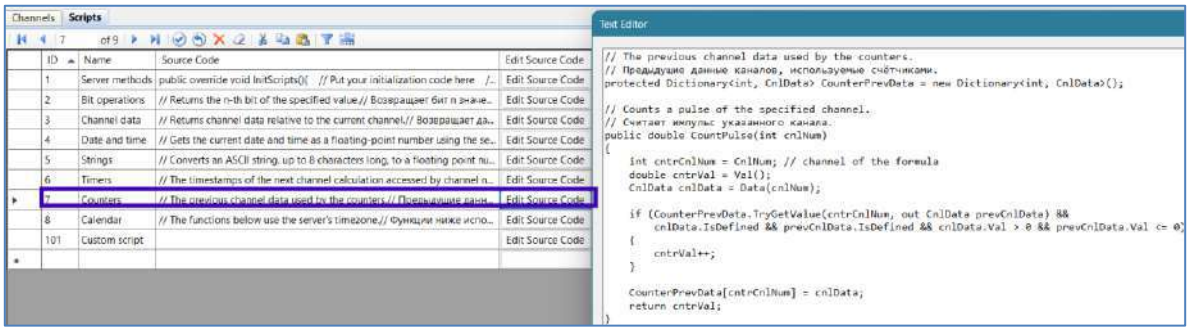


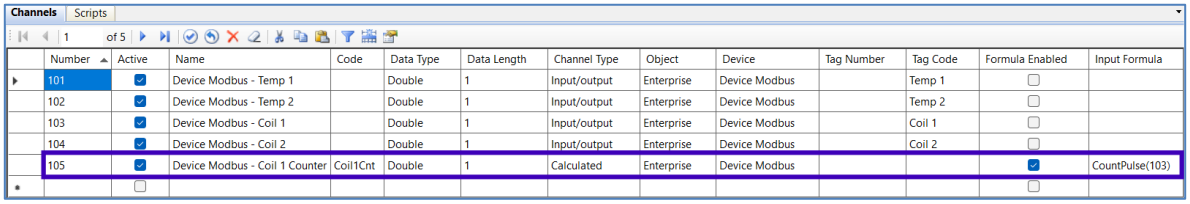
Figura 3.6.1 Fragmento de la tabla *Scripts* y del editor de código.
Fuente: Elaboración propia

Entre las funciones incluidas de manera predeterminada en la plantilla del sistema se encuentra *CountPulse*, diseñada para identificar transiciones de 0 a 1 en un canal digital y generar un valor acumulado. Su uso se limita a llamar la función desde un canal calculado, como se ve en el ejemplo 3.6.1:

Ejemplo 3.6.1 Llamada de función CountPulse en un canal calculado en *RapidSCADA*

$$CountPulse(103)$$

En términos operativos, esta llamada se configura directamente en el campo *Input Formula* del canal calculado dentro de la tabla *Channels*, como se observa en la Figura 3.6.2. Durante la ejecución normal del *Server*, la función es evaluada en cada ciclo de actualización, permitiendo detectar los flancos ascendentes del canal digital asociado y actualizar automáticamente el valor acumulado.



Number	Active	Name	Code	Data Type	Data Length	Channel Type	Object	Device	Tag Number	Tag Code	Formula Enabled	Input Formula
101	☑	Device Modbus - Temp 1		Double	1	Input/output	Enterprise	Device Modbus		Temp 1	☐	
102	☑	Device Modbus - Temp 2		Double	1	Input/output	Enterprise	Device Modbus		Temp 2	☐	
103	☑	Device Modbus - Coil 1		Double	1	Input/output	Enterprise	Device Modbus		Coil 1	☐	
104	☑	Device Modbus - Coil 2		Double	1	Input/output	Enterprise	Device Modbus		Coil 2	☐	
105	☑	Device Modbus - Coil 1 Counter	Coil1Cnt	Double	1	Calculated	Enterprise	Device Modbus			☑	CountPulse(103)

Figura 3.6.2 Fragmento de la tabla Channels con un canal calculado.
Fuente: Elaboración propia

Esta operación permite derivar nuevas variables sin modificar el dispositivo de campo ni el protocolo de comunicación. El resultado de esta prueba se presenta en la Figura 3.6.3, donde se muestra la visualización en ***Webstation*** de tres comandos de activación y desactivación enviados sobre el canal 103. Como consecuencia, el contador asociado incrementó su valor a 2, reflejando únicamente los flancos ascendentes detectados por el sistema.

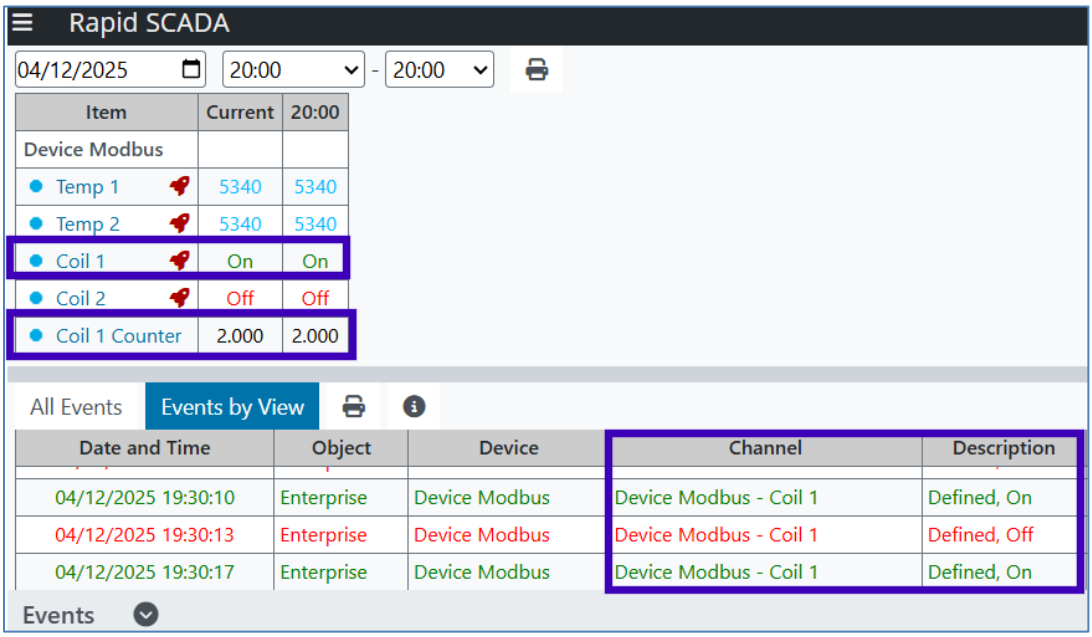


Figura 3.6.3 Resultado del contador asociado a la función *CountPulse* visualizado en *WebStation*.
Fuente: Elaboración propia

El conjunto, el mecanismo de *scripting* constituye una herramienta flexible para adaptar el procesamiento interno del sistema a las necesidades de cada proyecto. Su integración

directa con el módulo ***Server*** y la disponibilidad de funciones reutilizables permiten incorporar lógica adicional sin complejidad técnica, lo que resulta particularmente útil en contextos académicos y de experimentación.

3.7 Protocolos soportados

La comunicación entre *RapidSCADA* y los dispositivos de campo se basa en un sistema modular de controladores (*drivers*) que implementan distintos protocolos industriales. Esta funcionalidad se apoya en el diseño del módulo ***Communicator***, que abstrae las capas físicas de comunicación (serie, *Ethernet* o inalámbrica) mediante líneas de comunicación ejecutadas en paralelo [8]. Cada línea opera con un protocolo específico, y el ***Communicator*** actúa como cliente multiprotocolo, delegando en cada controlador las reglas de intercambio y la estructura de mensajes correspondiente.

Los datos adquiridos se convierten en variables internas (*Channels*) que el módulo ***Server*** procesa y almacena, mientras que ***Webstation*** los presenta al usuario. Esta arquitectura modular permite combinar protocolos como *Modbus, OPC o MQTT* sin modificar el núcleo del *software*, lo que favorece la interoperabilidad y la integración de dispositivos de distintos fabricantes [27].

- **Protocolo *Modbus* (*RTU, ASCII, TCP*)**

RapidSCADA ofrece soporte nativo para *Modbus* en sus variantes *RTU, ASCII* y *TCP*, lo que facilita su uso con equipos de bajo costo y dispositivos ampliamente disponibles en la industria. El módulo ***Communicator*** interpreta los registros *Modbus (coils, inputs, holding registers* e *input registers*) y los mapea a canales internos del sistema.

En la práctica, esto permite conectar sensores, *PLC* o microcontroladores (como *Arduino Nano* y los módulos *ADAM 4018+* y *ADAM 4024*) mediante una línea de comunicación configurada en modo *RTU* o *TCP*. El usuario define la línea, el dispositivo, la dirección *Modbus* y los parámetros de lectura. Por su estructura simple y su amplia adopción, *Modbus* resulta particularmente adecuado en entornos educativos y de laboratorio [28].

- **Protocolo *OPC***

El soporte de *RapidSCADA* para *OPC UA/DA* responde a requerimientos de integración con sistemas industriales existentes. El módulo ***Communicator*** puede actuar como cliente *OPC*, leyendo y escribiendo *tags* en servidores externos conforme a las especificaciones *COM/DCOM (DA)* y *TCP/TLS (UA).* Gracias a la arquitectura modular, esta funcionalidad se incorpora mediante controladores específicos sin modificar el núcleo de la aplicación.

De este modo, *RapidSCADA* puede intercambiar datos con *PLC*, historiadores y plataformas *SCADA* comerciales en un mismo entorno de supervisión, lo que facilita la consolidación y el seguimiento de la información de planta [29].

- **Protocolo *MQTT***

RapidSCADA incluye soporte para *MQTT* (*Message Queuing Telemetry Transport*), protocolo ampliamente utilizado en aplicaciones de *IIoT* (*Industrial Internet of Things*). En este caso, ***Communicator*** puede operar como cliente *MQTT*, conectándose a un *broker* (por ejemplo, *Mosquitto* o *HiveMQ*) para suscribirse a tópicos y publicar mensajes. Los mensajes recibidos se asocian a canales internos, de modo que las variables procedentes de sensores inalámbricos o de microcontroladores se procesan de la misma forma que los datos industriales tradicionales. Esta capacidad permite combinar comunicaciones clásicas basadas en *Modbus* u *OPC* con tecnologías orientadas a *IoT*, lo que amplía las posibilidades de uso de *RapidSCADA* en laboratorios de instrumentación y en proyectos de integración industrial [30].

3.8 Seguridad, redundancia y almacenamiento

La seguridad en *RapidSCADA* se gestiona a partir de la configuración de usuarios y roles, lo que permite controlar el acceso a la supervisión y a la ejecución de comandos. Cada usuario puede tener diferentes niveles de privilegio definidos en el módulo ***Administrator***, limitando las acciones disponibles en ***Webstation***. Según la documentación oficial, esta estructura jerárquica de permisos constituye la primera capa de protección frente a accesos no autorizados o errores de operación [31]. Asimismo, se recomienda restringir el acceso físico a los equipos que alojan el sistema, mantener los servicios actualizados y habilitar la comunicación cifrada mediante *HTTPS*, prácticas que contribuyen a preservar la integridad y confidencialidad de la información supervisada [32].

Además de los mecanismos de autenticación, *RapidSCADA* incorpora configuraciones orientadas a manejar fallos de comunicación. El módulo ***Communicator*** permite definir tiempos de espera y reintentos en la consulta a los dispositivos de campo, mientras que los errores detectados se registran en los archivos de *log* del sistema. Cuando una línea de comunicación se interrumpe, el estado de los canales correspondientes cambia a condiciones como “error” o “desconectado”, lo que facilita el diagnóstico y la recuperación del enlace una vez restablecido la comunicación [30, 27].

La redundancia en *RapidSCADA* se implementa mediante el módulo *Rapid Gate*, que posibilita la operación sincronizada entre un servidor principal y uno de respaldo. Ambos utilizan la misma base de configuración y mantienen la replicación de datos en tiempo cercano al tiempo real. En de que el servidor primario deje de responder, el secundario puede asumir la supervisión del proceso. Este esquema está alineado con prácticas de tolerancia a fallos utilizadas en sistemas industriales, en los que la continuidad operativa y la consistencia de los datos resultan aspectos relevantes [11].

El almacenamiento en *RapidSCADA* se organiza a partir del modelo de canales (*Channels*), que representan las variables medidas o calculadas por el sistema [33]. Cada canal conserva valores actuales, históricos y eventos asociados, los cuales son gestionados por

el módulo ***Server***. Dependiendo de la configuración, estos datos pueden almacenarse en archivos locales o enviarse a bases de datos externas mediante el controlador *DrvDBImport* [34]. Esta separación entre adquisición, procesamiento y almacenamiento favorece la coherencia temporal de los registros y la trazabilidad del proceso, manteniendo independencia entre las capas de *software* y los protocolos de comunicación utilizados.

En conjunto, las capacidades de seguridad, redundancia y almacenamiento que ofrece *RapidSCADA* aportan estabilidad operativa y resguardo de la información, características relevantes tanto para entornos industriales como para aplicaciones académicas donde la continuidad del sistema y la integridad de los datos resultan aspectos esenciales.

3.9 Conclusiones del capítulo

El análisis desarrollado en este capítulo permitió comprender la arquitectura modular de *RapidSCADA* como eje de su funcionamiento, donde los módulos ***Communicator, Server*** y ***Webstation*** interactúan de forma coordinada para adquirir, procesar y visualizar datos. Este diseño se apoya en principios de separación por capas y modularidad del *software*, lo que favorece la independencia funcional y facilita la integración con distintos dispositivos y protocolos.

La revisión de los protocolos soportados evidenció que la interoperabilidad del sistema se sustenta en su estructura abierta de controladores (*drivers*), capaces de interpretar estándares industriales como *Modbus, OPC UA/DA, MQTT y SNMP.* Este enfoque amplia las posibilidades de integración y muestra su adaptabilidad.

Asimismo, se identificó que *RapidSCADA* incorpora mecanismos de *scripting* interno en *C#*, los cuales amplían sus funciones mediante el procesamiento lógico de variables sin modificar el código base. Del mismo modo, los aspectos de seguridad, redundancia y almacenamiento aportan capacidades orientadas a mantener la continuidad operativa y la consistencias de los datos gestionados, mientras que la base de configuración establece las relaciones entre dispositivos, canales y vistas.

En conjunto, estos elementos permiten considerar a *RapidSCADA* como una plataforma adecuada para la enseñanza y la experimentación en automatización y control industrial, proporcionando los fundamentos técnicos necesarios para el diseño e implementación que se desarrollan en los capítulos siguientes.

4. Diseño e Implementación del Sistema *SCADA* Básico

4.1 Introducción y alcance

Este capítulo describe la implementación de un sistema *SCADA* básico utilizando la plataforma *RapidSCADA* en un entorno de simulación. La comunicación del sistema se estableció mediante los protocolos *Modbus TCP* y *MQTT*, empleando simuladores de proceso para la generación controlada de variables analógicas y digitales, lo que permite evaluar el funcionamiento del sistema sin interacción con *hardware* físico.

El desarrollo de esta etapa comprende la creación del proyecto base en *RapidSCADA*, la configuración del dispositivo simulado, la definición de la estructura de comunicación mediante plantillas y canales, y el diseño de vistas de supervisión en ***Webstation***. Asimismo, se incluyen la configuración de alarmas básicas, el registro histórico de variables y la validación funcional del sistema a partir de los datos recibidos desde los simuladores.

Todas las actividades descritas en este capítulo se ejecutaron en un entorno completamente virtual. La integración con dispositivos físicos y la comunicación mediante *Modbus RTU* sobre bus *RS-485* se abordan posteriormente, en el capítulo siguiente, como una extensión del sistema desarrollado.

4.2 Entorno de implementación e instalación de *RapidSCADA*

La implementación del sistema *SCADA* se realizó sobre un equipo con sistema operativo *Windows 11 Home Single Language (versión 25H2)*. Como plataforma *SCADA* se utilizó *RapidSCADA versión 6.4.3,* descargada desde el sitio oficial del proyecto e instalada mediante el instalador para *Windows* disponible en la distribución estable. El proceso de instalación y la selección de los módulos principales se ilustran en la Figura 4.2.1.

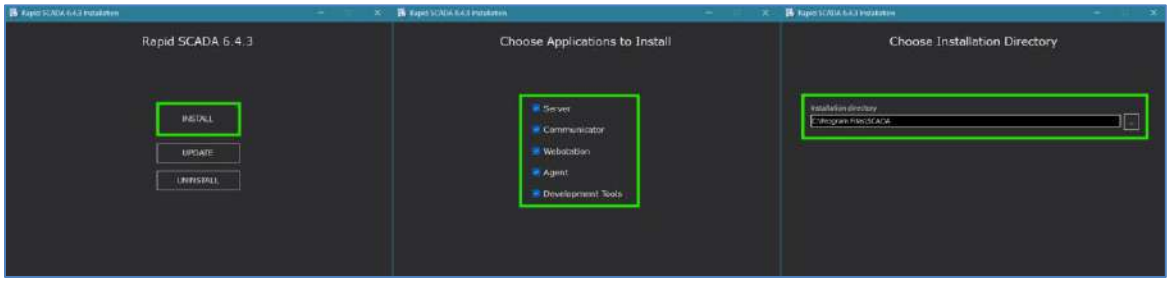


Figura 4.2.1 Instalador de *RapidSCADA* versión 6.4.3 para *Windows*.
Fuente: Elaboración propia.

La instalación se efectuó utilizando las opciones por defecto del instalador, manteniendo la ruta estándar del sistema en *C:\Program Files\SCADA*. Durante este proceso se verificó la presencia de los prerrequisitos necesarios para el funcionamiento de la plataforma, incluyendo el entorno *.NET 8.0* y los componentes requeridos para la ejecución de aplicaciones *web* basadas en *ASP.NET Core*. Adicionalmente, se habilitaron los servicios correspondientes en *Internet Information Services (IIS),* necesarios para la operación del

módulo Webstation y se confirmó la creación del *ScadaAppPool* para el acceso *web* al sistema, tal como se muestra en la Figura 4.2.2.

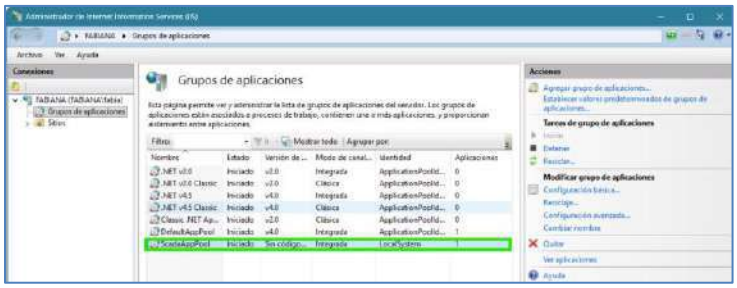


Figura 4.2.2 *ScadaAppPool* asociado a *RapidSCADA* en *Internet Information Services (IIS).*
Fuente: Elaboración propia.

Una vez finalizada la instalación, se comprobó la correcta ejecución de los servicios principales de *RapidSCADA* a través del administrador de servicios de *Windows, tal como se muestra en la Figura 4.2.3*. En particular, se verificó el estado activo de los servicios asociados a los ***módulos Server, Communicator y Agent***, asegurando su disponibilidad para la ejecución del proyecto.

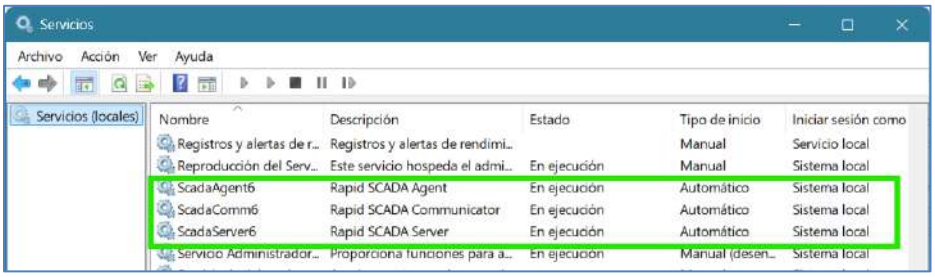


Figura 4.2.3 Servicios principales de *RapidSCADA* en ejecución en Windows.
Fuente: Elaboración propia.

El primer acceso a la plataforma se realizó a través del ***módulo Administrator***, desde el cual se identificó la instancia *Default* incluida en la instalación. Esta instancia sirvió como base para la creación y configuración del proyecto desarrollado en este capítulo. La verificación inicial del entorno permitió confirmar que los módulos de *RapidSCADA* se encontraban operativos y preparados para la configuración de las líneas de comunicación, dispositivos y demás elementos del sistema.

4.3 Diseño del sistema simulado

El diseño funcional describe el recorrido lógico de los datos entre los simuladores de proceso y los módulos principales de *RapidSCADA*, estableciendo cómo se adquiere, procesa y visualiza la información en un entorno *SCADA* básico, sin entrar aún en los detalles de configuración.

En esta etapa se consideran dos fuentes de datos simuladas: un esclavo *Modbus TCP* y un publicador *MQTT*. Ambos generan valores de proceso que son adquiridos por el ***módulo***

Communicator mediante sus respectivos controladores de protocolo. Los datos recibidos se transfieren al ***módulo Server***, donde se asocian a los canales definidos en la base de configuración y quedan disponibles para supervisión. Finalmente, ***módulo Webstation*** presenta esta información al usuario a través de vistas gráficas.

De forma resumida, el flujo funcional del sistema se compone de las siguientes acciones:

1. Generación de datos en los simuladores.
2. Adquisición de valores por el ***módulo Communicator***.
3. Procesamiento y almacenamiento lógico en el ***módulo Server***.
4. Visualización y envío de comandos desde **módulo *Webstation***.

La Figura 4.3.1 representa este flujo funcional con *Modbus TCP*, mostrando la relación entre los simuladores, los módulos de *RapidSCADA* y la interfaz de supervisión, sin considerar aún los procedimientos de instalación ni los parámetros específicos de configuración.

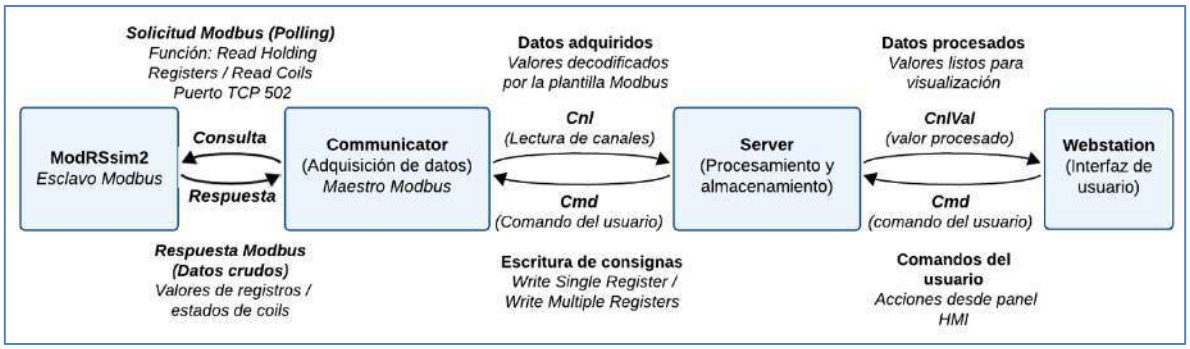


Figura 4.3.1 Flujo funcional del sistema *SCADA* simulado con *Modbus TCP*.
Fuente: Elaboración propia

Este diseño funcional sirve como referencia para las etapas de implementación que se desarrollan en las secciones posteriores del capítulo.

4.4 Configuración del entorno y creación del proyecto base en *RapidSCADA*

Para el desarrollo del sistema *SCADA* simulado fue necesario preparar el entorno de trabajo en *RapidSCADA* y definir la estructura mínima de comunicación que permitiera la interacción con los simuladores de proceso. En esta sección se describe exclusivamente la creación del proyecto base, la definición de las líneas de comunicación y el registro de los dispositivos asociados, sin abordar aún la configuración de canales, plantillas, vistas o validaciones funcionales, las cuales se desarrollan en secciones posteriores. El proceso de creación del proyecto base a partir de la plantilla por defecto se ilustra en la Figura 4.4.

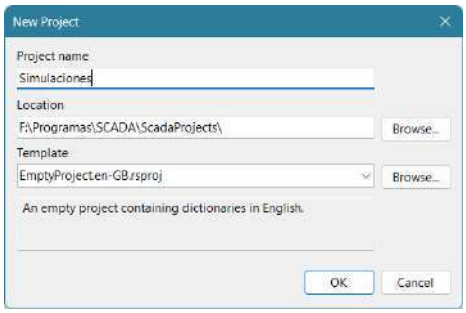


Figura 4.4.1 Creación del proyecto base en RapidSCADA a partir de la plantilla por defecto.
Fuente: Elaboración propia.

El proyecto se construyó a partir de la instancia *Default* incluida en *RapidSCADA*, la cual proporciona la estructura básica de los ***módulos Server, Communicator y Webstation***. Sobre esta base se definieron dos líneas de comunicación independientes, correspondientes a los protocolos Modbus TCP y MQTT, junto con sus respectivos dispositivos, de acuerdo con el esquema de simulación planteado para este capítulo.

4.4.1 Creación de la línea de comunicación

En el **módulo Communicator** se configuraron dos líneas de comunicación, cada una asociada a un protocolo distinto, con el objetivo de permitir la adquisición de datos desde entornos simulados heterogéneos. Esta configuración establece los canales lógicos a través de los cuales *RapidSCADA* gestiona el intercambio de información con los dispositivos o simuladores externos.

La creación de una línea de comunicación se inicia desde el ***módulo Administrator***, accediendo a la tabla Communication *lines* y seleccionando la opción para agregar una nueva línea. Al hacerlo, el sistema despliega la ventana de configuración inicial, en la cual se definen parámetros básicos como el nombre de la línea, su descripción y la instancia del ***módulo Communicator*** a la que quedará asociada, tal como se muestra en la Figura 4.4.2.

En este proyecto se creó una primera línea denominada *Modbus TCP* y una segunda línea denominada *MQTT*. Estas líneas constituyen la base de organización para la comunicación con los entornos de simulación definidos en el capítulo. La selección del protocolo específico, así como la configuración de direccionamiento y parámetros de comunicación se realiza posteriormente durante el registro de los dispositivos asociados a cada línea, de acuerdo con el protocolo implementado.

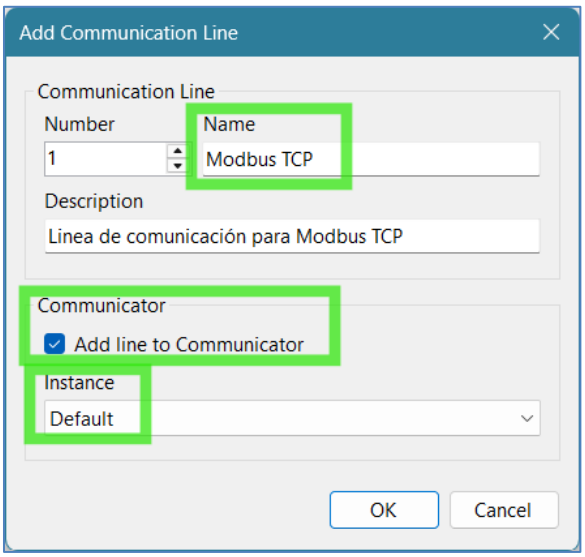


Figura 4.4.2 Ventana de creación de una línea de comunicación en el módulo Administrator de RapidSCADA.
Fuente: Elaboración propia

Una vez definidas ambas líneas, estas quedan registradas en la tabla *Communication lines*, donde es posible visualizar su identificación, nombre y descripción dentro del proyecto. El resultado final de este proceso, con las dos líneas de comunicación correctamente configuradas, se presenta en la Figura 4.4.3.

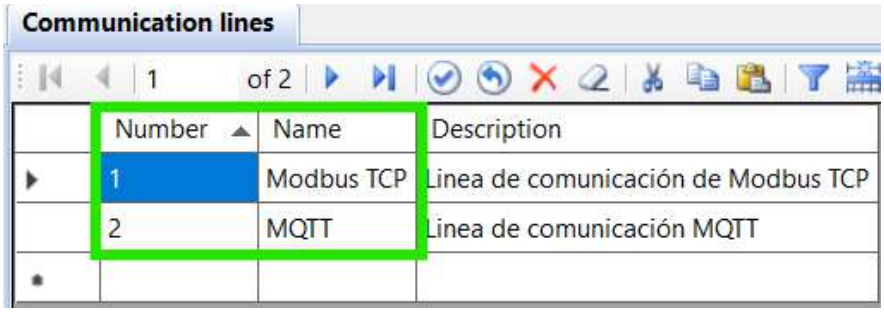


Figura 4.4.3 Líneas de comunicación configuradas en el módulo Communicator de RapidSCADA.
Fuente: Elaboración propia.

4.4.2 Registro de dispositivo asociados

Una vez definidas las líneas de comunicación, se realizó el registro de los dispositivos correspondientes. En *RapidSCADA*, un dispositivo representa la entidad lógica que vincula una línea de comunicación con las variables de proceso que serán posteriormente adquiridas y procesadas por el sistema *SCADA*.

Sobre la línea *Modbus TCP* se registró el dispositivo denominado *Device Modbus*, configurado para comunicarse con el simulador *ModRSsim2*. Este dispositivo se definió con

una dirección lógica igual a *1* y una dirección *IP 127.0.0.1,* utilizando el puerto estándar *502* del protocolo *Modbus TCP*. El controlador seleccionado fue *DrvModbus*, y el dispositivo quedó asociado exclusivamente a la línea de comunicación *Modbus TCP, tal como se muestra en la Figura 4.4.4.*

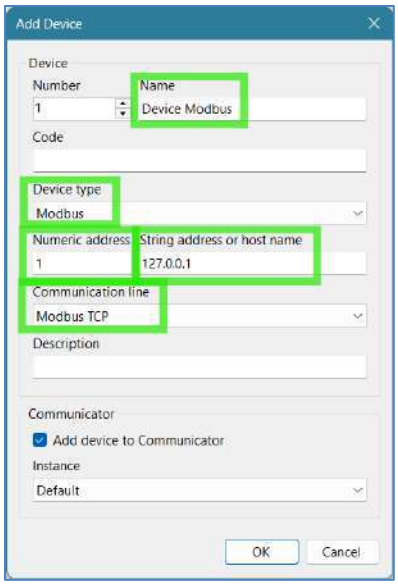


Figura 4.4.4 Ventana de registro de un dispositivo asociado a una línea de comunicación en RapidSCADA.
Fuente: Elaboración propia.

De manera análoga, sobre la línea *MQTT* se registró el dispositivo denominado *Device MQTT*, asociado a la comunicación con el *broker broker.mqtt-dashboard.com* a través del puerto *1883*. Este dispositivo representa la fuente de datos *MQTT* dentro del sistema y queda vinculado a la línea de comunicación *MQTT* previamente definida.

La Figura 4.4.5 presenta la tabla *Devices* del ***módulo Administrator***, donde se evidencian ambos dispositivos, su asociación correcta con cada línea de comunicación y los parámetros básicos utilizados para su registro.

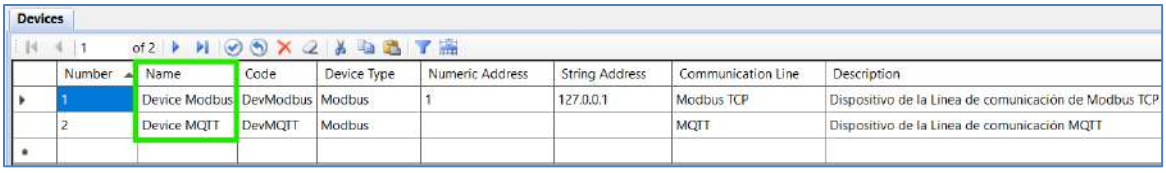


Figura 4.4.5 Dispositivos registrados y asociados a sus respectivas líneas de comunicación en RapidSCADA.
Fuente: Elaboración propia.

Con esta configuración, el proyecto base quedó preparado para establecer comunicación con los simuladores *Modbus TCP* y *MQTT*. A partir de esta estructura mínima, en las

secciones siguientes se aborda la definición de plantillas, la creación de canales y la configuración de los mecanismos de supervisión y registro de datos.

4.5 Configuración del simulador *ModRSsim2* como dispositivo *Modbus TCP*

Una vez definida la estructura básica del proyecto y registrados los dispositivos en *RapidSCADA*, se configuró el simulador *ModRSsim2* para suministrar las variables de prueba utilizadas durante la simulación del sistema *SCADA*. En esta etapa, el simulador cumple exclusivamente la función de fuente de datos *Modbus*, sin incorporar lógica de control ni procesamiento adicional.

En *ModRSsim2* se habilitó el servidor *Modbus TCP* y se definieron los registros que serían expuestos al sistema *SCADA*. La configuración incluyó dos *holding registers (registros de 16 bits),* destinados a representar variables analógicas simuladas, y dos *coils (variables binarias),* destinados a representar estados digitales de activación y desactivación. La configuración del servidor Modbus TCP en el simulador se ilustra en la Figura 4.5.1.

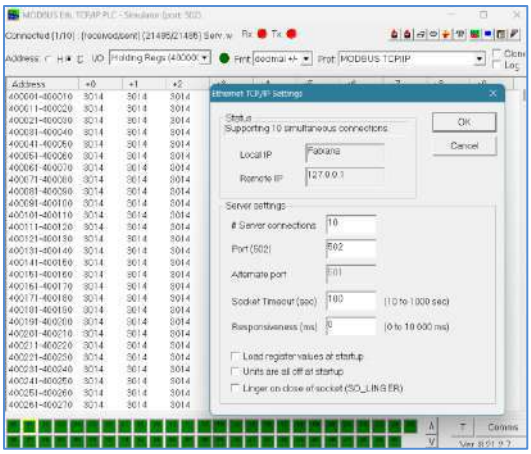


Figura 4.5.1 Configuración del servidor *Modbus TCP* en *ModRSsim2*.
Fuente: Elaboración propia

Los valores de los registros configurados pueden modificarse manualmente desde la interfaz del simulador, lo que permite generar cambios controlados en las variables analógicas y digitales. De igual forma, los *coils* pueden ser activados o desactivados para verificar la recepción de comandos enviados desde *RapidSCADA*. No se implementaron automatismos internos ni secuencias programadas dentro del simulador, manteniendo su comportamiento limitado a la respuesta directa a las solicitudes *Modbus*.

Con esta configuración, el simulador quedó preparado para interactuar con *RapidSCADA* mediante operaciones de lectura y escritura sobre los registros definidos. La correcta interpretación de estos datos por parte del sistema *SCADA* y su reflejo en los canales internos se analiza en las secciones siguientes, donde se aborda la creación de plantillas, canales y vistas de supervisión.

4.6 Configuración del simulador *MQTT* como fuente de datos

Una vez registrada la línea de comunicación *MQTT* y el dispositivo correspondiente en *RapidSCADA*, se configuró el simulador *MQTT Explorer* como fuente externa de datos para las pruebas del sistema *SCADA*. En esta etapa, el simulador se utilizó exclusivamente para la publicación manual de mensajes *MQTT*, sin incorporar lógica de control ni procesamiento adicional.

En *MQTT Explorer* se estableció una conexión con el *broker* público *broker.mqtt-dashboard.com*, utilizando el puerto estándar *1883* y sin autenticación. La conexión se realizó sin cifrado, manteniendo la configuración por defecto del cliente. Tras establecer la sesión, el simulador quedó habilitado para la publicación de mensajes en los *topics* definidos.

Para la simulación se emplearon dos *topics*, denominados *myparam1* y *myparam2*, sobre los cuales se publicaron mensajes en formato *JSON para uno de los topics*. Los mensajes contenían valores numéricos simulados, introducidos manualmente desde la interfaz del simulador, con el fin de generar cambios controlados en los datos transmitidos hacia *RapidSCADA*. La publicación de estos mensajes y la estructura de los *topics* configurados en *MQTT Explorer* se ilustran en la Figura 4.6.1.

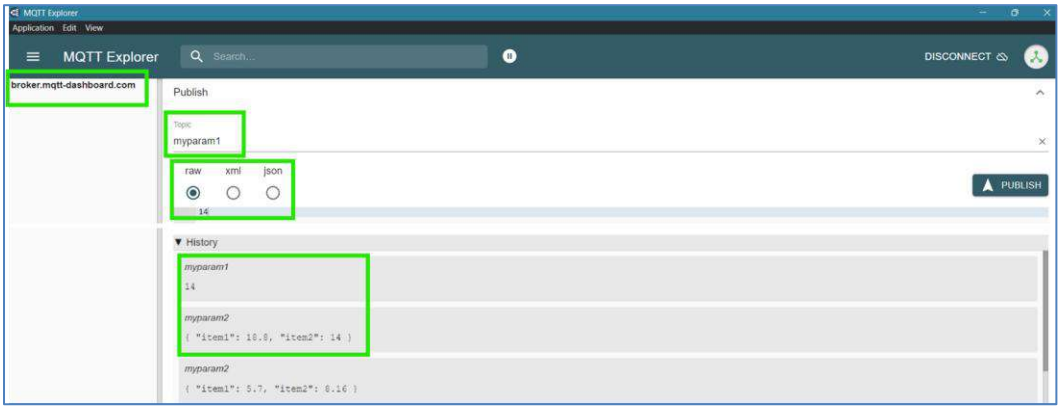


Figura 4.6.1 Publicación de mensajes *MQTT* en el simulador *MQTT Explorer* utilizando los *topics* *myparam1* y *myparam2* en formato *JSON*.
Fuente: Elaboración propia.

La recepción de los mensajes publicados fue verificada mediante los registros del ***módulo Communicator***, donde se evidenció la correcta carga del controlador *MQTT* y la llegada de datos desde el *broker* configurado. En esta fase no se definieron aún canales ni mecanismos de visualización, limitando el alcance de la prueba a la comprobación de la comunicación y disponibilidad de los datos.

Con esta configuración, el simulador *MQTT Explorer* quedó preparado como fuente de datos basada en mensajería para el sistema *SCADA*. La asociación de estos datos a

canales internos y su posterior visualización se aborda en las secciones siguientes del capítulo.

4.7 Creación de la estructura de datos y generación de canales

Una vez registradas las líneas de comunicación y los dispositivos asociados, se procedió a definir la estructura de datos que sería interpretada por *RapidSCADA* para ambos protocolos utilizados en el entorno simulado: *Modbus TCP* y *MQTT*. En esta etapa se estableció la correspondencia entre los datos recibidos desde los simuladores y los canales internos del sistema, sin aplicar aún configuraciones de límites, alarmas ni visualización.

4.7.1 Estructura de datos y canales para *Modbus TCP*

Para el protocolo *Modbus TCP*, la definición de la estructura de datos se realizó mediante una plantilla de dispositivo creada con el *Device Template Editor*. La plantilla utilizada se denominó *DrvModbus_Canales.xml* y fue construida desde cero, de acuerdo con los registros configurados previamente en el simulador *ModRSsim2*.

La plantilla incluyó dos grupos de elementos:

- *Holding Registers (registros de 16 bits),* correspondientes a variables analógicas, con dos registros definidos:
  - *Temp 1*
  - *Temp 2*

- *Coils (variables binarias)*, correspondientes a variables digitales, con dos registros definidos:
  - *Coil 1*
  - *Coil 2*

La organización de estos grupos dentro del *Device Template Editor*, así como su asignación a los bloques de datos correspondientes, se muestra en la Figura 4.7.1.

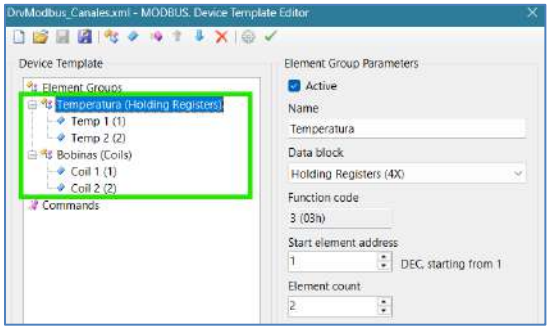
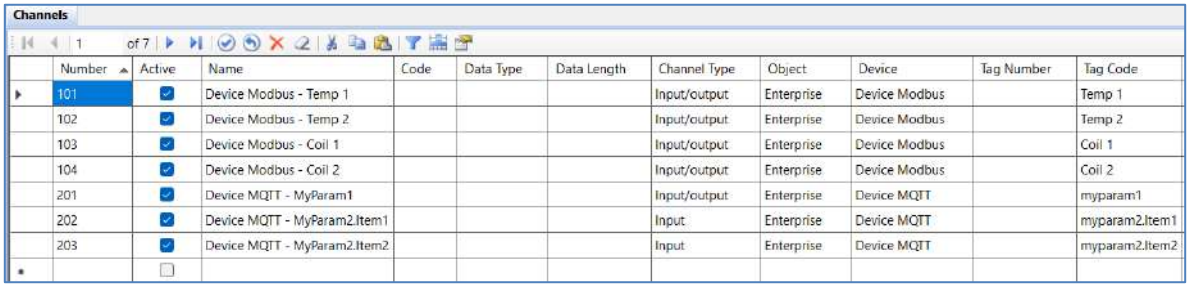


Figura 4.7.1 Editor de plantilla Device mostrando los grupos Holding Registers y Coils definidos.
Fuente: Elaboración propia.

En ambos casos se configuró la dirección inicial *Modbus* en *1*, utilizando los códigos de función estándar *03h* para la lectura de *holding registers* y *01h* para la lectura de *coils*. No se aplicaron conversiones, escalados ni restricciones adicionales dentro de la plantilla. Como resultado de esta configuración, *RapidSCADA* generó automáticamente los canales asociados a cada registro definido, los cuales se muestran en la Figura 4.7.2.



Number	Active	Name	Code	Data Type	Data Length	Channel Type	Object	Device	Tag Number	Tag Code
101	☑	Device Modbus - Temp 1				Input/output	Enterprise	Device Modbus		Temp 1
102	☑	Device Modbus - Temp 2				Input/output	Enterprise	Device Modbus		Temp 2
103	☑	Device Modbus - Coil 1				Input/output	Enterprise	Device Modbus		Coil 1
104	☑	Device Modbus - Coil 2				Input/output	Enterprise	Device Modbus		Coil 2
201	☑	Device MQTT - MyParam1				Input/output	Enterprise	Device MQTT		myparam1
202	☑	Device MQTT - MyParam2.Item1				Input	Enterprise	Device MQTT		myparam2.Item1
203	☑	Device MQTT - MyParam2.Item2				Input	Enterprise	Device MQTT		myparam2.Item2
*	☐									

Figura 4.7.2 Canales generados automáticamente a partir de la plantilla Devices Template en RapidSCADA.
Fuente: Elaboración propia.

Una vez definida la plantilla, se utilizó la herramienta *Create Channels* del ***módulo Administrator*** para generar automáticamente los canales asociados. Este procedimiento permitió crear los canales de forma consistente con la estructura de la plantilla, evitando configuraciones manuales repetitivas y asegurando la correspondencia directa entre registros *Modbus* y canales internos del sistema.

4.7.2 Estructura de datos y canales para *MQTT*

Para el protocolo *MQTT*, la definición de la estructura de datos no se realizó mediante plantillas, sino a través de la configuración del dispositivo *MQTT* en el ***módulo Communicator***, utilizando las propiedades del *MQTT Client*.

Se configuraron dos suscripciones:

- *MyParam1,* utilizada para una publicación básica sin procesamiento adicional. En esta suscripción no se habilitó la ejecución de *scripts* y el canal generado se configuró como entrada/salida, lo que permitió verificar la recepción directa de mensajes *MQTT*.
- *MyParam2*, utilizada para la recepción de mensajes en formato *JSON*. En esta suscripción se habilitó la opción *JavaScript enabled* y se asoció el archivo de *script MyParam2.js*, creado dentro del proyecto. Este *script* permitió interpretar el contenido del mensaje *JSON* y asignar los valores recibidos a *sub-ítems* internos.

La configuración de ambas suscripciones en el dispositivo MQTT Client se muestra en la Figura 4.7.3.

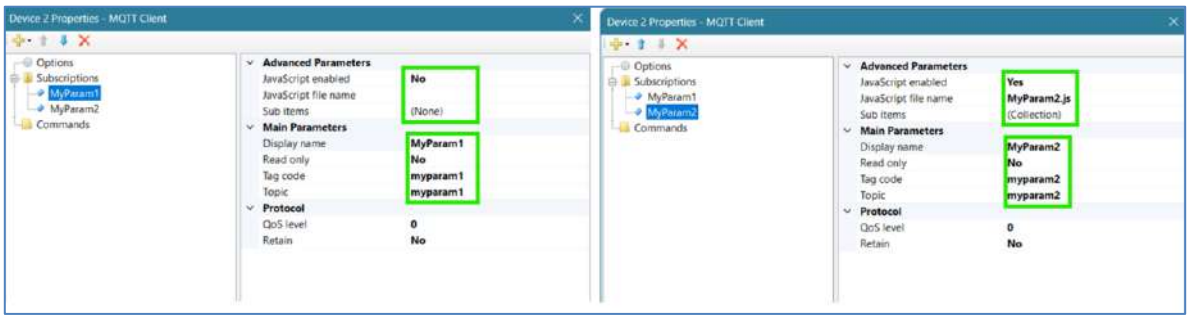


Figura 4.7.3 Configuración de suscripciones MQTT en el dispositivo MQTT Client de RapidSCADA.
Fuente: Elaboración propia

El mensaje publicado desde el simulador MQTT presentó la siguiente estructura lógica mostrada en la ecuación 1, la cual fue utilizada como entrada para la interpretación de datos en el sistema.

Ecuación 1 Mensaje publicado desde el simulador MQTT

$$\{\text{"}item1\text{"}: valor, \text{"}item2\text{"}: valor\}$$

Una vez configuradas las suscripciones del dispositivo *MQTT* en el ***módulo Communicator***, generé automáticamente los canales asociados a cada suscripción definida. Este proceso se realizó a partir de la estructura establecida en las propiedades del *MQTT Client*, incluyendo los tópicos suscritos y, en el caso de mensajes en formato *JSON*, los *sub-ítems* definidos mediante el *script* de interpretación. El resultado de este proceso se muestra en la Figura 4.7.4.

Number	Active	Name	Code	Data Type	Data Length	Channel Type	Object	Device	Tag Number	Tag Code
201	☑	Device MQTT - MyParam1				Input/output	Enterprise	Device MQTT		myparam1
202	☑	Device MQTT - MyParam2.Item1				Input	Enterprise	Device MQTT		myparam2.Item1
203	☑	Device MQTT - MyParam2.Item2				Input	Enterprise	Device MQTT		myparam2.Item2

Figura 4.7.4 Canales MQTT generados a partir de la configuración del dispositivo MQTT Client en RapidSCADA.
Fuente: Elaboración propia

De esta manera, los valores recibidos desde el *broker MQTT* quedaron vinculados directamente a canales internos del sistema, sin requerir la creación manual de canales ni la definición explícita de plantillas de comunicación.

4.8 Creación y configuración de vistas de supervisión en Webstation

Una vez generados los canales asociados a los protocolos *Modbus TCP* y *MQTT*, se procedió a la creación de las vistas de supervisión en *RapidSCADA* con el fin de visualizar los datos adquiridos por el sistema desde el ***módulo Webstation***. Esta etapa se centró exclusivamente en la configuración y organización de las vistas, sin modificar la estructura de comunicación, los dispositivos ni los canales previamente definidos.

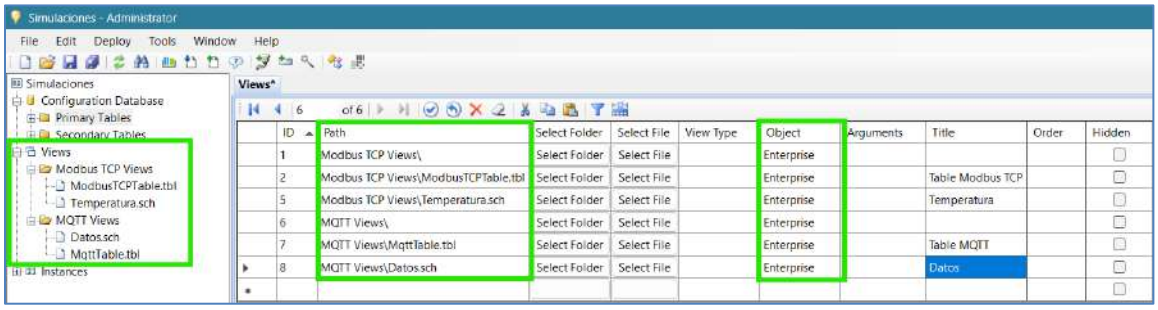


Figura 4.8.1 Configuración de vistas en el módulo Administrator de RapidSCADA.
Fuente: Elaboración propia.

El proceso se realizó desde el ***módulo Administrator***, en la sección *Views* de la base de configuración. En primer lugar, se crearon carpetas para organizar las vistas según el protocolo de comunicación, estableciendo una separación clara entre las vistas correspondientes a *Modbus TCP* y las asociadas a *MQTT*, tal como se muestra en la Figura 4.8.1. Posteriormente, dentro de cada carpeta se crearon vistas de tipo tabla, seleccionando el archivo de vista correspondiente y asociándolo al objeto *Enterprise*, de acuerdo con la estructura definida en la base de datos de configuración.

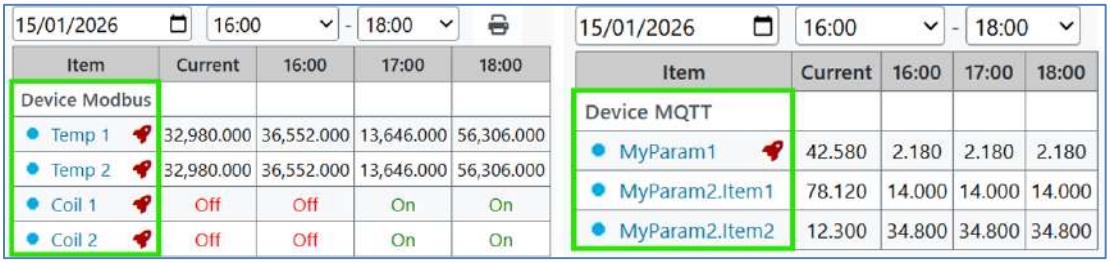


Figura 4.8.2 Visualización de canales *Modbus TCP* y *MQTT* en la vista tipo tabla de *Webstation*.
Fuente: Elaboración propia

Para las vistas asociadas al protocolo *Modbus TCP*, se incorporaron los canales correspondientes a las variables analógicas y digitales configuradas previamente, incluyendo las temperaturas simuladas y los estados de los *coils*. En el caso del protocolo *MQTT*, se agregaron los canales generados a partir de las suscripciones configuradas en el dispositivo *MQTT*, correspondientes tanto a la publicación básica como a la interpretación de mensajes en formato *JSON*. La visualización de estos canales en vistas tipo tabla desde ***Webstation*** se presenta en la Figura 4.8.2.

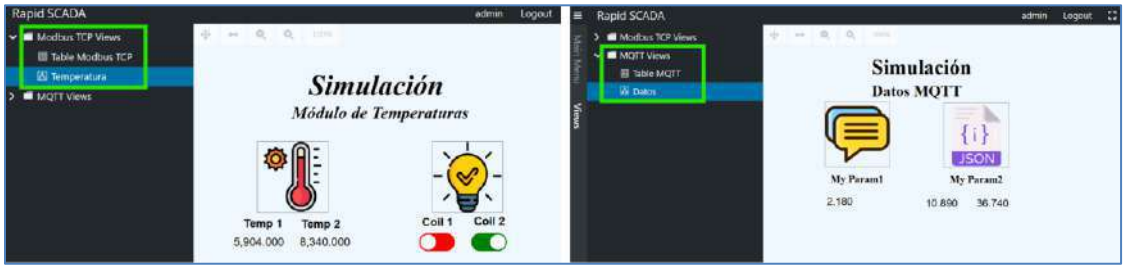


Figura 4.8.3 Vistas esquemáticas en Webstation para Modbus TCP y MQTT.
Fuente: Elaboración propia

Adicionalmente, se crearon vistas de tipo *Schematic*, en las cuales se incorporaron elementos gráficos básicos y se asociaron canales del sistema para permitir la visualización de valores analógicos, la representación de estados digitales y la interacción básica con los canales configurados como entrada/salida. Las vistas esquemáticas desarrolladas para los protocolos Modbus TCP y MQTT se muestran en la Figura 4.8.3.

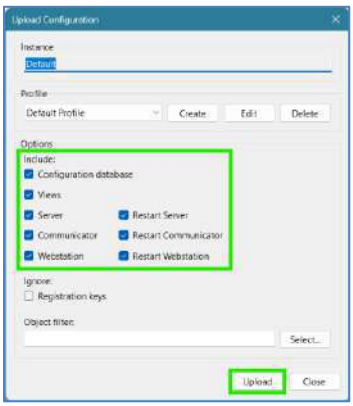


Figura 4.8.4 Ventana Upload de la configuración del proyecto en RapidSCADA desde el módulo Administrator.
Fuente: Elaboración propia.

Una vez finalizada la creación de las vistas, se ejecutó el proceso de *Upload Configuration* desde el ***módulo Administrator*** para aplicar los cambios en el entorno de ejecución, como se ilustra en la Figura 4.8.4. Posteriormente, se verificó el perfil activo de ***Webstation*** y el acceso local a la plataforma mediante la dirección configurada en el sistema como se evidencia en la figura 4.8.5. Desde la interfaz ***Webstation*** se comprobó la correcta visualización de las vistas organizadas por protocolo, la actualización de los valores provenientes de *Modbus TCP*, la visualización de los mensajes recibidos mediante *MQTT* y la interacción básica con los canales habilitados, confirmando el funcionamiento adecuado de la supervisión del sistema en un entorno completamente simulado.

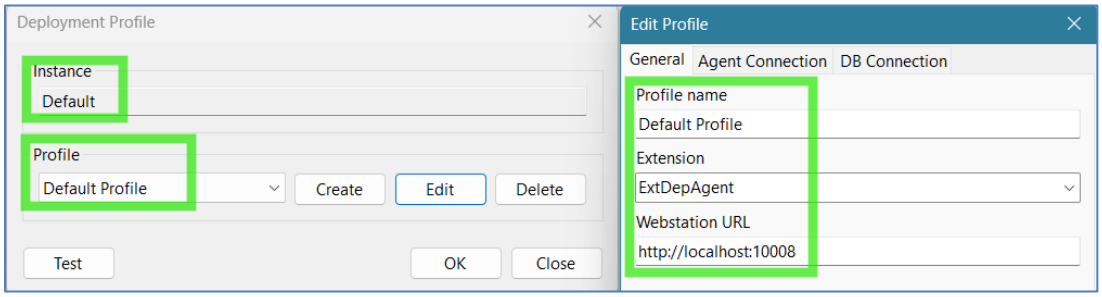


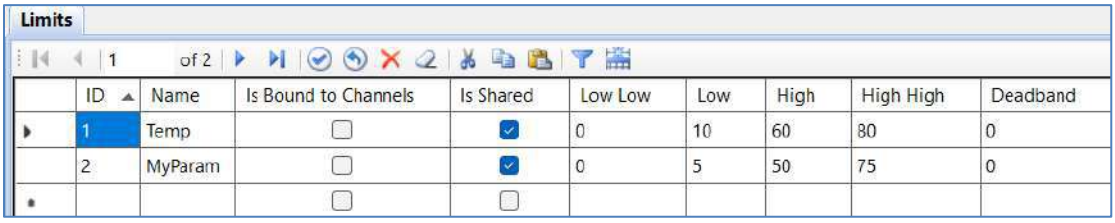
Figura 4.8.5 Perfil de despliegue (Deployment Profile) utilizado para aplicar la configuración del proyecto en RapidSCADA.
Fuente: Elaboración propia.

4.9 Alarmas, eventos y registro histórico

Con la estructura de canales y las vistas de supervisión ya operativas, se habilitaron las funciones de alarmas, eventos y registro histórico con el fin de validar el comportamiento del sistema *SCADA* ante variaciones en las variables simuladas y cambios de estado generados durante la operación. Esta configuración se realizó tanto para los canales

asociados al protocolo *Modbus TCP* como para aquellos correspondientes al protocolo *MQTT*, manteniendo un enfoque exclusivamente funcional.

En primer lugar, se definieron grupos de límites desde la tabla *Limits* del ***módulo Administrator***. Se creó un grupo denominado *Temp*, aplicado a los canales analógicos *Temp 1* y *Temp 2* del dispositivo *Modbus TCP*, y un grupo denominado *MyParam*, aplicado a los canales *MQTT* (*MyParam1, MyParam2.Item1* y *MyParam2.Item2*). La configuración de estos límites, incluyendo los umbrales inferiores y superiores, se muestra en la Figura 4.9.1.



	ID	Name	Is Bound to Channels	Is Shared	Low Low	Low	High	High High	Deadband
▶	1	Temp	☐	☑	0	10	60	80	0
	2	MyParam	☐	☑	0	5	50	75	0
*			☐	☐					

Figura 4.9.1 Configuración de los límites para los canales
Fuente: Elaboración propia

Los valores definidos para los límites fueron seleccionados únicamente para provocar cruces controlados de umbrales durante la simulación. Estos valores no representan condiciones reales de un proceso industrial, sino que se establecieron con fines de validación funcional del sistema.

Posteriormente, los grupos de límites fueron asociados a los canales correspondientes desde la tabla *Channels*. A partir de esta asociación, *RapidSCADA* quedó en capacidad de generar eventos automáticamente cuando los valores de los canales superaban o descendían por debajo de los rangos definidos.

De forma complementaria, se habilitó el registro de eventos mediante la configuración de la máscara de eventos (*Event Mask*) y la selección del tipo de archivo histórico en los canales. Para los canales analógicos y *MQTT* se activaron eventos asociados a cambios de valor y cambios de estado, mientras que en los canales digitales *Modbus TCP* se incluyó adicionalmente el registro de eventos por envío de comandos desde la interfaz ***Webstation***. La configuración de estas máscaras y opciones de archivo se presenta en la Figura 4.9.2.

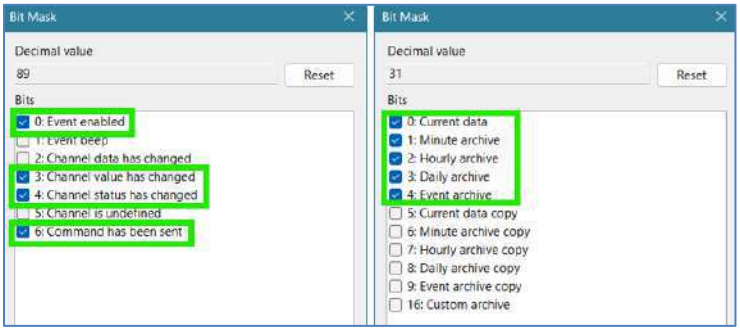


Figura 4.9.2 Máscara de eventos y archivo seleccionada para los canales.
Fuente: Elaboración propia

En cuanto al almacenamiento de datos, se utilizó el sistema de archivos históricos configurado por defecto en la plataforma. A todos los canales del proyecto se les asignó una máscara de archivo con valor *31*, habilitando el registro de datos actuales, archivos por minuto, por hora, por día y el archivo de eventos. Esta configuración permitió que el sistema almacenara automáticamente tanto los valores adquiridos como los eventos generados durante la simulación, sin necesidad de crear archivos históricos adicionales.

Finalmente, desde la interfaz ***Webstation*** se verificó que los eventos generados por cambios en las variables analógicas, estados digitales y mensajes *MQTT* eran registrados correctamente. La visualización de los eventos capturados durante la simulación se presenta en la Figura 4.9.3, confirmando la correcta operación de los mecanismos de alarmas y eventos del sistema.

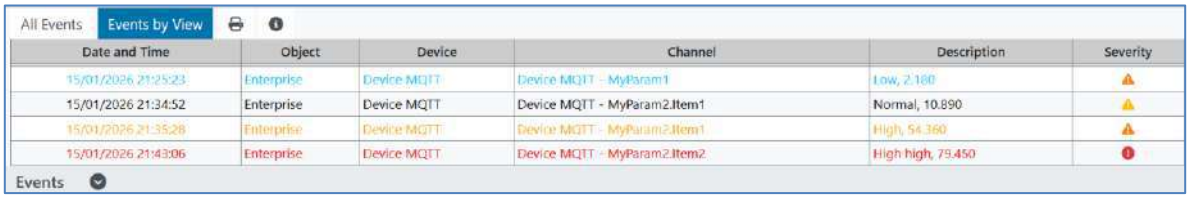


Figura 4.9.3 Eventos capturados durante la simulación *Modbus TCP*.
Fuente: Elaboración propia

La coherencia entre los valores simulados, los eventos registrados y la información visualizada confirmó el correcto funcionamiento de los mecanismos de alarmas, eventos y registro histórico en el entorno completamente simulado.

4.10 Resultados de validación del sistema simulado

La validación del sistema se realizó sobre el entorno completamente simulado, con el objetivo de comprobar la coherencia entre la adquisición de datos, el registro interno y la visualización en ***Webstation***, una vez finalizada la configuración de canales, eventos y almacenamiento histórico.

Para el protocolo *Modbus TCP*, los valores generados en el simulador fueron recibidos por *RapidSCADA* y asociados correctamente a los canales definidos. Las variables analógicas y digitales reflejaron los cambios esperados tanto en las vistas de supervisión como en los registros de eventos y archivos históricos. Las conmutaciones realizadas sobre los canales digitales desde la interfaz web quedaron registradas como eventos del sistema, confirmando la correcta propagación de los datos desde el simulador hasta los ***módulos Communicator*** y ***Webstation***, tal como se observa en la Figura 4.10.1.

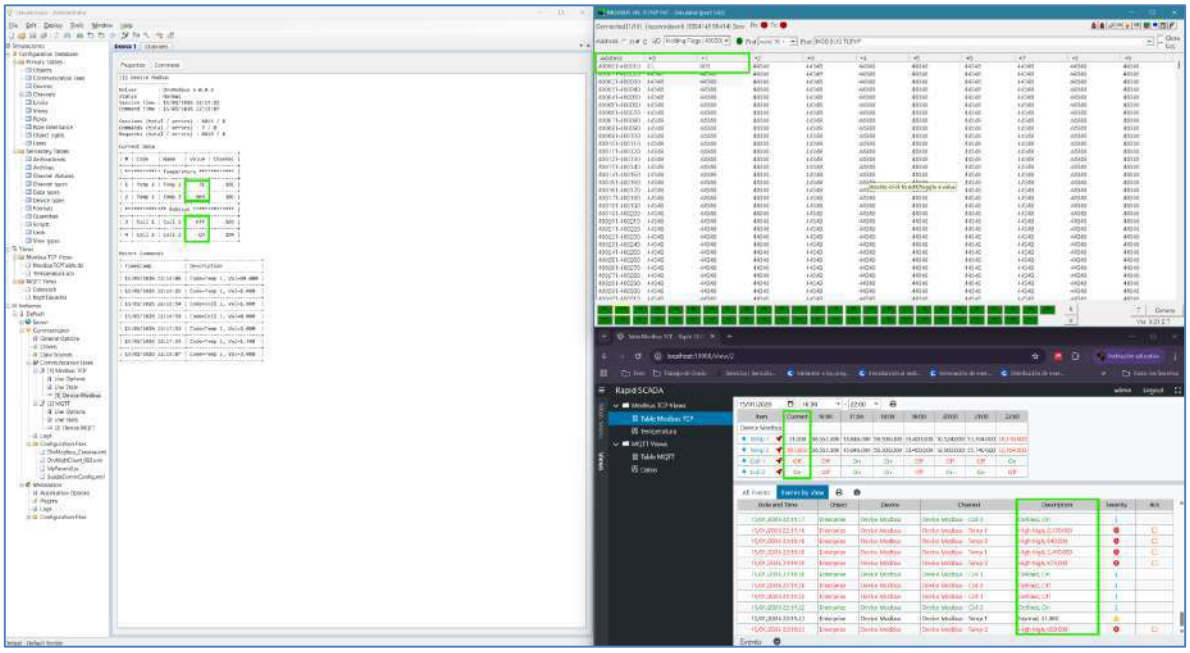


Figura 4.10.1 Validación del sistema simulado de *Modbus TCP*: correspondencia entre simulador, *Communicator* y *Webstation*.
Fuente: Elaboración propia

En el caso del protocolo *MQTT*, los mensajes publicados desde el simulador fueron procesados conforme a la estructura configurada en el dispositivo *MQTT*. Los valores asociados a las suscripciones definidas se reflejaron en los canales correspondientes y pudieron ser consultados tanto en las vistas de supervisión como en el historial del sistema. La correspondencia entre los mensajes publicados, los canales generados y los eventos registrados durante la simulación se presenta en la Figura 4.10.2.

El registro histórico permitió consultar la evolución temporal de las variables simuladas dentro de los intervalos definidos, confirmando la correcta asociación de los canales a los archivos de almacenamiento. Durante las pruebas no se observaron inconsistencias entre los datos adquiridos, los eventos generados y la información visualizada.

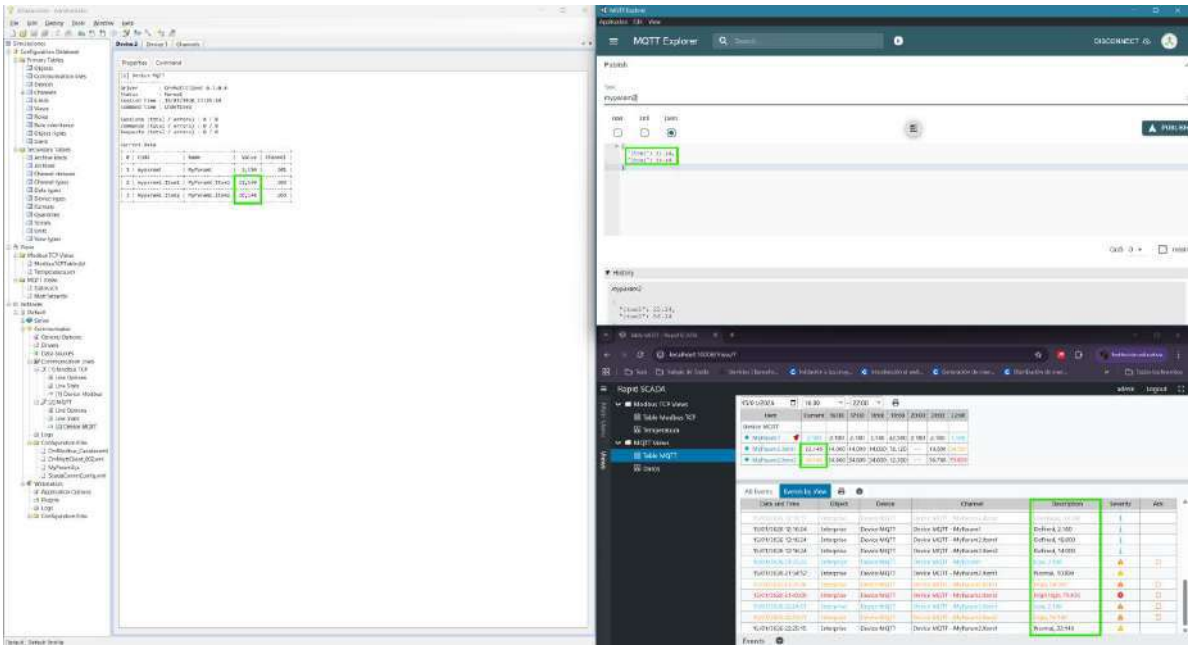


Figura 4.10.2 Validación del sistema simulado de *MQTT*: correspondencia entre simulador, Communicator y Webstation.
Fuente: Elaboración propia

Con base en estas verificaciones, se confirma que el sistema *SCADA* básico configurado opera de forma coherente en un entorno virtual, quedando preparado para la etapa de integración con dispositivos físicos que se desarrolla en el capítulo siguiente.

5. Integración con hardware físico mediante Modbus RTU

5.1 Introducción y alcance del capítulo

Este capítulo describe la transición desde el entorno simulado trabajado previamente hacia un escenario de comunicación *Modbus RTU* con *hardware físico*, utilizando *RapidSCADA* como sistema de supervisión. El objetivo es validar el funcionamiento del sistema cuando intervienen elementos propios de un entorno real, tales como el bus *RS-485* compartido, la asignación de direcciones *Modbus*, la configuración del puerto serie y la interacción directa con dispositivos industriales y educativos.

La integración se realizó mediante dos montajes independientes en *RapidSCADA*

- Un sistema industrial basado en módulos *ADAM-4018+* y *ADAM-4024*, conectados a un bus *RS-485* común.
- Un nodo didáctico basado en *Arduino Nano*, operando como esclavo *Modbus RTU* a través de conexión *USB*.

El capítulo se centra en los procedimientos reales de montaje, configuración, validación funcional y análisis de errores, sin reiterar conceptos teóricos del protocolo ni de la arquitectura *SCADA* ya tratados en capítulos anteriores.

5.2 Preparación de los módulos ADAM y adecuación a Modbus RTU

El Sistema corresponde a un lazo básico de control de temperatura implementado con módulos de adquisición y salida de la familia ADAM, comunicados mediante el protocolo Modbus RTU. La arquitectura general del sistema y la interconexión entre sus componentes se presentan en la Figura 5.2.1.

El módulo ADAM-4018+ es un módulo de adquisición configurable de ocho (8) entradas analógicas, utilizado en este proyecto para la lectura de termopares tipo K y señales de corriente en el rango 0–20 mA. Cada canal puede configurarse de forma independiente, permitiendo la adquisición simultánea de distintas magnitudes físicas.

Por su parte, el módulo ADAM-4024 es un módulo de salidas analógicas y digitales, que dispone de dos (2) salidas analógicas configurables en corriente (0–20 mA, 4–20 mA) o voltaje (0–10 V), así como cuatro (4) salidas digitales. En el sistema implementado, las salidas analógicas se emplean para accionar elementos de potencia y señalización.

Ambos módulos se comunican mediante el protocolo Modbus RTU utilizando comandos ASCII propios de la serie ADAM, lo que permite la lectura de variables de proceso y el envío de comandos de control desde el sistema SCADA.

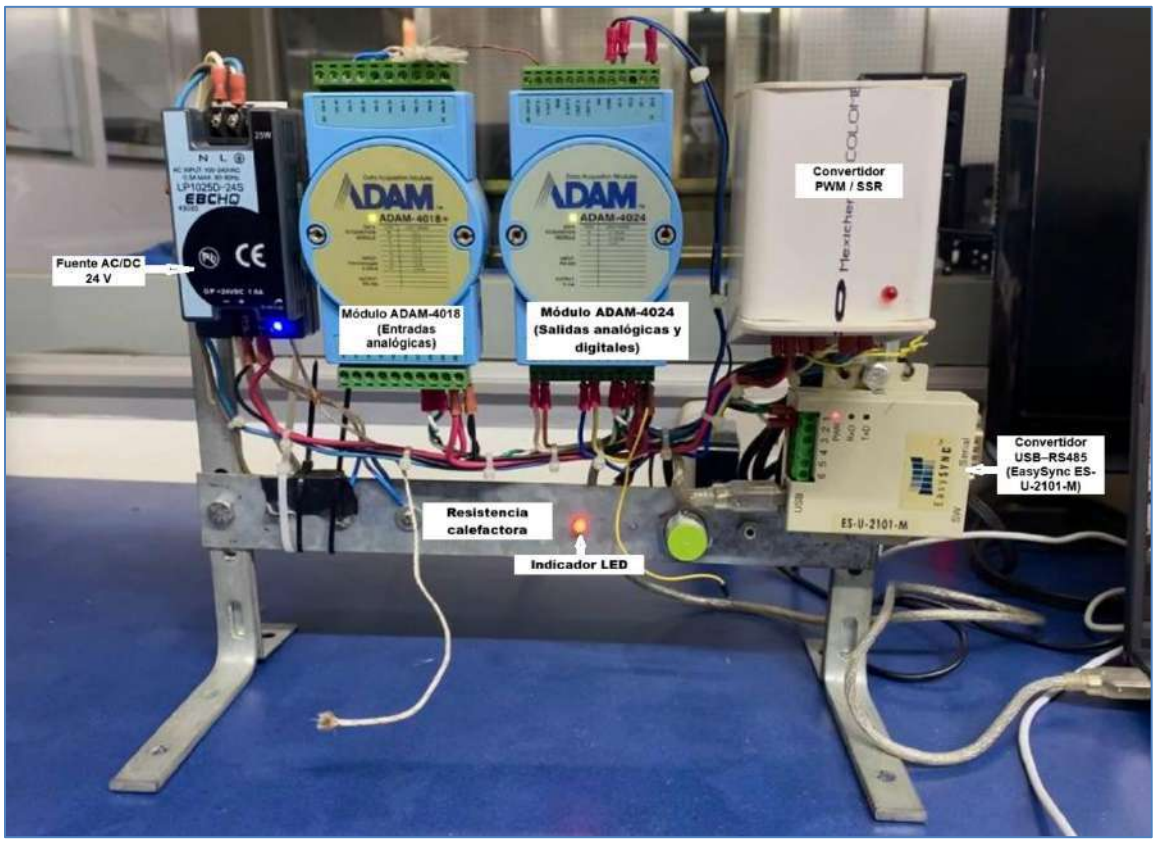


Figura 5.2.1 Montaje físico del sistema ADAM.
Fuente: Elaboración propia

Antes de integrar los módulos *ADAM* en *RapidSCADA*, fue necesario adecuarlos al protocolo *Modbus RTU*, ya que inicialmente se encontraban configurados en *ASCII*. Este procedimiento se realizó utilizando la herramienta *Adam 4000–5000 Utility*, siguiendo el flujo operativo recomendado por el fabricante.

Para cada módulo se efectuaron los siguientes pasos en una red *RS-485* aislada:

1. Energización individual del módulo.
2. Activación del modo de inicialización mediante el interruptor *INIT* (o conexión *INIT–GND* según el modelo).
3. Detección del módulo desde *Adam Utility* con parámetros por defecto *(9600, 8N1).*
4. Cambio explícito del protocolo de comunicación a *Modbus RTU*.
5. Asignación de una dirección *Modbus* única para cada equipo.
6. Reinicio del módulo en modo normal.

Una vez completado este proceso, se verificó la respuesta de cada módulo desde la misma herramienta, confirmando la comunicación antes de proceder a la integración en *RapidSCADA*. Esta verificación previa permitió descartar fallos físicos o de configuración básica antes de involucrar el sistema *SCADA*.

5.3 Integración de los módulos ADAM en el bus RS-485

Una vez finalizada la preparación lógica de los módulos *ADAM* descrita en la sección anterior, los dispositivos fueron integrados al bus *RS-485* para su operación conjunta con *RapidSCADA*.

En esta etapa no se realizaron modificaciones sobre la configuración interna de los módulos ni sobre el cableado previamente instalado. El objetivo fue verificar la correcta operación del bus *RS-485* como una línea multipunto única, condición necesaria para el funcionamiento del protocolo *Modbus RTU*, así como la correcta identificación del puerto de comunicación por parte del sistema operativo.

Ambos módulos fueron conectados simultáneamente al bus *RS-485* y asociados a un único puerto serie reconocido por el sistema operativo como *COM4*. Esta verificación se realizó mediante la herramienta de configuración de los dispositivos *ADAM*, donde se comprobó la disponibilidad del puerto y los parámetros básicos de comunicación serie, tal como se muestra en la Figura 5.3.1.

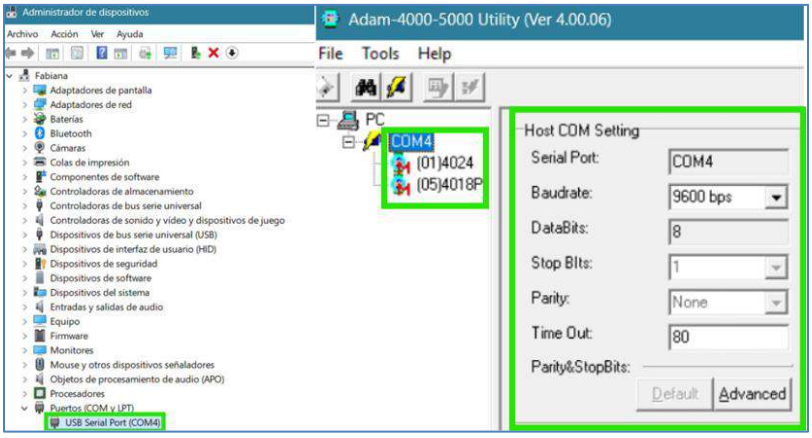


Figura 5.3.1 Comprobación del puerto *COM* y parámetros de comunicación serie utilizados por los módulos *ADAM* previo a la configuración de la línea *Modbus RTU* en *RapidSCADA*.
Fuente: Elaboración propia

La correcta identificación del puerto *COM* permitió que *RapidSCADA* actuara posteriormente como maestro *Modbus RTU*, realizando operaciones de lectura y escritura sobre los distintos dispositivos esclavos de acuerdo con la dirección previamente asignada a cada módulo.

Durante las pruebas se observó que cualquier inconsistencia en la configuración del bus, como direcciones *Modbus* duplicadas o errores en la conexión de la línea *RS-485*, provocaba la interrupción total de la comunicación. Este comportamiento es inherente al protocolo *Modbus RTU* en topologías multipunto y se analiza posteriormente como parte del estudio de errores inducidos.

5.4 Configuración de la línea *Modbus RTU* en *RapidSCADA*

Una vez verificada la correcta preparación de los módulos *ADAM* y confirmada la comunicación previa mediante la herramienta *Adam 4000–5000 Utility como se muestra en la figura 5.4.1*, previo a la integración con *RapidSCADA*.

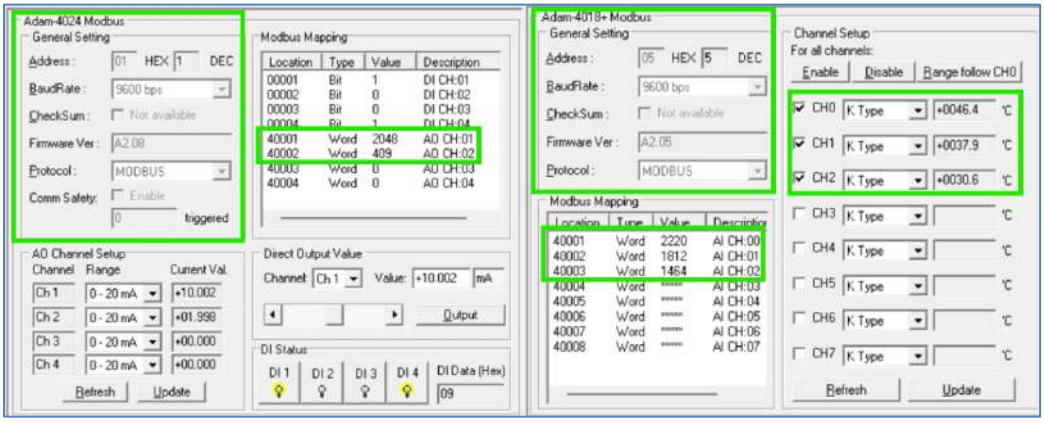


Figura 5.4.1 Verificación de direcciones Modbus y mapeo de registros en los módulos ADAM-4018+ y ADAM-4024 mediante la herramienta Adam 4000–5000 Utility, previa a la integración con RapidSCADA.
Fuente: Elaboración propia.

Para el sistema *ADAM* se creó una única línea de comunicación *Modbus RTU para ambos dispositivos industriales*, asociada al puerto *COM4 se observa en la Figura 5.4.2*, donde se evidencia la asignación del controlador, las direcciones Modbus y los parámetros de sondeo configurados.

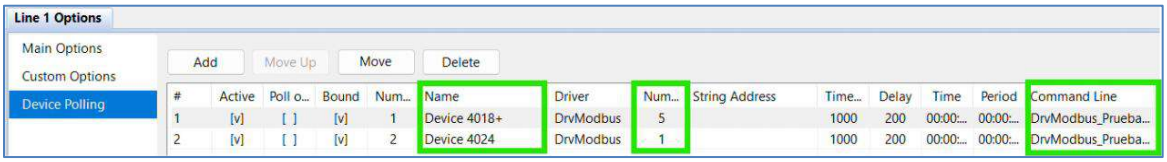


Figura 5.4.2 Línea Modbus RTU configurada en RapidSCADA con registro conjunto de los módulos ADAM-4018+ y ADAM-4024 sobre el puerto COM4.
Fuente: Elaboración propia.

Durante las pruebas iniciales se evaluó una configuración incorrecta, en la cual cada módulo *ADAM* fue asignado a líneas de comunicación independientes, aun cuando ambos compartían físicamente el mismo bus *RS-485*. Esta disposición provocó la pérdida total de comunicación, evidenciada por el estado anómalo de la línea y la necesidad de reiniciar los servicios del sistema.

La prueba permitió confirmar que, en *Modbus RTU*, todos los dispositivos conectados a un mismo bus físico deben gestionarse dentro de una única línea de comunicación en *RapidSCADA*, y que la separación lógica de líneas sobre un mismo puerto *COM* genera conflictos que impiden la operación del sistema.

Una vez consolidada correctamente la línea *RTU*, se procedió al registro y validación funcional de los módulos. En primer lugar, se incorporó el *ADAM-4018+,* verificando la lectura estable de los canales asociados a los termopares. Tras confirmar la coherencia y continuidad de estas lecturas, se registró el *ADAM-4024*, destinado a la escritura de señales analógicas tal como se presenta en la figura 5.4.3.

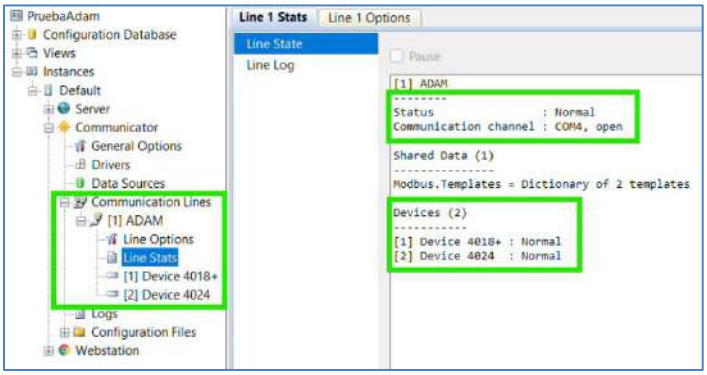


Figura 5.4.3 Estado operativo de la línea *Modbus RTU* en *RapidSCADA*, mostrando comunicación normal y reconocimiento correcto de ambos dispositivos *ADAM*.
Fuente: Elaboración propia.

Las pruebas de escritura en el *ADAM-4024* se realizaron enviando consignas de *0, 10 y 20 mA* desde ***Webstation***, mediante vistas tipo tabla como mediante vistas esquemáticas como se ilustra en la Figura 5.4.4.

Figura 5.4.4 Montaje físico del sistema *ADAM (4018+ y 4024)* y verificación de escritura de consignas desde Webstation mediante *Modbus RTU*.
Fuente: Elaboración propia.

Inicialmente se presentaron inconsistencias en la interpretación de los comandos, las cuales se corrigieron ajustando el tipo de canal y definiendo fórmulas de entrada y salida en los canales correspondientes del módulo Administrator. Las expresiones utilizadas, así como su ubicación dentro de la configuración del sistema, se documentan en la Figura A.1 del Anexo A y la Figura B.1 del Anexo B, garantizando que los valores enviados desde la interfaz correspondieran correctamente a la lógica interna del módulo.

La validación funcional confirmó la coherencia entre los valores escritos desde ***Webstation***, la respuesta física observada en el módulo mediante los indicadores *LED* y la estabilidad de la comunicación sobre el bus *RS-485* compartido durante la ejecución continua de las pruebas.

5.5 Integración del Arduino Nano como nodo Modbus RTU independiente

De forma paralela al sistema industrial basado en módulos *ADAM*, se integró un *Arduino Nano* como nodo *Modbus RTU* independiente, con el propósito de evaluar la viabilidad de utilizar dispositivos de bajo costo como recursos didácticos dentro de una arquitectura *SCADA*. Este nodo no compartió el bus *RS-485* con los módulos industriales, sino que se conectó directamente al computador mediante el puerto *USB*, siendo reconocido por el sistema operativo como *COM3*. Esta separación permitió mantener aisladas las pruebas del bus industrial y evitar interferencias durante la validación del sistema *ADAM*.

En el microcontrolador se cargó un *sketch* que implementa el protocolo *Modbus RTU* mediante la librería *modbusSlave* que se presenta en la figura 5.5.1. El programa fue diseñado para publicar tres señales periódicas generadas por *software* (una senoidal, una triangular y una rampa), las cuales se almacenan en registros *holding* consecutivos.

Las señales fueron representadas mediante el tipo de dato *uint16_t*, utilizando un rango de valores entre 0 y 1000, lo que permitió su transmisión directa como registros de 16 *bits* sin signo, compatibles con la estructura estándar de *Modbus RTU*. Esta representación facilitó el tratamiento del Arduino como un esclavo *Modbus* convencional desde el punto de vista del sistema *SCADA*.

```ino
#include <modbusSlave.h>

modbusDevice regBank;
modbusSlave  slave;

// Parámetros de señal
const uint16_t MAX_VAL = 1000;      // Escala para graficar cómodamente
const uint16_t UPDATE_MS = 100;     // 100 ms -> 10 Hz de muestreo

// Triángulo
int16_t tri = 0;
int16_t triInc = 10;

// Rampa
uint16_t ramp = 0;

// Control de periodo
unsigned long tPrev = 0;

void setup() {
  Serial.begin(9600);     // 9600 8N1
  regBank.setId(1);       // ID esclavo
  // Define holding registers base-0 -> 40001, 40002, 40003
  regBank.add(40001);     // idx 0
  regBank.add(40002);     // idx 1
  regBank.add(40003);     // idx 2
  slave._device = &regBank;
  slave.setBaud(9600);
}
```

```ino
void loop() {
  unsigned long now = millis();
  if (now - tPrev >= UPDATE_MS) {
    tPrev = now;

    // Tiempo en segundos (float) para el seno
    float t = now / 1000.0;

    // SENOIDAL  (0..MAX_VAL)
    // Período aprox 10 s: sin(2π * t / 10)
    float s = sin(2.0 * 3.1415926 * (t / 10.0));
    uint16_t sineVal = (uint16_t)((s * 0.5 + 0.5) * MAX_VAL);

    // TRIANGULO (0..MAX_VAL) - sube y baja
    tri += triInc;
    if (tri >= (int)MAX_VAL || tri <= 0) triInc = -triInc;
    uint16_t triVal = (uint16_t)tri;

    // RAMPA (0..MAX_VAL)
    ramp = (ramp + 5) % (MAX_VAL + 1);

    // Publicar en HRs
    regBank.set(40001, sineVal); // 40001: seno
    regBank.set(40002, triVal);  // 40002: triángulo
    regBank.set(40003, ramp);    // 40003: rampa
  }

  slave.run(); // Atender maestro Modbus
}
```

Figura 5.5.1 Sketch cargado en el Arduino Nano para la generación de señales periódicas.
Fuente: Elaboración propia

Antes de habilitar la visualización en ***Webstation***, se verificó la correcta lectura de los registros desde *RapidSCADA*, comprobando que el microcontrolador respondía de forma estable a las solicitudes *Modbus* mientras el puerto permanecía activo. Esta validación previa permitió confirmar que la comunicación *RTU* funcionaba correctamente a nivel de

protocolo y que los valores publicados eran coherentes con las señales generadas por el *sketch*.

El montaje físico del Arduino Nano y su integración como nodo *Modbus RTU* independiente se muestran en la Figura 5.5.2.

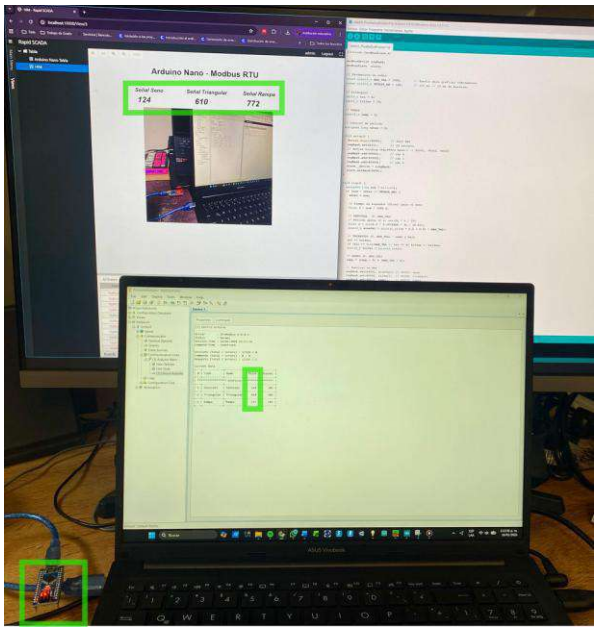


Figura 5.5.2 Montaje físico del Arduino Nano integrado mediante Modbus RTU.
Fuente: Elaboración propia.

El uso del *Arduino Nano* en este contexto permitió mostrar que, mediante una configuración adecuada tanto del microcontrolador como del sistema *SCADA*, es posible realizar pruebas funcionales de adquisición de datos *Modbus* utilizando equipos de bajo costo. Este enfoque resulta especialmente útil en entornos académicos, ya que facilita la comprensión del funcionamiento interno del protocolo y permite reproducir escenarios de supervisión sin depender exclusivamente de *hardware* industrial especializado.

5.6 Análisis del bus Modbus RTU y errores observados

Durante la integración del bus *Modbus RTU* se identificaron fallas típicas asociadas principalmente a la configuración del sistema y no al *hardware* empleado. Estas situaciones se presentaron tanto de manera espontánea durante las pruebas como de forma intencional, con el objetivo de evaluar el comportamiento del bus *RS-485* y los mecanismos de diagnóstico de *RapidSCADA*.

Los errores observados estuvieron relacionados con la asignación incorrecta de direcciones *Modbus*, el uso inadecuado del puerto *COM*, la creación de líneas *RTU* independientes sobre un mismo bus físico y configuraciones incompletas dentro del proyecto, como límites no asignados o archivos históricos no vinculados. En todos los casos, las fallas se

manifestaron como pérdida de comunicación, ausencia de datos en ***Webstation*** o estados anómalos en el ***módulo Communicator***.

La Tabla 5.6.1 resume los errores más representativos detectados durante las pruebas y su causa probable. Este análisis permite establecer criterios prácticos para el diagnóstico de fallas en sistemas *Modbus RTU* y resalta la importancia de mantener una única línea *RTU* por bus físico, direcciones diferenciadas y una configuración coherente del proyecto en *RapidSCADA*.

Tabla 5.6.1 Errores típicos durante la integración *Modbus RTU* y su causa probable
Fuente: Elaboración propia

Error observado	Causa probable
La comunicación cae al conectar dos módulos *ADAM* al mismo bus	Direcciones *Modbus* duplicadas o configuración inicial con dos líneas *RTU* separadas utilizando el mismo puerto *COM*.
Communicator muestra "*Port is closed*"	El puerto *COM* estaba siendo utilizado por otra aplicación o la línea no fue habilitada o cerrada correctamente.
No se muestran valores en ***Webstation***	Las vistas creadas no fueron cargadas al servidor (*Deploy* incompleto) o se seleccionó una vista incorrecta dentro de la estructura del proyecto.
Los límites no aparecen en ***Webstation***	La tabla *Limits* no fue guardada correctamente o no se asignaron rangos válidos a los canales antes del *Deploy*.
La tabla histórica muestra "--" sin datos	El marcador (*archive mask*) no estaba correctamente definido o el canal no fue asignado al archivo histórico correspondiente.
Lecturas intermitentes o mensaje "*no response*"	Tiempos de espera bajos, consultas demasiado rápidas o interferencias puntuales en el bus *RS-485*.
El *ADAM-4024* tarda en responder o deja de actualizar	El módulo requiere tiempo interno para aplicar escrituras analógicas (latencia propia del dispositivo).
Arduino Nano deja de responder en ***Communicator***	El puerto *COM3* cambió de número, fue tomado por otra aplicación o el cable *USB* se desconectó momentáneamente.
Canales duplicados no funcionan	Clones con nombres repetidos o *IDs* en conflicto, lo que impide a *RapidSCADA* asignarlos correctamente.
Valores incoherentes al escribir en un *ADAM*	Línea compartida saturada por consultas simultáneas o fórmula de conversión incorrecta que envía valores fuera de rango.

5.7 Validación funcional del sistema ADAM

La validación del sistema se realizó mediante pruebas funcionales directas sobre los módulos *ADAM-4018+* y *ADAM-4024*, observando la correspondencia entre los valores supervisados en *RapidSCADA* y el comportamiento físico del montaje durante la operación real.

La validación funcional de la lectura de temperatura en el módulo *ADAM-4018+,* mediante la interacción física con termopares tipo *K* y su visualización en la interfaz ***Webstation***, se ilustra en la Figura 5.7.1, donde se evidencia la correspondencia entre la variación térmica aplicada y los valores supervisados por el sistema *SCADA*.

Figura 5.7.1 Validación funcional de la lectura de temperatura en el módulo ADAM-4018+ mediante interacción física con los termopares tipo K.
Fuente: Elaboración propia

Para el *ADAM-4024*, la validación se centró en el envío de consignas de corriente dentro del rango de 0–20 *mA* desde ***Webstation***, permitiendo la interacción directa del usuario con el sistema. La respuesta del módulo se evaluó mediante la observación del comportamiento físico, utilizando los indicadores *LED* integrados como referencia visual inmediata y contrastando dicha respuesta con los valores mostrados en la interfaz *SCADA*, tal como se presenta en la Figura 5.7.2.

Figura 5.7.2 Validación de la escritura de consignas analógicas en el módulo ADAM-4024 mediante RapidSCADA.
Fuente: Elaboración propia

Las consignas y lecturas asociadas a las salidas analógicas se procesaron mediante fórmulas de conversión definidas en los canales correspondientes. En este contexto, durante las pruebas se observaron diferencias entre los valores de consigna y los valores generados por el módulo, evidenciadas directamente en la interfaz del sistema y en el comportamiento físico del montaje.

La comprobación detallada del comportamiento de los módulos durante estas pruebas, incluyendo la correspondencia entre consignas enviadas, respuesta física observada y valores supervisados en el sistema SCADA, se documenta en la Figura H.1 del Anexo H y la Figura I.1 del Anexo I, donde se presentan evidencias adicionales del funcionamiento del ADAM-4018+ y del ADAM-4024, respectivamente.

Durante las pruebas se comprobó que el usuario podía modificar manualmente las consignas desde ***Webstation*** y observar la respuesta del módulo con un retardo mínimo, atribuible al tiempo interno de actualización del dispositivo y al ciclo de comunicación del sistema. Este comportamiento se mantuvo estable durante la ejecución continua de las pruebas.

La validación funcional permitió confirmar la coherencia entre las consignas enviadas, la respuesta física del sistema y los valores supervisados en *RapidSCADA*, demostrando el correcto funcionamiento del montaje industrial bajo condiciones reales de laboratorio.

5.7.1 Consideraciones sobre la interpretación temporal y de error en la validación

La validación presentada en la sección 5.7 permitió verificar la integración funcional entre la plataforma SCADA y los dispositivos de campo. Sin embargo, los resultados obtenidos corresponden a observaciones de carácter operativo y no a una caracterización cuantitativa del desempeño del sistema.

En particular, no se realizó la medición de latencia ni la cuantificación del error en la transmisión de datos mediante un protocolo experimental controlado. La estimación de estas métricas requiere la implementación de mecanismos de registro temporal automatizado, así como la repetición sistemática de eventos bajo condiciones definidas, lo cual no formó parte del diseño experimental desarrollado.

Desde el punto de vista de las comunicaciones industriales, el protocolo Modbus RTU opera mediante transacciones de tipo solicitud–respuesta, cuya duración depende de la longitud de la trama, la velocidad de transmisión en baudios, los tiempos de espera entre tramas y el procesamiento interno del dispositivo esclavo. En este contexto, el tiempo observado durante las pruebas corresponde al comportamiento global del sistema, el cual incluye la transmisión sobre el medio físico, el procesamiento del dispositivo y el ciclo de sondeo del SCADA, por lo que no puede atribuirse de manera directa a la latencia de un componente específico.

Adicionalmente, la arquitectura de RapidSCADA establece una separación funcional entre adquisición (Communicator), procesamiento (Server) y presentación (Webstation), lo que implica que la actualización de una variable es el resultado de múltiples etapas del sistema. En ausencia de instrumentación temporal adecuada, no es posible descomponer este comportamiento ni estimar la latencia asociada exclusivamente al software SCADA.

En relación con el error en la transmisión de datos, es necesario precisar que, al tratarse de un sistema de comunicación digital, la información intercambiada se encuentra codificada en formato binario y protegida mediante mecanismos de detección de errores, como el CRC en las tramas Modbus. En consecuencia, la evaluación del error de transmisión se asocia a la integridad de las tramas y no a la desviación de la magnitud física

medida. Bajo las condiciones de prueba implementadas, no se evidenciaron fallos de comunicación en los registros disponibles; sin embargo, no se realizó una cuantificación de estos eventos mediante métricas específicas.

En este sentido, los resultados obtenidos deben interpretarse como una validación funcional del sistema, en la cual se verifica la correcta operación del proceso de adquisición y control. Bajo estas condiciones, el sistema demostró un comportamiento consistente en la comunicación y supervisión de variables, sin que ello implique una caracterización cuantitativa de su desempeño temporal ni de la calidad de la transmisión de datos.

6. Conclusiones

En el desarrollo de este trabajo de grado se logró la instalación y puesta en funcionamiento de la plataforma *RapidSCADA versión 6.4.3* sobre un sistema operativo *Windows 11 Home Single Language (25H2)*, verificando la ejecución correcta de los servicios principales del sistema (***Server, Communicator y Webstation***). La instalación se realizó utilizando el instalador oficial para *Windows* y manteniendo la ruta de instalación por defecto, sin presentarse fallos asociados al sistema operativo, dependencias de *software* ni conflictos con los servicios del entorno.

Se implementó una arquitectura *SCADA* básica que permitió la comunicación con tres dispositivos físicos, correspondientes a dos módulos industriales *ADAM (4018+ y 4024)* de *Advantech* y un microcontrolador *Arduino Nano*, integrados en proyectos independientes dentro de *RapidSCADA*. La comunicación se realizó empleando los protocolos *Modbus TCP, Modbus RTU y MQTT*, sobre diferentes capas físicas, incluyendo *Ethernet*, *RS-485* mediante convertidor *USB–RS-485*, y *USB* directo, lo que permitió evaluar escenarios de supervisión con tecnologías heterogéneas dentro de un mismo entorno *SCADA*.

En el caso de *Modbus TCP* y *MQTT*, se validó la adquisición de variables analógicas y digitales simuladas mediante el uso de herramientas de simulación y publicación de datos, confirmando la correcta configuración de líneas de comunicación, dispositivos, plantillas y canales dentro de *RapidSCADA*. En particular, *Modbus TCP* permitió trabajar bajo un esquema de consulta maestro–esclavo, mientras que *MQTT* se integró mediante el modelo de publicación–suscripción, sin que esta diferencia afectara el objetivo formativo del sistema. En ambos casos, se verificó el flujo completo de adquisición, procesamiento y visualización de datos simulados antes de la integración con *hardware* físico, asegurando la coherencia de la configuración base del sistema.

La integración física mediante *Modbus RTU* sobre bus *RS-485* permitió comprobar el comportamiento del protocolo en topología multipunto. Se verificó la lectura de señales analógicas provenientes de termopares tipo K a través del módulo *ADAM-4018+,* así como la escritura de consignas analógicas de 0–20 mA hacia el módulo *ADAM-4024* desde la interfaz ***Webstation***, tanto en vistas tipo tabla como en vistas esquemáticas. Durante este proceso se identificaron inconsistencias iniciales en la interpretación de las consignas, las cuales se resolvieron mediante el ajuste del tipo de canal y la aplicación de fórmulas de entrada y salida en RapidSCADA.

Las pruebas realizadas evidenciaron que los valores de corriente obtenidos no correspondieron de forma exacta a los valores nominales configurados (por ejemplo, 10 mA o 20 mA, presentándose valores aproximados como 9.98 mA o 19.9 mA). Esta diferencia se asocia a la naturaleza del proceso de conversión digital-analógica y a las características propias del módulo, dentro del contexto de validación funcional del sistema implementado.

De manera complementaria, se integró un *Arduino Nano* como nodo *Modbus RTU* independiente, conectado directamente por *USB* y sin compartir el bus *RS-485* con los módulos industriales. En este dispositivo se cargó un programa que implementa el protocolo *Modbus RTU* y publica señales periódicas en registros *holding*. Esta integración permitió contrastar el comportamiento de un esclavo *Modbus* implementado por *software* con el de

módulos industriales, evidenciando que, bajo configuraciones adecuadas, es posible supervisar dispositivos de bajo costo dentro de un entorno *SCADA* real.

Durante el desarrollo del proyecto se identificaron y resolvieron problemas asociados a la configuración de puertos de comunicación, duplicación de direcciones *Modbus*, asignación incorrecta de líneas de comunicación, ocupación de puertos *COM* y configuración del sistema operativo, los cuales provocaron fallos de comunicación y requirieron reinicio de servicios o ajustes en la configuración. El análisis de estos errores permitió comprender la importancia de la coherencia entre la configuración física y lógica del sistema *SCADA*.

En conjunto, el trabajo permitió cumplir los objetivos planteados dentro del alcance definido para el proyecto. A partir de la experiencia obtenida, se evidencio que *RapidSCADA* puede utilizarse como una alternativa viable en entornos académicos, especialmente para actividades de formación que requieren la supervisión de variables simuladas y reales sin depender de licencias comerciales.

Las pruebas realizadas mostraron que, bajo configuraciones adecuadas y escenarios controlados, la plataforma permite integrar distintos protocolos de comunicación industrial y trabajar con dispositivos físicos y simulados dentro de un mismo entorno *SCADA*. Estas características, evaluadas en el contexto del laboratorio y con fines formativos, permiten considerar a *RapidSCADA* como una herramienta adecuada para apoyar la enseñanza práctica de conceptos fundamentales de supervisión y automatización industrial.

Es importante señalar que el alcance del presente trabajo se centró en la validación funcional del sistema, orientada a verificar la correcta integración entre la plataforma SCADA y los dispositivos de campo. En este sentido, no se abordó la caracterización cuantitativa del desempeño del sistema en términos de latencia ni de error en la transmisión de datos. La evaluación de estas métricas requiere un diseño experimental específico, basado en la instrumentación temporal automatizada y en la repetición controlada de eventos, aspectos que se plantean como líneas de trabajo futuro.

6.1 Recomendaciones y trabajos futuros

A partir de la validación funcional desarrollada y de las limitaciones metodológicas identificadas en la sección 5.7.1, se plantean las siguientes líneas de trabajo orientadas a ampliar el alcance del sistema implementado y profundizar su análisis en escenarios futuros.

En relación con la evaluación del desempeño, se propone el desarrollo de un diseño experimental orientado a la medición del comportamiento temporal del sistema y a la evaluación de la integridad de la comunicación. Este tipo de análisis requiere la implementación de mecanismos de registro temporal automatizado, la repetición controlada de eventos y la variación de parámetros de configuración como el período de sondeo y la velocidad de comunicación. Bajo estas condiciones, sería posible analizar el comportamiento del sistema dentro de un esquema de comunicación basado en transacciones solicitud-respuesta, como se define en la especificación del protocolo Modbus.

Adicionalmente, se considera pertinente evaluar el sistema en arquitecturas distribuidas, donde los módulos de adquisición, procesamiento y supervisión operen en diferentes nodos. Esta configuración permitiría analizar el comportamiento del sistema bajo condiciones de red más cercanas a entornos reales, incluyendo aspectos asociados a la comunicación entre componentes, la estabilidad operativa y la actualización de variables en escenarios no centralizados.

Otra línea de trabajo corresponde a la ampliación del banco de pruebas mediante la incorporación de un mayor número de dispositivos y variables de proceso, así como la integración de protocolos adicionales soportados por la plataforma *RapidSCADA*. La inclusión de múltiples nodos esclavos permitiría analizar el comportamiento del sistema ante mayores cargas de comunicación y evaluar su capacidad de escalabilidad dentro del enfoque de bajo costo planteado en el proyecto.

Asimismo, se propone explorar la integración del sistema con arquitecturas orientadas a entornos distribuidos mediante el uso de protocolos de mensajería como *MQTT* y la incorporación de un *broker* externo. Esta integración permitiría complementar el modelo de comunicación implementado y analizar su comportamiento en esquemas de publicación-suscripción, ampliando las posibilidades de interoperabilidad con otras plataformas.

En cuanto a la arquitectura del sistema, se sugiere analizar configuraciones que incorporen mecanismos de control de acceso y gestión de conexiones, tales como la definición de usuarios y perfiles, así como el uso de canales de comunicación cifrados en la interfaz de supervisión. Estas configuraciones permitirían evaluar el comportamiento del sistema en escenarios donde el acceso requiere ser gestionado de manera controlada.

Finalmente, se plantea el desarrollo de un entorno de laboratorio con fines académicos, estructurado mediante guías de práctica, ejercicios de configuración y criterios de evaluación. Este enfoque permitiría utilizar el sistema implementado como herramienta de apoyo en procesos formativos, facilitando la comprensión de conceptos asociados a sistemas *SCADA*, comunicaciones industriales y supervisión de procesos.

En conjunto, estas líneas de trabajo permiten extender el alcance del sistema desarrollado, manteniendo coherencia con la validación funcional realizada y estableciendo una base para futuras evaluaciones bajo condiciones más controladas y escenarios de mayor complejidad.

Referencias

[1] A. W. Colombo, S. Karnouskos, O. Kaynak, Y. Shi y S. Yin, «Industrial Cyberphysical Systems: A Backbone of the Fourth Industrial Revolution,» *IEEE Industrial Electronics Magazine,* vol. 11, nº 1, pp. 6-16, March 2017.

[2] «Introduction,» [En línea]. Available: https://rapidscada.net/docs/en/latest/software-overview/introduction. [Último acceso: 2025].

[3] S. O. E. V. D. F. A. a. J. G. L. I. Minchala, «An open source SCADA system to implement advanced computer integrated manufacturing,» *IEEE Latin America Transactions,* vol. 14, nº 12, pp. 4657-4662, 2016.

[4] A. Mejia Rojas y G. Barbieri, «A Low-Cost and Scaled Automation System for Education in Industrial Automation,» de *24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Zaragoza, 2019.

[5] G. Clarke, D. Reynders y E. Wright, Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems, ELSEVIER.

[6] T. S. a. J. J. M. Wollschlaeger, «The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0,» *IEEE Industrial Electronics Magazine,* vol. 11, nº 1, pp. 17-27, 2017.

[7] «What is Rapid SCADA,» [En línea]. Available: https://rapidscada.org/. [Último acceso: 2025].

[8] «Applications,» [En línea]. Available: https://rapidscada.net/docs/en/latest/software-overview/applications.

[9] «Documentation,» [En línea]. Available: https://rapidscada.org/product/documentation/. [Último acceso: 2025].

[10] «Software Architecture,» [En línea]. Available: https://rapidscada.net/docs/en/latest/software-overview/architecture.

[11] «Modulo Rapid Gate,» [En línea]. Available: https://rapidscada.net/docs/es/5.8/modules/mod-rapid-gate.

[12] «Foro de Rapid SCADA,» [En línea]. Available: https://forum.rapidscada.org/. [Último acceso: 2025].

[13] M. Castrillon Varela y S. Salazar Marulanda , Sistema SCADA basado en un ambiente de programación open source, Envigado: Universidad EIA, 2018.

[14] J. E. Duque Pardo, E. D. Villa Perez y R. E. Herrera Puello, Sistema SCADA en Web para control de nivel en tanques, Cartagena: Universidad Tecnologica de Bolivar, 2011.

[15] E. Boada Robayo, Análisis y diseño de un sistema SCADA del sistema de microrred no convencional del edificio Julio Mario Santo Domingo de la Universidad de los Andes, Enero, 2024.

[16] W. I. Carangui Rodriguez, SISTEMA SCADA DE BAJO COSTO PARA LA, Azogues : Universidad Católica de Cuenca, 2020.

[17] D. J. Cerezo Quina, T. Felix Santos, A. A. Echaiz, J. A. Butron Llamoca, A. Ortiz Salazar y E. R. Llanos Villarreal, «Implementación de un sistema Scada Freeware para una microrred de tensión continua/alterna,» October 2024.

[18] T. D. C. Rocha Devesa de Miranda, USO DO SCADABR PARA DESENVOLVIMENTO DE SISTEMAS SUPERVISÓRIOS DE PROCESSOS QUÍMICOS SIMULADOS, Niterói: Universidad Federal Fluminense, 2021.

[19] K. G. Carrillo Baquerizo y León Arias, Carlos Antonio, Diseño e implementación de un sistema SCADA basado en el software libre Node-red para el monitoreo y operación de la planta didáctica MPS PA. Compact Workstation de Festo, Guayaquil: Universidad Politecnica Salesiana, 2021.

[20] Mohammad E. Alim , Thomas H. Morris y Shelton R. Wright , «A Laboratory-Scale Spillway SCADA System Testbed for Cybersecurity Research,» University of Alabama in Huntsville, Alabama, 2021.

[21] A. Alcayde, I. Robalo, F. Montoya y F. Manzano, «SCADA System for Online Electrical Engineering Education,» *Inventions,* vol. 7, 2 December 2022.

[22] C. Vargas Salgado, J. Aguila Leon, C. Chiñas Palacios y D. A. Solar, «Sistema de Control Supervisor y Adquisición de Datos aplicado a una microrred con fines investigativos basada en Energías Renovables,» de *Conferencia Internacional sobre Innovación, Documentación y Educación*, Valencia, 2021.

[23] T. Cruz y P. Simoes, «Down the Rabbit Hole: Fostering Active Learning through Guided Exploration of a SCADA Cyber Range,» *Applied Sciences,* vol. 11, nº 20, 2021.

[24] K. Tawiah Ndede, A LOW-COST MICROCONTROLLER-BASED PLC FOR INDUSTRIAL AUTOMATION OF A GOLD PROCESSING PLANT, ASHESI UNIVERSITY, 2022.

[25] «Configuration Database,» [En línea]. Available: https://rapidscada.net/docs/en/latest/configuration/database/.

[26] «Scripts and Formulas,» [En línea]. Available: https://rapidscada.net/docs/en/latest/configuration/scripts.

[27] «Configuración de la comunicación con los dispositivos,» [En línea]. Available: https://rapidscada.net/docs/es/5.8/software-configuration/communication-with-devices.

[28] «Controlador esclavo Modbus,» [En línea]. Available: https://rapidscada.net/docs/en/5.8/use-cases/modbus-protocol.

[29] «Conexión de dispositivos mediante el estándar OPC,» [En línea]. Available: https://rapidscada.net/docs/en/5.8/use-cases/opc-standard.

[30] «Device Polling,» [En línea]. Available: https://rapidscada.net/docs/en/latest/configuration/device-polling/.

[31] «User Management,» [En línea]. Available: https://rapidscada.net/docs/en/latest/configuration/user-management.

[32] «Safety Recommendations,» [En línea]. Available: https://rapidscada.net/docs/en/latest/installation/safety.

[33] «Channels,» [En línea]. Available: https://rapidscada.net/docs/en/latest/configuration/channels.

[34] «Database Import Driver,» [En línea]. Available: https://rapidscada.net/docs/en/latest/modules/drv-db-import.

ANEXOS

Anexo A

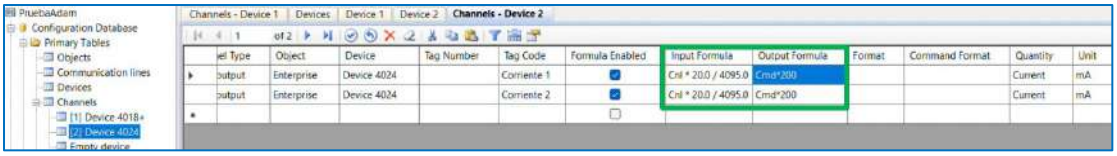


Figura A.1 Configuración de fórmulas de entrada y salida en los canales del módulo ADAM-4024 para la conversión de consignas lógicas a señales analógicas de corriente (0–20 mA) en RapidSCADA.
Fuente: Elaboración propia

Anexo B

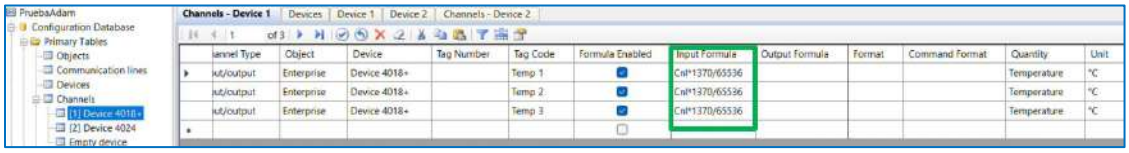


Figura B.1 Configuración de fórmulas de entrada en los canales del módulo ADAM-4018+ para la conversión de valores crudos a magnitudes de temperatura en RapidSCADA.
Fuente: Elaboración propia

Anexo C

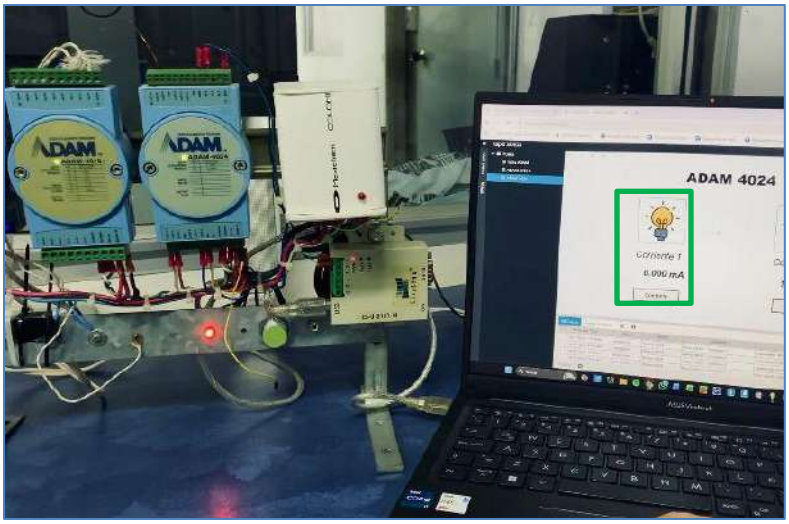


Figura C.1 Respuesta del sistema ante consigna de 0 mA en el módulo ADAM-4024, visualizada en la interfaz Webstation.
Fuente: Elaboración propia

Anexo D

Figura D.1 Respuesta del sistema ante consigna de 5 mA en el módulo ADAM-4024, visualizada en la interfaz Webstation.
Fuente: Elaboración propia

Anexo E

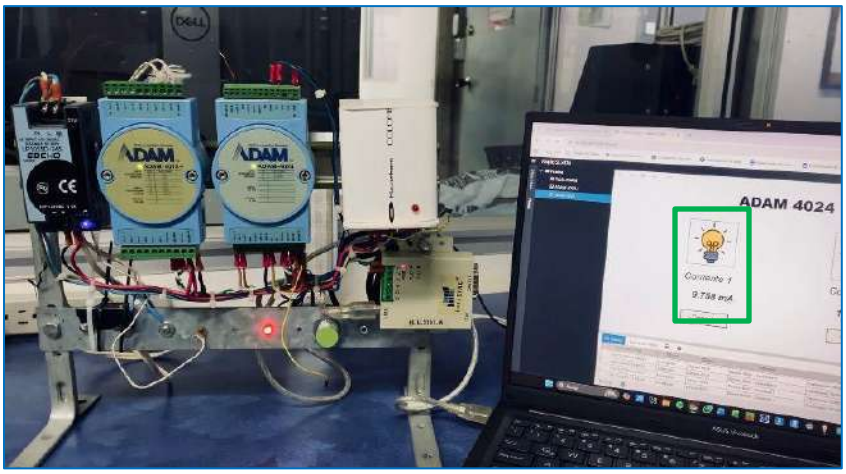


Figura E.1Respuesta del sistema ante consigna de 10 mA en el módulo ADAM-4024, visualizada en la interfaz Webstation.
Fuente: Elaboración propia

Anexo F

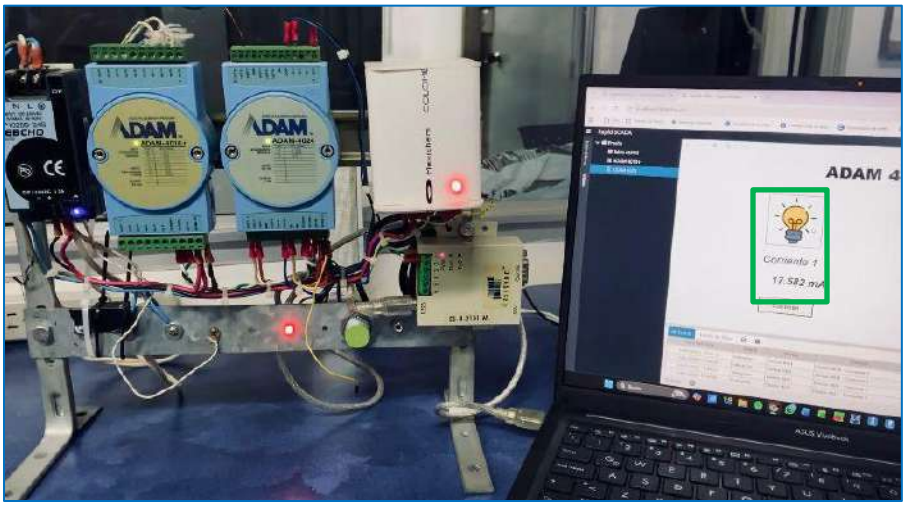


Figura F.1 Respuesta del sistema ante consigna de 18 mA en el módulo ADAM-4024, visualizada en la interfaz Webstation.
Fuente: Elaboración propia

Anexo G

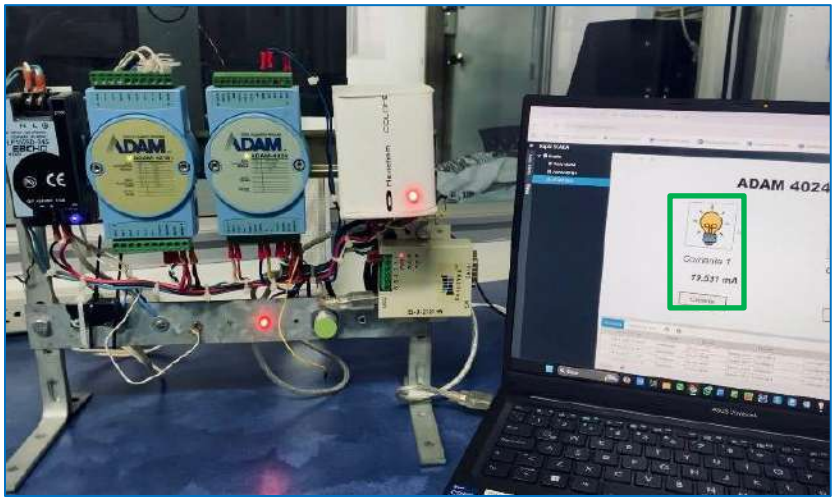


Figura G.1 Respuesta del sistema ante consigna de 20 mA en el módulo ADAM-4024, visualizada en la interfaz Webstation.
Fuente: Elaboración propia

Anexo H

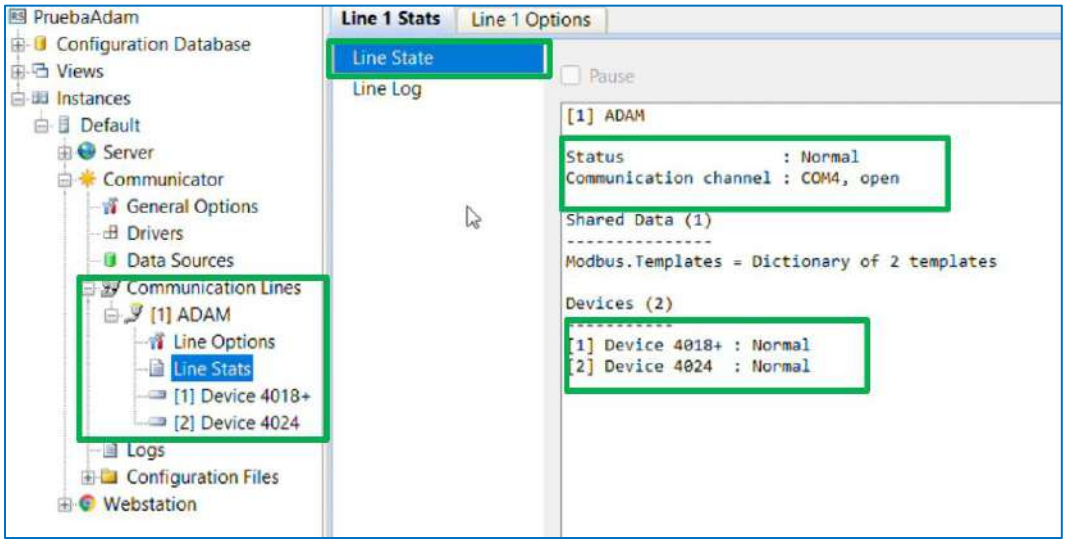


Figura H.1 Estado operativo de la línea Modbus RTU en RapidSCADA, mostrando la apertura del puerto COM4 y el reconocimiento simultáneo de los dispositivos ADAM-4018+ y ADAM-4024 con estado de comunicación normal.

Fuente: Elaboración propia.

Anexo I

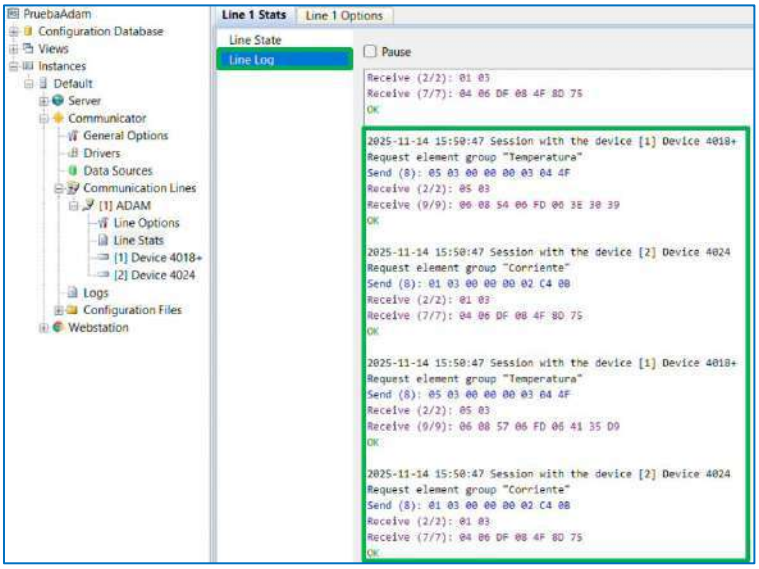


Figura I.1 Registro de intercambio de tramas Modbus RTU en RapidSCADA, evidenciando solicitudes y respuestas correctas de los módulos ADAM-4018+ y ADAM-4024 durante la adquisición y escritura de datos.

Fuente: Elaboración propia