

**UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
SEDE CONCEPCIÓN – REY BALDUINO DE BÉLGICA.**

**Diseño e implementación de un sistema destinado
al monitoreo sobre variables críticas en procesos
de calefacción y ventilación.**

Trabajo de Titulación para optar al Título
Profesional de Ingeniero de Ejecución en
Control e Instrumentación Industrial

Alumno:
Ignacio Rodolfo Castillo Hernández

Profesor Guía:
Nelson Vásquez Concha



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado
Título del trabajo: Diseño e implementación de un sistema destinado al monitoreo sobre variables críticas en procesos de calefacción y ventilación
Nombre del candidato(a): Ignacio Rodolfo Castillo Hernández
Carrera / Grado: Ing. Ejecución en control e instrumentación industrial
Campus: Concepción Departamento: Electrónica e Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Nelson Vásquez Concha, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO** contiene información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por (marcar una opción):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 25-03-2026 Firma: [Firma]

Estudiante o Candidato(a):

Fecha: 25/03/2026 Firma: [Firma]

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Índice

Introducción	8
Problemática	9
Objetivo General	10
Objetivos Específicos	11
Alcances del Proyecto	12
Restricciones	13
1. Marco Teórico	14
1.1. Protocolos de comunicación	14
1.1.1. Modbus	14
1.1.2. MQTT	16
1.2. Hardware de adquisición	18
1.2.1. Sensores	18
1.2.2. Resistencia shunt	18
1.2.3. ESP32-DevkitC-V4 (núcleo DOWDQ6)	19
1.2.4. Variador de frecuencia Mitsubishi FR-800	21
1.2.5. Modulo MAX485	22
1.2.6. Multiplexor CD74HC4067	23
1.3. Gestión de Energía y Protección	25
1.3.1. Sistemas de Respaldo Energético (UPS)	25
1.3.2. Regulación de Voltaje DC-DC (Convertidores Step-Down)	25
1.4. Softwares	26
1.4.1. HiveMQ Cloud: bróker online	26
1.4.2. Node-RED	27
1.4.3. InfluxDB : Base de Datos de Series Temporales (TSDB)	28
1.4.4. Grafana: Plataforma visual	29
2. Ingeniería de hardware y acondicionamiento de señales	31
2.1. La Rama Analógica: Del Lazo de Corriente al Voltaje (Acondicionamiento)	31
2.1.1. Validación y calibración del prototipo	31
2.2. La Escalabilidad: Multiplexación de Entradas	32
2.3. La Rama Digital: Comunicación con el VDF (Modbus RTU)	32
2.4. Gestión de Energía: La Fuente Única y el Step-Down	33
3. Implementación de la infraestructura de software y telemetría	34
3.1. Implementación del broker MQTT en la nube (HiveMQ)	34
3.1.1. Despliegue del Cluster y Configuración de Red	34
3.1.2. Gestión de Identidad y Accesos (Credentials)	34
3.1.3. Definición de la Jerarquía de Tópicos (Topic Namespace)	35

3.1.4.	Pruebas de Conectividad (Web Client)	36
3.2.	Procesamiento de datos Node-RED	37
3.2.1.	Palette Manager	37
3.2.2.	Configuración de nodo MQTT	38
3.2.3.	Lógica de Demultiplexación y Segregación de Variables (Function Node)	39
3.2.4.	Sistema de Detección y Gestión de Alertas Críticas	40
3.2.5.	Sistema de Notificación (Telegram Bot y Notify Node)	42
3.3.	Gestión de Base de Datos de Series de Tiempo (InfluxDB)	43
3.3.1.	Configuración de la Organización y Buckets	43
3.3.2.	Generación de Token de Seguridad (API Token)	44
3.3.3.	Integración de Nodos de Salida (InfluxDB Out)	44
4.	Implementación de la Plataforma de Visualización y Supervisión (Grafana)	45
4.1.	Creación de Instancia (Free Tier)	45
4.2.	Vinculación de la Fuente de Datos (Data Source)	45
4.2.1.	Configuración del Conector Flux	45
4.3.	Diseño y Construcción de Dashboards	46
4.3.1.	Implementación de Umbrales Visuales (Thresholds)	47
4.4.	Consultas Flux (Query Scripting)	48
4.4.1.	Código para Indicadores de Tiempo Real (Gauge y Stat)	48
4.4.2.	Código para Gráficas Históricas (Time Series)	49
4.5.	Análisis de Resultados y Validación de la Integridad de Datos	50
4.5.1.	Verificación de Concordancia (Input vs Output)	50
4.5.2.	Análisis del Comportamiento ante Pérdida de Señal (Estado “No Data”)	51
4.5.3.	Validación de la Lógica de Umbrales (Thresholds)	51
5.	CONCLUSIONES Y TRABAJOS FUTUROS	53
5.1.	Conclusiones	53
5.2.	Análisis de Variables Críticas y Diagnóstico Operativo	53
5.2.1.	Temperatura de Agua	53
5.2.2.	Presión de Impulsión (Variable Hidráulica)	53
5.2.3.	Corriente Eléctrica / Amperaje (Variable Eléctrica)	54
5.2.4.	Voltaje de Alimentación (Variable de Red)	54
5.3.	Trabajos Futuros	54
5.3.1.	Implementación en Empresa de Mantenimiento (Escalabilidad)	54
5.3.2.	Encapsulado y Grado de Protección Industrial (Clasificación IP)	55
5.3.3.	Implementación de Control Bidireccional (Telecontrol)	55
5.3.4.	Algoritmos de Mantenimiento Predictivo	55
6.	Anexos	56
6.1.	Anexo A: ESPECIFICACIONES TÉCNICAS (DATASHEETS)	56
6.1.1.	Microcontrolador ESP32 (Espressif Systems)	56
6.1.2.	Variador de Frecuencia Mitsubishi Electric Serie FR-F800	57
6.1.3.	Módulo Conversor TTL a RS-485 (MAX485)	58

6.2.	Anexo B: PLANOS Y ESQUEMAS DE CONEXIÓN	59
6.2.1.	B.1. Diagrama de Conexión Prototipo (Maqueta)	59
6.2.2.	B.2. Esquema de Conexión Industrial Teórico (Modbus)	59
6.3.	Bibliografía	60

Índice de figuras

1.1.	Arquitectura Maestro-Esclavo Modbus RTU.	14
1.2.	Arquitectura de publicación/suscripción MQTT.	17
1.3.	Pinout del ESP32-DevkitC-V4.	19
1.4.	Variador de Frecuencia Mitsubishi F800.	21
1.5.	Módulo MAX485.	23
1.6.	Módulo Multiplexor CD74HC4067.	24
1.7.	Convertidor DC-DC Step-Down LM2596.	25
1.8.	Interfaz de Node-RED.	27
1.9.	Ejemplo de Dashboard en Grafana.	30
2.1.	Parámetros Modbus.	32
2.2.	Diagrama de flujo de la arquitectura de hardware propuesta.	33
3.1.	Despliegue del Cluster y Configuración de red en HiveMQ.	34
3.2.	Gestión de credenciales y permisos en HiveMQ.	35
3.3.	Estructura del mensaje JSON enviado por el ESP32.	36
3.4.	Interfaz Web Client para envío y suscripción de pruebas.	36
3.5.	Recepción exitosa del mensaje de prueba en la consola.	37
3.6.	Instalación de nodos adicionales en Palette Manager.	37
3.7.	Configuración de conexión al servidor MQTT HiveMQ.	38
3.8.	Configuración del tópico y formato de salida JSON.	38
3.9.	Código JavaScript para la segregación de variables por sala.	39
3.10.	Configuración de umbrales en el nodo Switch.	40
3.11.	Formateo del mensaje de alerta usando plantillas Mustache.	41
3.12.	Configuración del límite de tasa para evitar spam de alertas.	42
3.13.	Flujo completo de procesamiento en Node-RED.	43
3.14.	Explorador de datos en InfluxDB mostrando las mediciones ingresadas.	44
4.1.	Portal de administración de Grafana Cloud.	45
4.2.	Configuración de los parámetros de conexión a InfluxDB.	46
4.3.	Selección y prueba del Data Source.	46
4.4.	Script Flux para obtener el último valor registrado.	48
4.5.	Indicador visual de pérdida de comunicación (No Data).	49
4.6.	Script Flux con agregación dinámica para series de tiempo.	49
4.7.	Validación de lectura correcta de presión (8.20 bar).	50
4.8.	Validación de cambio de color por superación de umbral.	51
4.9.	Estado del Dashboard ante pérdida de comunicación (No Data).	52
4.10.	Estado operativo del Dashboard con visualización en tiempo real.	52
6.1.	Portada Datasheet ESP32 Series.	56
6.2.	Portada Manual de Instrucciones Inverter FR-F800.	57
6.3.	Portada Datasheet Maxim Integrated MAX485.	58
6.4.	Conexionado, maqueta prototipo.	59
6.5.	Diagrama de flujo de datos y conexión del sistema.	59

Índice de cuadros

1.1.	Estructura de Trama Modbus RTU.	15
1.2.	Tabla comparativa entre el ESP8266 y el ESP32.	20
3.1.	Variables críticas, rangos de medición y condiciones de alarma.	41

Introducción

Actualmente, los sistemas de calefacción y ventilación son de gran importancia para garantizar el confort térmico y la calidad del aire en el ambiente, ya sea en instalaciones industriales, edificios comerciales o domiciliarios. Sin embargo, la falta de monitoreo continuo de variables críticas genera muchas dificultades e incertidumbres cuando se trata de detectar fallas en el proceso. Esto provoca retrasos en la atención de emergencias, altos costos de reparación o reemplazo de equipos, malos diagnósticos, sumado al tiempo de inactividad del proceso.

Las empresas que brindan apoyo mediante servicios de mantenimiento y emergencia acuden a las instalaciones para solucionar el problema, sin contar con la información previa necesaria sobre la condición del proceso y los equipos. Dificultando la respuesta rápida y eficiente, incrementando riesgos de daños y pérdidas económicas.

Por esta problemática, nace la necesidad de implementar un sistema de monitoreo remoto que permita supervisar en tiempo real las variables críticas del proceso “Calefacción y ventilación” adquiriendo información importante como el registro de datos históricos de cada variable y la implementación de alertas cuando ocurra alguna anomalía en el proceso. De esta manera, se acotan los tiempos para detectar fallas, se optimiza la toma de decisiones y se establece un mantenimiento preventivo, prolongando la vida útil de la maquinaria.

El presente proyecto tiene como finalidad diseñar e implementar un sistema donde los datos recolectados por los sensores de campo sean subidos a una nube y transmitidos en una página web, en la cual toda persona con el permiso otorgado pueda acceder con normalidad a esta información. En esta plataforma se podrá visualizar el estado en el que se encuentra el proceso, registros históricos y variables en tiempo real, además del estado de los equipos mecánicos según sus parámetros. Por último, se contará con notificaciones de emergencia o alerta temprana, con el fin de generar un impacto positivo en la gestión del proceso de calefacción y ventilación.

Problemática

En una empresa dedicada a la climatización, especializada en procesos de calefacción y ventilación, se han identificado deficiencias críticas en la gestión de emergencias operativas. Actualmente, la planta no cuenta con sensores ni sistemas de monitoreo en tiempo real, lo que imposibilita la adquisición de información oportuna y precisa sobre el estado de los procesos.

Ante una emergencia, el personal debe acudir al lugar sin conocer previamente la magnitud ni la ubicación exacta del problema, retrasando significativamente la capacidad de respuesta, comprometiendo la eficiencia operativa y derivando en daños mayores como reparaciones costosas o incluso reemplazar componentes claves para el sistema. La ausencia de información histórica y en tiempo real sobre las variables críticas del proceso y el estado de sus componentes limita la capacidad de prevención, reacción y predicción ante fallos del sistema.

Objetivo General

Diseñar e implementar un sistema de monitoreo remoto para el proceso de calefacción y ventilación, que permita visualizar en tiempo real variables de carácter crítico para el proceso y estado de los equipos, mediante una plataforma web con permisos y acceso seguro. En ella se podrán almacenar datos históricos y generar alertas automatizadas ante fallas, con el fin de optimizar los procesos de mantenimiento reactivo, preventivo y predictivo, reduciendo costos de pérdida en equipos importantes y entregando un mejor servicio.

Objetivos Específicos

- I. Diseñar la interfaz de adquisición de datos mediante ESP32 con señales analógicas y protocolo Modbus RTU.
- II. Configurar la transmisión de telemetría inalámbrica utilizando protocolo MQTT y HiveMQ.
- III. Implementar el procesamiento de datos en Node-RED y almacenamiento histórico en InfluxDB.
- IV. Desarrollar paneles de visualización web y alertas en plataforma Grafana.

Alcances del Proyecto

I. Diseño de arquitectura híbrida para instrumentación industrial.

Se contempla el diseño teórico y lógico para la integración simultánea de equipos de potencia, específicamente el Variador de Frecuencia Mitsubishi F-800 mediante protocolo Modbus RTU, y de instrumentación de campo (sensores de presión y temperatura) mediante la estandarización de señales de corriente 4-20 mA.

II. Implementación de prototipo funcional para validación de telemetría.

Construcción de un nodo de adquisición basado en ESP32 para validar el flujo de datos. Para la medición térmica, se integró un sensor RTD PT-100 acoplado a un transmisor industrial de 4-20 mA. Esto permitió validar de forma física la etapa de acondicionamiento de señal mediante resistencia shunt, la conversión analógica-digital, el escalamiento de variables y la estabilidad de la transmisión inalámbrica vía protocolo MQTT hacia la nube.

III. Centralización y visualización mediante stack de software.

El proyecto abarca la configuración integral del servidor de datos, integrando Node-RED para la gestión de mensajería, InfluxDB para el registro histórico eficiente y Grafana para la interfaz de usuario. El alcance de visualización incluye el diseño de tableros de control (Dashboards) interactivos y seguros para monitorear las tres áreas definidas: Sala de Bombas, Sala de Calderas y Sala de Ventilación.

IV. Gestión de alertas y seguridad de la información.

Se incluye la configuración de lógica de alarmas preventivas ante valores fuera de rango (simulación de fallas críticas) y la implementación teórica de un sistema de respaldo energético (UPS) para el servidor central. Esto garantiza la integridad de los registros históricos y la continuidad del monitoreo ante fallos del suministro eléctrico.

Restricciones

I. Disponibilidad de equipamiento industrial para pruebas destructivas.

El proyecto no cuenta con la disponibilidad de variadores de frecuencia Mitsubishi F-800 dedicados exclusivamente para experimentación. Debido a que los equipos existentes en las salas de bombas y calderas se encuentran en operación crítica continua, no es factible su intervención física o eléctrica en la etapa de potencia. En consecuencia, la validación de las variables eléctricas y de control de los variadores se restringe a lecturas controladas o emuladas, sin manipular voltajes industriales (380 V). Sin embargo, en el ámbito de la instrumentación térmica, se logró superar esta limitante mediante la implementación y calibración física de un lazo de medición de estándar industrial, compuesto por un sensor RTD PT-100 y un transmisor de 4-20 mA, lo que permite una validación fidedigna del acondicionamiento de la señal analógica.

II. Dependencia de infraestructura de comunicaciones existente.

La transmisión de telemetría desde el nodo IoT (ESP32) hacia el Broker en la nube está condicionada a la disponibilidad y estabilidad de la red de internet local (WiFi) del recinto. El alcance del proyecto no contempla el diseño, implementación y certificación de una infraestructura de red corporativa (cableado estructurado o enlaces de fibra óptica), limitándose a la configuración del cliente inalámbrico en el microcontrolador.

III. Validación mediante prototipo unitario.

Aunque la arquitectura del sistema está diseñada para ser escalable a múltiples edificios y salas de máquinas, el presente proyecto se restringe a la implementación y validación de un único nodo de monitoreo. Este nodo servirá como prueba de concepto para demostrar la viabilidad técnica del flujo completo de datos (Sensor-Bróker-Node RED- Base de Datos-Dashboard), representando la funcionalidad que posteriormente se replicará de manera industrial.

IV. Restricción presupuestaria y enfoque Costo-Beneficio. El desarrollo del prototipo está sujeto a un presupuesto acotado, lo que restringe la selección de hardware a tecnologías de código abierto y bajo costo (ESP32, sensores genéricos, software gratuitos) en lugar de PLCs industriales o licencias de software SCADA comerciales. Esta restricción válida, tiene el objetivo de demostrar la viabilidad económica de una solución, frente a las alternativas tradicionales de alto costo.

1. Marco Teórico

1.1. Protocolos de comunicación

1.1.1. Modbus

Es un protocolo de comunicación abierto que permite la transmisión de datos entre dispositivos electrónicos, como controladores lógicos programables (PLC), sensores, actuadores y otros equipos industriales. Este protocolo es utilizado para la supervisión y el control de equipos en entornos industriales. Su popularidad se debe a su simplicidad, confiabilidad y facilidad de implementación. Las variantes más conocidas son Modbus RTU y Modbus TCP/IP (Ethernet).

Modbus RTU. Modbus RTU (Remote Terminal Unit) es un protocolo de comunicación en serie, considerado un estándar de facto en la industria durante décadas debido a su robustez y simplicidad. Su propósito principal es definir un formato de mensaje y un conjunto de reglas para que los dispositivos intercambien información de control y datos de manera fiable. La característica distintiva de la variante “RTU” es su método de transmisión: utiliza una codificación binaria directa. Esto significa que cada byte de información (8 bits) se envía como un solo byte binario, lo que lo hace muy eficiente y rápido, optimizando la comunicación en las líneas serie de baja velocidad.

Arquitectura. Utiliza una arquitectura de Maestro-Eslavo; en cualquier red Modbus RTU solo puede haber un maestro, que es por lo general el controlador lógico programable (PLC). Él es el encargado de iniciar la comunicación, enviando consultas o peticiones a uno o más esclavos del sistema. Los esclavos (por lo general sensores, VDF) pueden ser múltiples dispositivos en la misma red; cada uno tiene su dirección única que va desde el 1 al 247 y tienen que permanecer siempre “escuchando”, esperando que el maestro les solicite alguna información que proporcionar. El flujo completo de esta arquitectura consta de que el maestro inicia la comunicación preguntando específicamente al esclavo N°7 de la red; en ese momento todos los esclavos reciben el mensaje, pero solo el N°7 lo procesa y prepara una respuesta para luego proporcionarla al maestro, siendo esta un parámetro de temperatura, nivel, presión, etc.

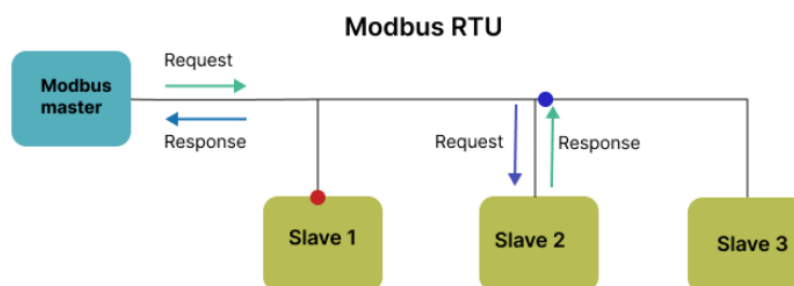


Figura 1.1: Arquitectura Maestro-Eslavo Modbus RTU.

La capa física RS-485. Modbus RTU no es un estándar de cableado, es un protocolo de software (un lenguaje) pero este necesita un estándar de hardware (capa física) para viajar. La transmisión de datos se realiza mediante interfaces serie como RS-232, RS-485 o RS-422, lo que garantiza la compatibilidad con las redes industriales. La más común es RS-485 que permite a la conexión en red “multi-drop” conectar múltiples dispositivos esclavos en un solo par de cables trenzados a largas distancias (de hasta 1.200 metros) y es altamente resistente al ruido eléctrico utilizando señalización diferencial.

Trama. La estructura de trama RTU constituye la base del protocolo de comunicación Modbus RTU, definiendo la organización y transmisión de datos Modbus entre dispositivos. Esta transmisión se habilita a través de la capa de transporte, que abarca los protocolos y técnicas utilizados para garantizar una transferencia de datos fiable y eficiente. La estructura de trama incluye los siguientes elementos:

Dirección (1 byte)	Función (1 byte)	Datos (N bytes)	CRC (2 bytes)
------------------------------	----------------------------	---------------------------	-------------------------

Tabla 1.1: Estructura de Trama Modbus RTU.

- **Dirección (1 byte):** Identifica al dispositivo esclavo al que va dirigido el mensaje (si es el maestro) o de qué esclavo proviene el mensaje (si es una respuesta).
- **Función (1 byte):** Indica el tipo de acción que solicita el maestro, cómo leer registros (Holding Registers) o escribir datos.
- **Datos (N bytes):** Parte variable del mensaje que contiene la información relevante para la función, incluidas las direcciones y valores de los registros.
- **CRC (2 bytes):** Cyclic Redundancy Check proporciona comprobación de errores mediante una comprobación de redundancia cíclica.

Detección de errores “CRC”. El CRC “Cyclic Redundancy” es la garantía de Modbus RTU y funciona de la siguiente manera: El emisor (maestro) calcula el CRC de 16 bits para el mensaje y lo adjunta al final de la trama (es un valor matemático complejo que se calcula en función de todos los bytes anteriores del mensaje). El receptor (esclavo) recibe la trama y realiza exactamente el mismo cálculo matemático con los datos que recibió (sin incluir el CRC). Luego compara el cálculo del CRC con el cálculo del “emisor”.

- Si coincide: El mensaje es válido y se procesa.
- Si no coincide: Significa que el mensaje se corrompió por el ruido del cable, simplemente se descarta el mensaje y se intenta nuevamente.

Los silencios del mensaje. Una regla estricta del protocolo establece que debe haber un período de silencio de al menos 3,5 “tiempos de carácter” entre tramas. Si se detecta este silencio, el dispositivo receptor sabe que el mensaje anterior ha terminado y que cualquier dato nuevo que llegue es el comienzo de una nueva trama (empezando con una dirección de esclavo).

1.1.2. MQTT

MQTT, que significa “Message Queuing Telemetry Transport” es un protocolo de mensajería optimizado que opera bajo el modelo de Publicación-Suscripción (Publish-Subscribe). Esta arquitectura es su característica más importante y la diferencia fundamentalmente de los protocolos de petición-respuesta como Modbus o HTTP. En un sistema de petición-respuesta, un cliente debe conocer la dirección de un servidor y solicitarle activamente datos, creando una conexión directa y dependiente, mientras que MQTT elimina esta dependencia. Los dispositivos, llamados “clientes”, no se comunican directamente entre sí. En su lugar, todos se conectan a un servidor central llamado bróker. Los dispositivos que envían datos (Publicadores) simplemente envían sus mensajes al bróker, y los dispositivos que reciben datos (Suscriptores) le indican al bróker qué mensajes desean recibir. El resultado es un sistema “desacoplado” donde el publicador no sabe quién (o cuántos) suscriptores recibirán sus datos, y los suscriptores no saben quién (o cuántos) publicadores están generando la información. Esta separación permite una flexibilidad y escalabilidad masiva, ya que se pueden agregar o quitar sensores o aplicaciones de monitoreo en cualquier momento sin necesidad de reconfigurar ningún otro componente del sistema.

Su principal ventaja es que es extremadamente ligero: el encabezado mínimo de un mensaje MQTT es de solo 2 bytes, lo que reduce drásticamente el consumo de ancho de banda y la potencia de procesamiento necesaria. Además, es un protocolo asíncrono; el publicador puede “disparar y olvidar” el mensaje sin esperar una respuesta, permitiéndole volver a un modo de bajo consumo (deep-sleep) de inmediato para ahorrar batería. Para manejar redes inestables, MQTT incorpora Niveles de Calidad de Servicio (QoS), que permiten al cliente definir la garantía de entrega: QoS 0 (entregar como máximo una vez, es rápido, pero se pueden perder mensajes), QoS 1 (al menos una vez, garantiza la entrega, pero puede duplicar mensajes) y QoS 2 (exactamente una vez, el más fiable pero más lento). Finalmente, MQTT maneja el estado de la conexión a través de la función “Última Voluntad y Testamento” (Last Will and Testament). Un cliente puede decirle al bróker, al conectarse, que si se desconecta abruptamente (por fallo de energía o de red), el bróker debe publicar automáticamente un mensaje de “error” o “desconectado” en un tema específico, permitiendo que el resto del sistema sepa al instante que ese dispositivo está fuera de línea.

Arquitectura de publicación/suscripción MQTT

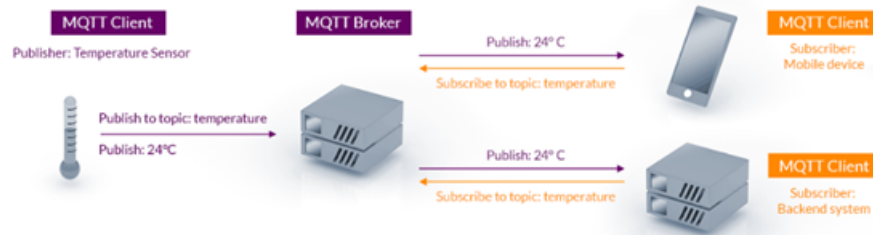


Figura 1.2: Arquitectura de publicación/suscripción MQTT.

Componentes claves.

- **Publisher (Publicador):** Es el dispositivo que envía la información. No sabe quién la va a recibir. Simplemente “publica” un mensaje sobre un tema específico.
- **Broker (Agente):** Es el servidor central que recibe todos los mensajes. Su trabajo es filtrar todos los mensajes y entregarlos a cualquiera que se haya “suscrito” al tema de ese mensaje.
- **Subscriber (Suscriptor):** Es quien quiere recibir la información de cierto tema específico del cual esté interesado.

Topics. La forma en que el bróker sabe a quién enviar cada mensaje es a través de un sistema de direccionamiento simple pero poderoso llamado “Topics”. Un tema es una cadena de texto, similar a una ruta de archivo, que utiliza barras diagonales para crear una jerarquía (por ejemplo, dep1/caldera/temperatura). Cuando un cliente quiere publicar un dato, adjunta ese dato (conocido como payload o carga útil) a un tema específico y lo envía al bróker. Por otro lado, un cliente se “suscribe” a los temas que le interesan. Puede suscribirse a un tema exacto o usar comodines (“#” que permite suscribirse a todos los tópicos disponibles). El bróker recibe cada mensaje, revisa su tema y lo envía instantáneamente a todos los clientes que tengan una suscripción que coincida.

1.2. Hardware de adquisición

1.2.1. Sensores

En el dominio de la ingeniería de control e instrumentación, un sensor es un dispositivo técnico, clasificado como un tipo de transductor, cuya función primaria es detectar, cuantificar y reaccionar a eventos o cambios en su entorno físico o químico. Actúa como el elemento de interfaz fundamental entre una magnitud física (la variable de proceso o PV) y un sistema de adquisición de datos o de control. El propósito del sensor es transformar la energía asociada a la variable de proceso. Como la energía térmica (temperatura), cinética (caudal, vibración), potencial (presión, nivel) o electromagnética (corriente), en una señal eléctrica análoga o digital. Esta señal de salida, denominada variable medida (MV), es cuantitativamente proporcional a la magnitud de la variable física de entrada, permitiendo así su interpretación, monitoreo o manipulación por un controlador lógico programable (PLC), un sistema SCA-DA o, un microcontrolador.

Acondicionamiento de Señal y el Transmisor. Debido a la naturaleza de la señal del elemento primario, está casi siempre debe pasar por una etapa de acondicionamiento de señal. Este proceso incluye varias operaciones críticas: filtrado (para eliminar el ruido), amplificación (para elevar el nivel de la señal), linealización (para corregir la no linealidad inherente del sensor) y compensación (para corregir desviaciones por factores ambientales, como la temperatura ambiente en una termocupla). En la instrumentación industrial moderna, el sensor y el circuito de acondicionamiento de señal están integrados en una sola carcasa. A este dispositivo completo se le denomina transmisor. Por lo tanto, un “transmisor de temperatura 4-20 mA” contiene el sensor (ej. un RTD), un microprocesador para linealización y compensación, y un bucle de corriente que “traduce” el rango de temperatura calibrado (ej. 0-100 °C) a la señal estándar de la industria (4-20 mA), asegurando una transmisión de datos robusta e inmune al ruido a largas distancias.

1.2.2. Resistencia shunt

Una resistencia shunt, también conocida como resistencia de derivación, es un componente pasivo de alta precisión y baja inductancia. Su característica principal es poseer un valor óhmico muy exacto y estable, especificado por una tolerancia ajustada (típicamente 1 % o inferior) y un bajo coeficiente de temperatura (TCR). Esto garantiza que su resistencia no varíe significativamente con los cambios de temperatura ambiente o por autocalentamiento. En la instrumentación, su función primordial es actuar como un transductor pasivo de corriente-a-voltaje (I-V). Al ser insertada en serie dentro de un circuito, la totalidad de la corriente que se desea medir fluye a través de ella. De acuerdo con la Ley de Ohm, este flujo de corriente (I) genera una caída de potencial (voltaje, V) diferencial entre sus terminales, la cual es directamente proporcional a la corriente que la atraviesa.

Aplicación: Para la adquisición de datos, se presenta una incompatibilidad de dominios: el dispositivo de adquisición (ESP32) opera midiendo voltaje (específicamente, una caída de tensión), mientras que el transmisor industrial entrega una señal de corriente en el rango de 4-20 mA. Para solucionar esta problemática de interfaz, se recurre a la Ley de Ohm. La solución consiste en instalar una resistencia de precisión (conocida como resistencia shunt) de 150 Ohm en el terminal de retorno del bucle de 4-20 mA. Esta resistencia convierte la corriente que fluye a través de ella en una caída de tensión directamente proporcional. Se seleccionó este valor de 150 Ohm específicamente porque escala la señal de 4-20 mA a un rango de voltaje que es seguro y medible para el ESP32, cuyo rango de lectura nativo en el ADC es de 0 V a 3,3 V.

La siguiente demostración, basada en la Ley de Ohm ilustra cómo se transforma la señal:

- Corriente mínima (4 mA): $V_{min} = 0,004 A \times 150 \text{ Ohm} = 0,6 V$
- Corriente máxima (20 mA): $V_{max} = 0,020 A \times 150 \text{ Ohm} = 3 V$

Por lo tanto, el uso de una resistencia shunt de precisión de 150 Ohm convierte, de forma lineal y fiable, la señal industrial de 4-20 mA en un rango de voltaje de 0,6 V a 3 V. Este rango es ideal, ya que se encuentra de forma segura dentro de los límites operativos del ADC del ESP32 (0 V a 3,3 V) proporcionando un margen de seguridad (headroom) y permitiendo una medición precisa sin riesgo de daño al microcontrolador.

1.2.3. ESP32-DevkitC-V4 (núcleo DOWDQ6)

El ESP32 es una familia de microcontroladores de bajo costo y alta integración, creados por la empresa china Espressif Systems. No es solo un microcontrolador, sino un “Sistema en un Chip” (SoC) completo, cuya característica más destacada es la inclusión de conectividad Wi-Fi y Bluetooth. Este sistema es capaz de conectar el hardware de campo (industria o terreno) con internet (nube), esto lo convierte en una solución extremadamente potente y rentable para proyectos, ya que puede conectarse a redes inalámbricas y a Internet sin necesidad de módulos o “shields” adicionales.

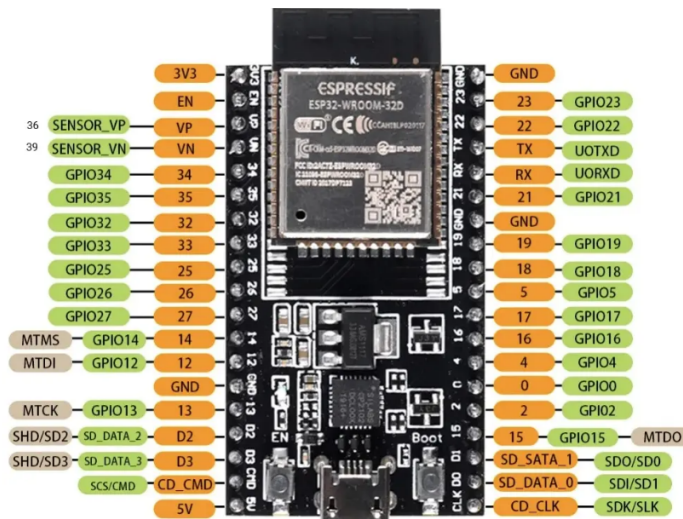


Figura 1.3: Pinout del ESP32-DevkitC-V4.

Alimentación. La placa de desarrollo ESP32 opera con lógica de 3,3 V pero admite alimentación de 5 V se conecta a través del pin “VIN” el cual viene con un regulador de voltaje AMS1117-3,3, disipando en forma de calor los 1,7 V restantes, no es un divisor de voltaje sino un circuito activo que está constantemente midiendo la salida.

Características principales.

- **Wi-Fi integrado:** Permite conectarse a una red local e internet para enviar datos a un bróker en la nube, sin la necesidad de módulos externos.
- **Pines UART (Serial):** Tiene tres puertos UART por hardware que facilita la comunicación con múltiples dispositivos en serie, como por ejemplo con la comunicación Modbus RTU.
- **Pines ADC:** múltiples pines que pueden utilizarse como ADCs de 12 bits, permitiendo leer voltajes analógicos de sensores en un rango de 0 a 4095, con un rango de voltaje ideal 1,8 V - 3,3 V.
- **Bajo costo:** Es extremadamente barato para la potencia que ofrece, lo que lo hace viable para proyectos académicos y para despliegues masivos en la industria.
- **Flexibilidad:** Es posible programarlo utilizando el entorno IDE de Arduino (en C++) o MicroPython, lo cual otorga acceso a múltiples librerías desarrolladas para Modbus, MQTT, Wi-Fi, etc. Con una entrada micro-USB para cargar el código (firmware).

Problemática. El problema principal se basa en que el periférico ADC2 (Conversor Analógico-Digital 2) del ESP32 comparte recursos de hardware con el controlador de WI-FI. Cuando el Wi-Fi está encendido (ya sea para escanear redes, conectarse o mantener una conexión), el controlador de Wi-Fi toma control total de esos recursos. Como resultado, el periférico ADC2 se vuelve completamente inutilizable para el código o programa.

- Los GPIO inutilizables son: GPIO (0, 2, 4, 12, 13, 14, 15, 25, 26 y 27).
- Los GPIO utilizables del ADC1 son: GPIO (32, 33, 34, 35, 36, 39).

Especificaciones	ESP8266	ESP32
MCU	Xtensa Single-Core 32-bit L106	Xtensa Dual-Core 32-bit LX6
Frecuencia Típica	80 MHz	160 MHz
SRAM	160 kB	512 kB
GPIO	17	36
ADC	10-bit	12-bit
Bluetooth	None	Bluetooth 4.2 and below

Tabla 1.2: Tabla comparativa entre el ESP8266 y el ESP32.

1.2.4. Variador de frecuencia Mitsubishi FR-800

Un variador de frecuencia (VFD), también conocido como convertidor de frecuencia o drive de velocidad variable, es un dispositivo de electrónica de potencia que permite el control preciso sobre la velocidad y el par de un motor eléctrico de corriente alterna (AC). Su principio de funcionamiento se basa en la rectificación de la señal de entrada de AC (tensión y frecuencia fijas de la red eléctrica) a corriente continua (DC), para luego, mediante un circuito inversor (basado en transistores IGBT), generar una nueva señal de AC con una tensión y frecuencia variables.

Dado que la velocidad de un motor asíncrono está directamente ligada a la frecuencia de la alimentación que recibe, el VFD permite un control de proceso que un motor de arranque directo no puede ofrecer. Más allá del simple control de velocidad, los VFD son un pilar fundamental en la eficiencia energética. En aplicaciones con cargas de par variable o cuadrático, como las bombas y los ventiladores, el consumo de energía es proporcional al cubo de la velocidad. Una leve reducción en la velocidad del motor (ej. 20 %) puede resultar en un ahorro de energía drástico (superior al 40-50 %).



Figura 1.4: Variador de Frecuencia Mitsubishi F800.

Segunda interfaz Serie (RS-485): Esta es la interfaz más relevante para la integración con el sistema ESP32. El diagrama de cableado de los bornes de señales con los terminales TXD+, TXD- (Datos enviados), RXD+, RXD- (Datos recibidos) y SG (Tierra de señal). Esta es una implementación física estándar de RS-485, que es la capa física sobre la que opera el protocolo Modbus RTU.

Parámetros de configuración de comunicación Modbus RTU: Para establecer la conexión Modbus RTU con el ESP32, se deben configurar los parámetros de la segunda interfaz serie

- **Pr. 331 (Número de estación):** Es la dirección de esclavo Modbus del variador (de 0 a 247).
- **Pr. 332 (Tasa de transferencia):** La velocidad de comunicación (ej. 9.600, 19.200 bps).
- **Pr. 334 (Comprobación de paridad):** Define la paridad (ej. None, Even, Odd).
- **Pr. 539 (Intervalo de tiempo de chequeo de datos Modbus RTU):** Define el timeout de comunicación.

Registro monitoreo de variables (Corriente, Voltaje y frecuencia:) Es importante distinguir entre “Parámetros” (configuración) y Registros de Monitoreo”(datos en tiempo real). Las variables de monitoreo (Corriente, Voltaje, Frecuencia) no son parámetros “Pr.” que se ajustan, sino valores operativos que se leen. Registros Modbus:

- **40201:** Frecuencia.
- **40202:** Corriente.
- **40203:** Voltaje.

1.2.5. Modulo MAX485

El MAX485 es un circuito integrado (IC) de bajo consumo, que funciona como un transceptor (transmisor + receptor). Su propósito técnico no es ser un protocolo, sino implementar la capa física (Capa 1 del modelo OSI) para el estándar de comunicación en serie EIA/TIA-485, comúnmente conocido como RS-485. Su función es salvar la incompatibilidad entre los voltajes lógicos (TTL/CMOS) del puerto UART del ESP32 y las señales eléctricas robustas requeridas por el bus industrial RS-485 para comunicarte con el variador F800.

Principio de funcionamiento. El estándar RS-485 utiliza señalización diferencial y el modulo MAX485 lo implementa mediante dos líneas de datos: Línea A (no inversora) y Línea B (inversora). La lógica binaria no se define por un voltaje absoluto, sino por la diferencia de potencial entre estas dos líneas ($V_a - V_b$). La principal ventaja de este método es la alta inmunidad al ruido de modo común. Si un motor industrial induce un pico de voltaje (ruido electromagnético), este se sumará por igual a ambas líneas A y B. Dado que el receptor solo lee la diferencia entre ellas, el ruido se anula eficazmente. Esto permite al RS-485 transmitir datos de forma fiable a largas distancias (hasta 1.200 metros) y en entornos eléctricos hostiles, como una sala de bombas.

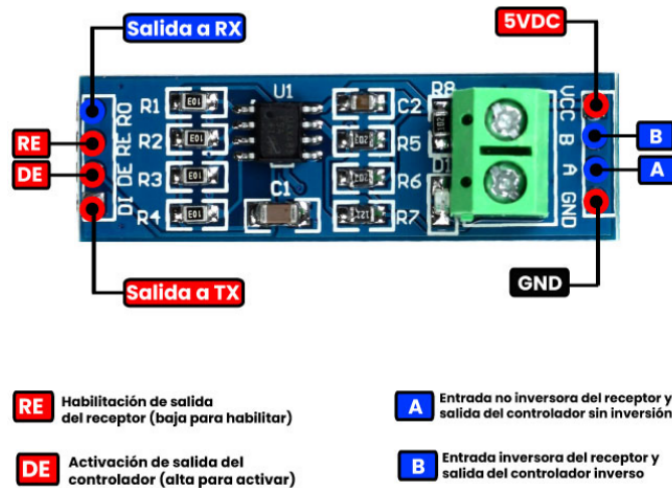


Figura 1.5: Módulo MAX485.

Topología de red y operación. El MAX485 está diseñado para redes “multi-drop”, donde un solo par de cables (el bus) puede conectar múltiples dispositivos (típicamente hasta 32 transceptores). Este módulo opera en modo half-duplex. Esto significa que los datos pueden fluir en ambas direcciones (enviar y recibir), pero no simultáneamente. El dispositivo solo puede transmitir o recibir en un instante determinado. Esta limitación introduce la necesidad de un control de dirección, que se gestiona mediante dos pines de habilitación.

- **DE (Driver Enable):** Un HIGH en este pin activa el transmisor (el “Driver”) para enviar datos desde el ESP32 al bus RS-485.
- **RE (Receiver Enable):** Un LOW en este pin activa el receptor (el Receiver”) para escuchar los datos del bus y enviarlos al ESP32.
- **Modo Transmisión:** El ESP32 pone este pin de control en HIGH. El transmisor se enciende y el receptor se apaga.
- **Modo Recepción:** El ESP32 pone este pin de control en LOW. El transmisor se apaga (entrando en alta impedancia) y el receptor se enciende.

1.2.6. Multiplexor CD74HC4067

El CD74HC4067 es un conmutador (switch) analógico de 16 a 1, controlado digitalmente. Pertenecce a la familia lógica CMOS de alta velocidad (la ‘HC’ en su nombre), lo que significa que es rápido, y lo más importante, es totalmente compatible con los niveles lógicos de 3,3 V del ESP32:

- 16 canales de entrada/salida (Y0-Y15): conexión de sensores.
- 1 pin común de entrada/salida: (Z) Aquí se conecta el pin del ADC (ESP32 GPIO 34).
- 4 pines de selección (S0, S1, S2, S3) entradas de control digital.
- 1 pin de habilitación: Un pin maestro para encender (LOW) o apagar (HIGH).

Solución a la problemática Puertos ADC en el ESP32. Cuando el Wi-Fi está activo, el hardware de la radio del ESP32 toma el control del periférico ADC2, dejándolo inutilizable. Esto te restringe a usar únicamente los pines del ADC1 (como GPIO 32, 33, 34, 35, 36, 39) dejando un déficit de pines al tener más sensores. El CD74HC4067 soluciona esta limitación permitiendo “compartir” un solo pin ADC entre 16 sensores. El costo de esta expansión implica el uso de 4 pines digitales (GPIO) del ESP32 para controlar el multiplexor; sin embargo, a cambio se obtienen 15 entradas analógicas adicionales

Arquitectura de conexión.

- **Pin Común (Z):** Se conecta a un único pin de ADC1 del ESP32 (GPIO 34). Este será el único pin que lea todos los valores analógicos.
- **Sensores (0-15):** Las salidas de voltaje de las resistencias shunt (0,6-3 V) se conectan a los 16 canales del multiplexor.
- **Control (S0-S3):** Los 4 pines de selección se conectan a 4 pines GPIO digitales del ESP32 (GPIO 18, 19, 21, 22).
- **Habilitación (EN):** El pin EN se conecta a GND (tierra) para mantener el chip permanentemente activado.



Figura 1.6: Módulo Multiplexor CD74HC4067.

Lógica de software. El proceso de operación es un bucle de sondeo, gestionado por el ESP32. Para leer un sensor específico, el ESP32 primero configura sus cuatro pines de control digital (conectados a los pines de selección S0 a S3 del CD74HC4067) con la dirección binaria de 4 bits (un número de 0 a 15) que corresponde al canal del sensor deseado. Instantáneamente, el multiplexor actúa como un conmutador electrónico, creando una conexión física de baja resistencia entre el pin de entrada del sensor seleccionado (ej. Y5) y el único pin de salida común (Z). Este pin Z está conectado permanentemente a un solo pin ADC1 del ESP32 (GPIO 34). Inmediatamente después de establecer esta conexión, el ESP32 ejecuta una única lectura analógica en ese pin ADC, capturando así el voltaje de ese sensor. Para leer el siguiente sensor, el ESP32 simplemente cambia la dirección binaria en los pines S0-S3 y repite la lectura en el mismo pin GPIO 34, obteniendo esta vez el valor del nuevo sensor seleccionado.

1.3. Gestión de Energía y Protección

1.3.1. Sistemas de Respaldo Energético (UPS)

La integridad de los datos en sistemas de monitoreo industrial depende estrictamente de la continuidad del suministro eléctrico. Una interrupción abrupta de energía en el servidor central no solo detiene la adquisición de datos, sino que puede causar la corrupción irreversible de los archivos de la base de datos (especialmente en bases de datos que mantienen índices en memoria RAM antes de escribir en disco). Para mitigar este riesgo, se emplean Sistemas de Alimentación Ininterrumpida (UPS). Estos dispositivos cumplen dos funciones críticas:

- **Respaldo de Energía:** Proporcionan energía temporal a través de baterías internas, permitiendo que el sistema siga operando durante cortes breves o ejecute una secuencia de apagado seguro (graceful shutdown) durante cortes prolongados.
- **Acondicionamiento de Energía:** Filtran perturbaciones eléctricas como picos de voltaje, ruido y fluctuaciones de frecuencia que podrían dañar el hardware sensible del servidor o del gateway.

Existen tres topologías principales de UPS, siendo la Línea Interactiva (Line-Interactive) el estándar para servidores de monitoreo no críticos, ya que incorpora un regulador automático de voltaje (AVR) para corregir fluctuaciones sin utilizar la batería, extendiendo su vida útil.

1.3.2. Regulación de Voltaje DC-DC (Convertidores Step-Down)

En entornos industriales, el estándar de alimentación para instrumentación y control es de 24 V. Sin embargo, los microcontroladores modernos como el ESP32 operan a voltajes lógicos de 5 V o 3,3 V. La adaptación de voltaje mediante reguladores lineales tradicionales (como la serie LM7805) es ineficiente en estas condiciones, ya que la gran diferencia de potencial ($24\text{ V} - 5\text{ V} = 19\text{ V}$) se disipa en forma de calor, lo que puede provocar sobrecalentamiento y fallos térmicos. Para solucionar esto, se utilizan convertidores de conmutación tipo Buck o Step-Down, como el LM2596. A diferencia de los lineales, estos dispositivos utilizan un transistor que conmuta a alta frecuencia (típicamente 150 kHz) y un inductor para almacenar y transferir energía a la salida con pérdidas mínimas. Esto permite eficiencias superiores al 80-90%, reduciendo drásticamente la generación de calor y garantizando una alimentación estable para la electrónica digital sensible dentro de un gabinete industrial.



Figura 1.7: Convertidor DC-DC Step-Down LM2596.

1.4. Softwares

1.4.1. HiveMQ Cloud: bróker online

HiveMQ es una plataforma de mensajería y un bróker MQTT de nivel empresarial, diseñado para la transferencia de datos rápida, fiable y masivamente escalable en el contexto del Internet de las Cosas (IoT). Su arquitectura está optimizada para manejar millones de conexiones de clientes simultáneas y un alto volumen de mensajes (alto rendimiento) con baja latencia, adhiriéndose plenamente a los estándares del protocolo MQTT. HiveMQ se encarga de la infraestructura de hardware, la instalación de software del bróker, la seguridad de la red, la escalabilidad y el mantenimiento. Al momento de registrarse en la plataforma, se proporcionan tres elementos clave:

- URL de host única.
- Un puerto (8883 para MQTT puerto seguro).
- Usuario y contraseña de autenticación.

El bróker actúa como punto de encuentro entre el publicador (ESP32) y el suscriptor (Node-Red).

Ventajas de utilizar un bróker en la nube:

- **Accesibilidad:** Cualquier dispositivo con internet puede conectarse desde cualquier parte del mundo no solo en una red local.
- **Mantenimiento:** HiveMQ gestiona toda la infraestructura, como usuario no se debe realizar ninguna gestión.
- **Fiabilidad:** Alta, ya que se está alojando en clusters de servidores redundantes, garantizando una alta disponibilidad.
- **Escalabilidad:** Se pueden instalar millones de conexiones de ser necesario en planes de pago, en comparación con la local que deben compartir la misma red.
- **Dependencia:** Requiere internet todo el tiempo, si el WI-FI se cae del ESP32 no se pueden enviar datos.
- **Seguridad:** La conexión es segura por defecto (TLS/SSL -cifrado estándar de la industria).

Plan gratuito:

- 100 conexiones simultáneas: Permite que 100 dispositivos (clientes MQTT) estén conectados al mismo tiempo.
- 10 GB de tráfico de datos al mes: Los mensajes MQTT (como lecturas de sensores).
- Seguridad estándar: Autenticación robusta (usuario y contraseña) y cifrado de transporte TLS/SSL (Puerto 8883) permitiendo que los datos no viajen por el internet como texto plano.
- Soporte MQTT completo: Compatible con los estándares MQTT 5.0 y 3.1.1
- Niveles de Calidad de Servicio (QoS): QoS 1, QoS 2, QoS 3.

1.4.2. Node-RED

Es una herramienta de programación visual basada en flujos, construida sobre el entorno de ejecución del servidor Node.js. Su propósito fundamental es abstraer la complejidad de la programación de backend, permitiendo a los desarrolladores “conectar” (wire) diversos nodos de hardware, APIs (Interfaces de Programación de Aplicaciones) y servicios en línea mediante una interfaz gráfica en un navegador web.

La lógica se construye arrastrando “nodos” (representan piezas de funcionalidad, como “recibir MQTT” o “conectar a base de datos”) a un lienzo, y conectándose mediante “cables” (wires). Los datos fluyen a través de estos cables en forma de un objeto JavaScript estándar, conocido como el objeto “msg”. Cada nodo recibe un objeto “msg”, realiza una operación sobre él (modificando su “msg.payload” o “msg.topic”) y lo pasa al siguiente nodo del flujo.

Es la capa lógica central que se sitúa entre el bróker de mensajería (que solo retransmite datos) y la base de datos (que solo almacena datos). Su función no es ni generar, ni almacenar los datos, sino recibir, transformar, enriquecer, filtrar y enrutar esos datos.

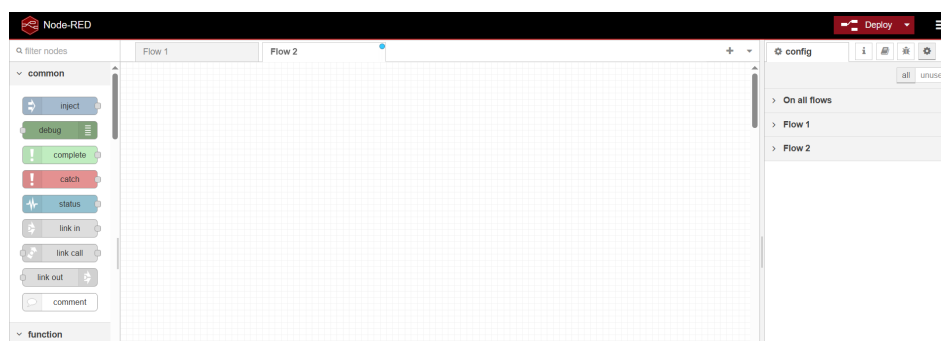


Figura 1.8: Interfaz de Node-RED.

Componentes a utilizar

- **Paleta de Nodos (Node Palette):** Ubicada en el lateral izquierdo, dispone de los bloques de programación preinstalados clasificados por función (common, function, network, dashboard). La arquitectura abierta de Node-RED permite incorporar nuevos nodos mediante la función “Manage Palette”, facilitando la integración de protocolos específicos o bases de datos no nativas.
- **Flow (Flujo de Trabajo):** Representa la unidad lógica de procesamiento. El entorno permite la creación de múltiples flujos en paralelo, lo que otorga modularidad al sistema. Esta característica válida la escalabilidad del proyecto, permitiendo gestionar la lógica de múltiples departamentos o procesos (calefacción, ventilación) en pestañas separadas sin interferencia entre ellas.
- **Panel Lateral (Sidebar):** Situado a la derecha, concentra las herramientas de diagnóstico. Incluye la pestaña “Debug”, utilizada para monitorear el tráfico de mensajes y detectar errores en tiempo real, y la pestaña “Info”, que proporciona documentación técnica detallada sobre la configuración y uso de cada nodo seleccionado.

1.4.3. InfluxDB : Base de Datos de Series Temporales (TSDB)

Es un sistema de gestión de bases de datos (DBMS) de código abierto, escrito en Go, diseñado y optimizado específicamente para el almacenamiento y la consulta de datos de series temporales (time-series data). A diferencia de las bases de datos relacionales (como MySQL) que organizan los datos en tablas y relaciones, InfluxDB es una base de datos NoSQL cuyo modelo de datos está indexado fundamentalmente por el tiempo. Su función es recibir los datos procesados de Node-RED y almacenarlos de forma permanente para su posterior análisis y visualización. Sin InfluxDB, los datos de sensores desaparecerían un instante después de ser procesados por Node-RED.

Protocolo de línea (Line Protocol). Técnicamente, Node-RED no se comunica con InfluxDB mediante comandos SQL. En su lugar, utiliza la API HTTP/S de InfluxDB para escribir datos formateados en un estándar de texto plano de alta eficiencia llamado Protocolo de Línea (Line Protocol). El nodo “influxdb out” en Node-RED se encarga de tomar el objeto JSON y lo transforma al protocolo de línea antes de enviarlo. Un ejemplo es el siguiente:

```
[Sist_ventilacion,id_equipo=VDF_bomb_1, variable=corriente valor=5.7, timeslap]
```

- **Measurement (Sist_ventilacion):** Contenedor principal de los datos.
- **Tag Set (id_equipo=VDF_bomb_1, variable=corriente):** Conjunto de metadatos indexados. Los tags se utilizan para filtrar y agrupar esta información.
- **Field Set: (valor=5,7)** Conjunto de datos reales o mediciones.
- **Timeslap:** Es la marca de tiempo en nanosegundos.

Características de plan gratuito

- Escritura: 5MB cada 5 min (17kb/s).
- Lectura: 300 MB cada 5 min.
- Retención de datos: 30 días.
- Hasta 2 bucket o carpetas.

1.4.4. Grafana: Plataforma visual

Grafana es una plataforma de software de código abierto, líder en la industria para la visualización de datos, el monitoreo y el análisis de series temporales. Su función principal es actuar como una potente e interactiva interfaz de visualización que se conecta a diversas fuentes de datos (data sources) donde los datos ya están almacenados. Su arquitectura está diseñada para consultar múltiples fuentes de datos (como InfluxDB, Prometheus, MySQL, etc.) de forma simultánea, agregar los resultados y presentarlos en un panel de control (dashboard) unificado. Utiliza un modelo de cliente-servidor, donde Grafana actúa como un servidor web que expone una interfaz de usuario, basada en navegador, para la creación y visualización de estos dashboards.

Funciones en el proyecto.

- **Conexión a la data source:** Se configura Grafana para que apunte a la base de datos InfluxDB como su fuente de datos.
- **Renderizado de datos:** Grafana envía esta consulta a InfluxDB, que responde con los datos de series temporales. Luego renderiza (dibuja) esos datos en el tipo de visualización escogido (gráfico de líneas).
- **Monitoreo y Alertas Visuales:** permite definir umbrales (thresholds) directamente en los gráficos. Es posible configurar que, si la temperatura (“msg.payload.temperatura” leída de InfluxDB) supera los 80 °C, el fondo del gráfico cambie a color rojo, proporcionando una alerta visual inmediata al operador.

Conexión del usuario.

- **Alojamiento del hosting:** Grafana se instala en un servidor (PC donde se ejecute Node-Red e InfluxDB).
- **Acceso por navegador:** El usuario solo debe abrir el navegador web de su preferencia (Safari, Chrome, etc) y puede ingresar de manera local con la dirección IP del servidor donde Grafana está corriendo, seguido del puerto que por defecto es 3000.
- **Local:** http://192.168.1.100:3000
- **Nube:** http://nombre_del_proyecto.grafana.net:3000

- **Autenticación:** Grafana presenta una pantalla de inicio de sesión. El usuario debe ingresar un nombre de usuario y contraseña (que se crea previamente en la configuración de Grafana) para poder acceder a los dashboards.



Figura 1.9: Ejemplo de Dashboard en Grafana.

2. Ingeniería de hardware y acondicionamiento de señales

Durante el diseño de la arquitectura de hardware de este sistema, surge un desafío común en la industria: la integración del mundo analógico tradicional (sensores) con el mundo digital de control (variadores), centralizando todo en un microcontrolador de bajo voltaje como el ESP32. La respuesta es una topología híbrida, dividida en tres etapas críticas: Acondicionamiento Analógico, Interfaz de Comunicación Digital y Gestión de Energía.

2.1. La Rama Analógica: Del Lazo de Corriente al Voltaje (Acondicionamiento)

En el entorno industrial ideal, el monitoreo de temperatura se realiza mediante sensores PT-100. Estos, por sí solos, son resistivos y difíciles de leer a larga distancia. Por ello, en el diseño teórico se integra un transmisor de temperatura, el cual se alimenta con la fuente industrial de 24 V y convierte la temperatura en una señal estándar de corriente de 4-20 mA.

Aquí surge el primer problema técnico: El ESP32 es un dispositivo que lee voltaje (0 V a 3,3 V), no corriente. Conectar la señal de 4-20 mA de manera directa ocasionaría daños en el microcontrolador. Para solucionar esto, se diseñó una etapa de conversión corriente-voltaje utilizando una resistencia Shunt de precisión de 150 Ohm conectada entre la salida del transmisor y la tierra (GND).

La matemática detrás: Según la Ley de Ohm ($V = I \times R$), cuando circulan 20 mA (máxima temperatura) por 150 Ohm se generan exactamente 3,0 V. El resultado: Esto transforma la señal industrial peligrosa en un rango seguro de 0,6 V a 3,0 V, perfecto para el convertidor analógico-digital (ADC) del ESP32.

2.1.1. Validación y calibración del prototipo

El lazo industrial de medición de temperatura se validó en el prototipo físico mediante la integración de una sonda RTD PT-100 y un transmisor de 4-20 mA. Para asegurar la precisión de la lectura analógica, se llevó a cabo un procedimiento de calibración instrumental del transmisor utilizando un potenciómetro de precisión y un multímetro digital. Basándose en la curva característica normalizada de la PT-100, se ajustó el potenciómetro a un valor de 100 Ohm (equivalente a 0 °C) y se reguló el potenciómetro de “Cero” (Zero) integrado en el transmisor hasta obtener una corriente de salida exacta de 4 mA. Posteriormente, se ajustó el potenciómetro de prueba a 138,5 Ohm (equivalente a 100 °C) y se calibró la “ganancia” (Span) del transmisor hasta alcanzar los 20 mA. De esta forma, se garantizó que el rango completo del sensor operara correctamente y de forma lineal antes de ser acondicionado por la resistencia shunt hacia el convertidor analógico-digital (ADC) del ESP32.

2.2. La Escalabilidad: Multiplexación de Entradas

Como el objetivo del proyecto es monitorear múltiples variables en distintas salas (bombas, calderas, ventilación), los pines analógicos (ADC) del ESP32 se agotarán rápidamente. Para resolver esta limitante, se integró en la topología el Multiplexor CD74HC4067. Este chip actúa como un “conmutador digital de alta velocidad”. Este componente permite conectar hasta 16 sensores (simulados o reales acondicionados) y enviarlos a un solo pin GPIO del ESP32 (por ejemplo, el GPIO 34). El ESP32 simplemente le informa al multiplexor “dame el sensor 1”, luego “dame el sensor 2”, etc., optimizando drásticamente el uso de recursos del microcontrolador.

2.3. La Rama Digital: Comunicación con el VDF (Modbus RTU)

Por otro lado, para el monitoreo del Variador de Frecuencia (VDF) Mitsubishi F-800, se prescindió del uso de sensores externos, procediendo a interrogar directamente al equipo. El VDF utiliza el protocolo Modbus RTU sobre una capa física RS-485. El desafío aquí es que el RS-485 usa voltajes diferenciales (A y B) que el ESP32 no entiende, ya que el ESP32 se comunica por UART (Serial TTL). La solución de ingeniería implementada es el módulo MAX485. Este componente sirve de “traductor”: toma las señales diferenciales del cable que viene del VDF y las convierte a niveles lógicos de 3,3 V que entran a los pines RX y TX (GPIO 16 y 17) del ESP32. Esto permite que el algoritmo solicite datos de frecuencia, corriente y voltaje directamente al controlador del variador, por los registros modbus, en este caso:

- 40201: Frecuencia.
- 40202: Corriente.
- 40203: Voltaje.

Monitor item	Increment and unit	Pr. setting	RS-485	MODBUS RTU	Negative indication (-) ^{*1}	Description
Output frequency (speed) ^{*17}	0.01 Hz ^{*16}	1/0/100	H01	40201	○ ^{*19}	The inverter output frequency is displayed.
Output current ^{*7*9*17}	0.01/0.1 A ^{*6}	2/0/100	H02	40202		The inverter output current effective value is displayed.
Output voltage ^{*7*17}	0.1 V	3/0/100	H03	40203		The inverter output voltage is displayed.

Figura 2.1: Parámetros Modbus.

2.4. Gestión de Energía: La Fuente Única y el Step-Down

Finalmente, para evitar el uso de múltiples enchufes y fuentes, se centralizó la alimentación. Se utilizó una fuente principal de 24 V, que es el estándar para alimentar el transmisor de 4-20 mA y los periféricos industriales. Sin embargo, el ESP32 funciona a 5 V (vía pin VIN) o 3,3 V. Conectar los 24 V directo lo quemaría instantáneamente. Para esto, se integra un convertidor DC-DC Step-Down LM2596. A diferencia de un regulador lineal que desperdicia energía en calor, el LM2596 reduce eficientemente los 24 V de la fuente principal a unos estables 5 V para alimentar el ESP32, el módulo MAX485 y el Multiplexor, cerrando así el circuito de alimentación de manera robusta y eficiente.

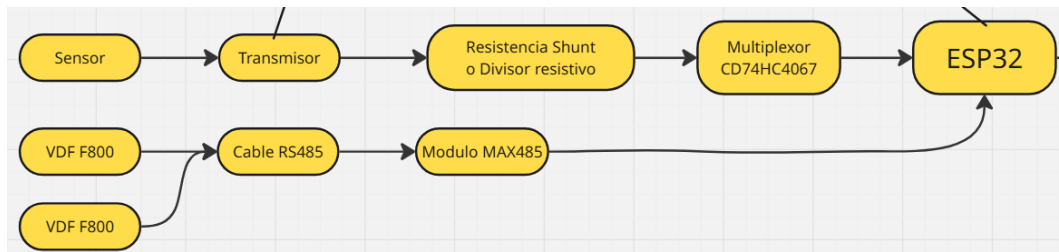


Figura 2.2: Diagrama de flujo de la arquitectura de hardware propuesta.

3. Implementación de la infraestructura de software y telemetría

Este capítulo expone de manera detallada la ingeniería de software empleada en la construcción de la infraestructura de comunicaciones y gestión de datos. Se describe el procedimiento técnico de forma secuencial, abarcando desde el despliegue del servidor de mensajería en la nube (HiveMQ), la implementación de la lógica de orquestación en Node-red, hasta el almacenamiento persistente de la información en InfluxDB.

3.1. Implementación del broker MQTT en la nube (HiveMQ)

Para garantizar la disponibilidad de los datos desde cualquier ubicación y evitar las complejidades de configurar direcciones IP públicas estáticas en la red local, se implementó un bróker MQTT alojado en la nube mediante la plataforma HiveMQ Cloud. Esta etapa constituye la columna vertebral de la comunicación M2M (Machine-to-Machine).

3.1.1. Despliegue del Cluster y Configuración de Red

El primer paso consistió en la creación de un Serverless Cluster en la plataforma HiveMQ. Al finalizar el despliegue, la plataforma asignó una dirección URL del host que sirve como punto de entrada único para todos los dispositivos. Además, para la capa de transporte, se configuró el uso estricto del puerto 8883, lo que fuerza el uso de seguridad de la capa de transporte (TLS/SSL). Esto asegura que los paquetes de datos viajen cifrados desde el ESP32 hasta la nube, protegiendo la información sensible de la planta contra interceptaciones en la red pública.

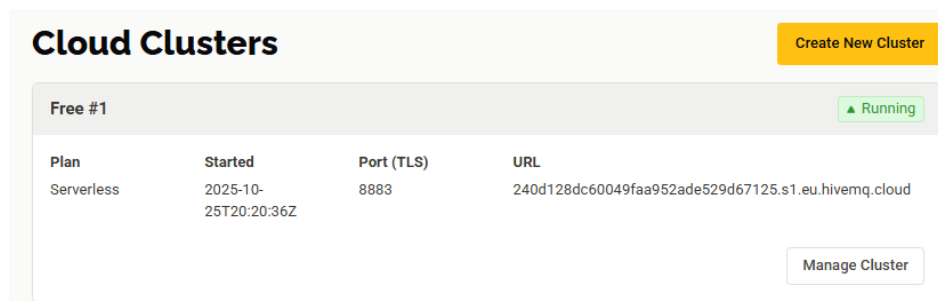


Figura 3.1: Despliegue del Cluster y Configuración de red en HiveMQ.

3.1.2. Gestión de Identidad y Accesos (Credentials)

La estructura administrativa en HiveMQ Cloud comienza con la definición de la organización. Al registrar la cuenta, se estableció una organización raíz (vinculada al correo del administrador, por nombre “Universidad Técnica Federico Santa María”) que actúa como el contenedor lógico bajo el cual se despliegan los clústeres y se gestionan las políticas de seguridad. Una vez provisionado el clúster dentro de esta organización, se procedió a la configuración de la seguridad en la sección “Access Management”.

Creación de credenciales. Para la fase de prototipo y validación, se implementó una estrategia de credencial unificada. Se generó un único usuario denominado “tesis”, al cual se le asignó una contraseña alfanumérica segura.

Asignación de Permisos (Permissions). Al usuario “tesis” se le otorgó el rol de “PUBLISH_SUSCRIBE”. Este perfil de permisos completos permite que la misma credencial sea utilizada tanto por el dispositivo de borde (ESP32) para enviar datos (Publicar), como por el servidor central (Node-RED) para recibirlos (Suscribir) y por el cliente web para pruebas, simplificando la gestión de claves en el entorno de desarrollo.

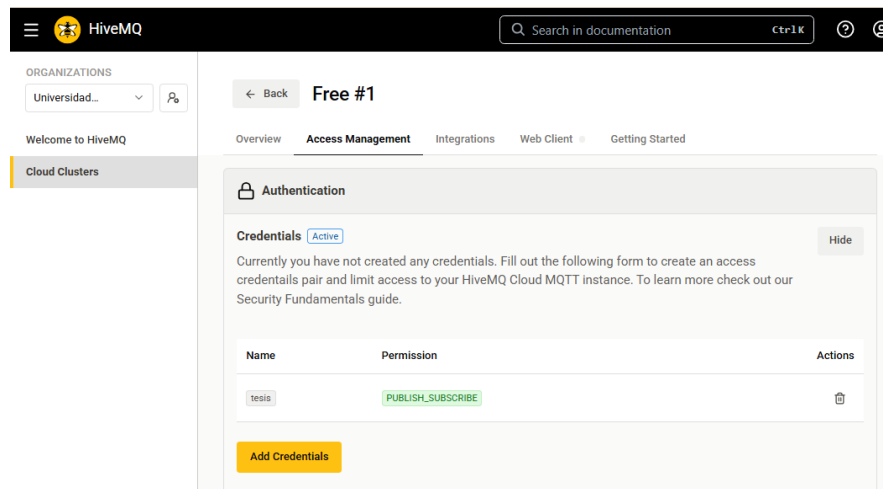


Figura 3.2: Gestión de credenciales y permisos en HiveMQ.

3.1.3. Definición de la Jerarquía de Tópicos (Topic Namespace)

Se diseñó una estructura de tópicos lógica y escalable para organizar el tráfico de mensajes. Además, con el fin de optimizar el ancho de banda y simplificar la lógica de transmisión en el microcontrolador, se implementó una estrategia de Tópico Unificado. En lugar de publicar mensajes individuales en múltiples canales, el nodo (ESP32) concentra todas las variables del departamento en una única transmisión periódica.

Tópico Implementado. Se definió la ruta “tesis/departamento1/sensores” Este canal actúa como el único punto de entrada de datos hacia la nube.

Estructura del Mensaje (JSON Payload). Los datos se encapsulan en un objeto estandarizado en formato JSON (JavaScript Object Notation). El ESP32 construye una cadena de texto estructurada que agrupa las variables críticas con claves identificables.

```

{
  "sala_bombas": {
    "presion": 8.5, alarma 8 bar
    "caudal": 120.5,
    "vfd_frecuencia": 50.0,
    "vfd_corriente": 12.5,
    "vfd_voltaje": 380.0
  },
  "sala_caldera": {
    "presion_gas": 22.0,
    "presion_agua": 4.2,
    "temp_agua": 75.5,
    "amp_bomba": 5.5,
    "volt_bomba": 220.1
  },
  "sala_ventilacion": {
    "voltaje": 219.8,
    "corriente": 3.2
  }
}

```

Figura 3.3: Estructura del mensaje JSON enviado por el ESP32.

3.1.4. Pruebas de Conectividad (Web Client)

Para verificar la correcta propagación de los permisos antes de programar los dispositivos físicos, se utilizó la herramienta de diagnóstico “Web Client” integrada en el panel de HiveMQ. Se ingresó con las credenciales del usuario tesis. Se realizó una suscripción de prueba al tópico “tesis/departamento1/sensores”. Se envió un mensaje manual (“payload”) desde la misma interfaz. La recepción exitosa del mensaje confirmó que el usuario “tesis” estaba activo y que el puerto seguro (TLS) del clúster operaba correctamente dentro de la organización configurada.

The screenshot displays the HiveMQ Web Client interface. At the top, there's a 'Topic Subscriptions' section with a tab indicator '1' and a button 'Unsubscribe from all topics'. Below this, a text block explains how to subscribe to topics and set the Quality of Service (QoS). A table lists the current subscription: 'tesis/departamento1/sensores' with 'QoS: 1' and an 'Actions' column containing a trash icon. Below the table is a 'Subscribe' form with a 'Topic Name' input field and a 'Subscribe' button. The bottom section is titled 'Send Message' with a help icon. It includes a text block about message visibility, a 'Message' input field with placeholder text 'Your message', and a 'Send Message' button. Below the message input, there's a 'Topic' input field with 'tesis/departamento1/sensores' and a 'QoS' dropdown menu set to 'QoS: 1'.

Figura 3.4: Interfaz Web Client para envío y suscripción de pruebas.

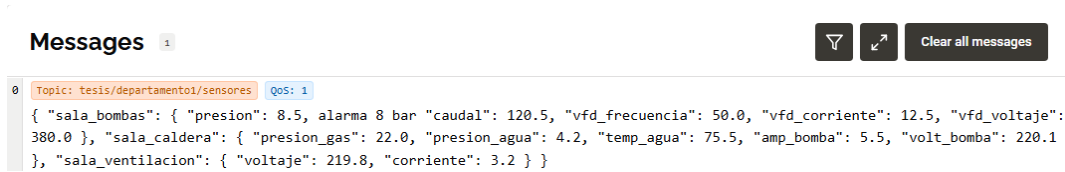


Figura 3.5: Recepción exitosa del mensaje de prueba en la consola.

3.2. Procesamiento de datos Node-RED

El núcleo lógico del sistema se desarrolló sobre Node-RED. Esta plataforma actúa como middleware, recibiendo los datos crudos, procesándolos, evaluando condiciones de alarma y distribuyéndolos hacia la base de datos y los sistemas de notificación.

3.2.1. Palette Manager

Para extender las capacidades nativas de Node-RED, se procedió a la instalación de módulos específicos mediante el gestor de paquetes Manage Palette. Para esto se integraron las siguientes librerías:

- **node-red-contrib-influxdb:** Habilita los nodos de salida para la escritura en bases de datos de series temporales. Incluye 4 nodos.
- **node-red-contrib-telegrambot-home:** Proporciona los nodos receptores y emisores para la interacción con la API de Telegram. Incluye 5 nodos.

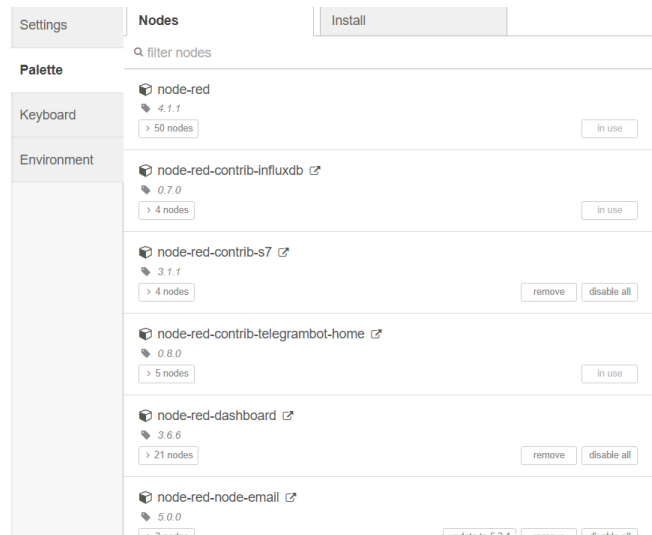


Figura 3.6: Instalación de nodos adicionales en Palette Manager.

3.2.2. Configuración de nodo MQTT

La entrada de datos al sistema se realiza mediante un nodo “mqtt in”.

Configuración del Servidor. Se definió la conexión apuntando al host del clúster HiveMQ (URL), puerto 8883 y protocolo seguro TLS.

Figura 3.7: Configuración de conexión al servidor MQTT HiveMQ.

Suscripción. Se configuró el tópico de suscripción como tesis/departamento1/sensores. El uso del carácter comodín # (Wildcard) permite que un único nodo de entrada capture el tráfico de todas las salas simultáneamente, optimizando el consumo de recursos del servidor.

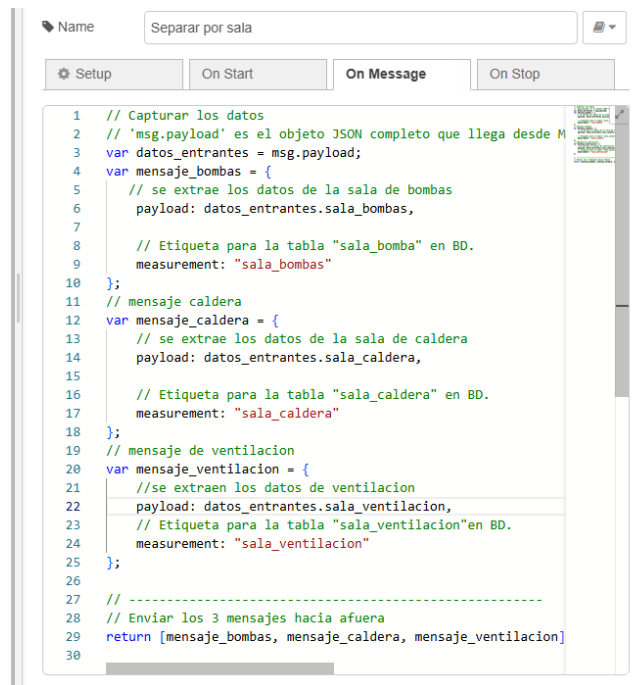
Formato de Salida. Se forzó la salida como “a parsed JSON object”. Esto convierte automáticamente la cadena de texto recibida (String) en un objeto JavaScript manipulable, permitiendo acceder a las propiedades como “msg.payload.temperatura” en los nodos subsiguientes.

Figura 3.8: Configuración del tópico y formato de salida JSON.

3.2.3. Lógica de Demultiplexación y Segregación de Variables (Function Node)

Debido a que el dispositivo transmite todas las variables encapsuladas en un único objeto JSON a través de un solo tópico, se implementó una etapa de procesamiento para “desempaquetar” y distribuir la información hacia sus respectivos canales de almacenamiento. Para ello, se utilizó un nodo de tipo “Function” programado en JavaScript, configurado con 3 salidas distintas. Este nodo recibe el mensaje global y genera tres sub-mensajes independientes, extrayendo las propiedades específicas para cada sala.

Código de Distribución Implementado: Dentro del nodo, se programó la siguiente lógica para separar el objeto `msg.payload`.



```

1 // Capturar los datos
2 // 'msg.payload' es el objeto JSON completo que llega desde M
3 var datos_entrantes = msg.payload;
4 var mensaje_bombas = {
5   // se extrae los datos de la sala de bombas
6   payload: datos_entrantes.sala_bombas,
7
8   // Etiqueta para la tabla "sala_bomba" en BD.
9   measurement: "sala_bombas"
10 };
11 // mensaje caldera
12 var mensaje_caldera = {
13   // se extrae los datos de la sala de caldera
14   payload: datos_entrantes.sala_caldera,
15
16   // Etiqueta para la tabla "sala_caldera" en BD.
17   measurement: "sala_caldera"
18 };
19 // mensaje de ventilacion
20 var mensaje_ventilacion = {
21   //se extraen los datos de ventilacion
22   payload: datos_entrantes.sala_ventilacion,
23   // Etiqueta para la tabla "sala_ventilacion" en BD.
24   measurement: "sala_ventilacion"
25 };
26
27 // -----
28 // Enviar los 3 mensajes hacia afuera
29 return [mensaje_bombas, mensaje_caldera, mensaje_ventilacion]
30

```

Figura 3.9: Código JavaScript para la segregación de variables por sala.

Resultados del procesamiento

- **Salida 1:** Entrega exclusivamente los datos al nodo InfluxDB de “sala_bombas”.
- **Salida 2:** Entrega exclusivamente los datos al nodo InfluxDB de “sala_caldera”.
- **Salida 3:** Entrega exclusivamente los datos al nodo InfluxDB de “sala_ventilacion”.

Esta estrategia de segregación por software optimiza el orden en la base de datos, asegurando que cada variable se almacena en su tabla (measurement) correcta sin mezclar información de distintas áreas. El nodo de InfluxDB aún no será configurado ya que se necesita ingresar directamente a la página.

3.2.4. Sistema de Detección y Gestión de Alertas Críticas

Paralelamente al almacenamiento de datos, se derivó una rama del flujo para el monitoreo de seguridad en tiempo real. Esta etapa tiene como objetivo identificar valores fuera de rango y notificar al personal de mantenimiento de manera inmediata. La lógica se implementa mediante una cadena de procesamiento secuencial: Evaluación, Formateo y Notificación.

Evaluación Condicional (Switch node). Para discriminar los estados de operación normales de los críticos sin recurrir a código complejo, se utilizó un nodo de control de flujo tipo “switch”. Este nodo actúa como una compuerta lógica que solo permite el paso del mensaje si se cumple una condición numérica específica.

- **Propiedad Evaluada:** Se configuró el nodo para analizar la variable específica proveniente del paso anterior (“msg.payload.sala_caldera.temp_agua” o “msg.payload.sala_bombas.presion”).
- **Reglas de Disparo:** Se establecieron operadores de comparación matemática (“<” o “>”) contra los Setpoints de seguridad definidos.
- **Ejemplo:** Si `msg.payload.sala_caldera.temp_agua > 65°C`, el mensaje se enruta hacia el puerto de salida de alarma.

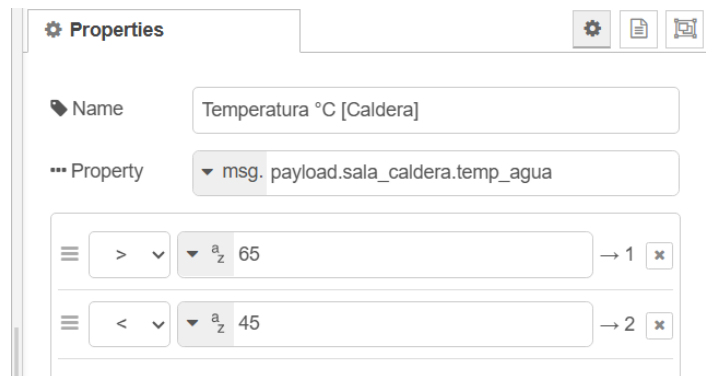


Figura 3.10: Configuración de umbrales en el nodo Switch.

Rangos a medir de variables críticas sobre la climatización

Ubicación	Medición	Rango Medible	Alarma	Tipo de Sensor
Sala de Bombas	Presión de agua	0-12 bar	Bajo 7 bar Sobre 9 bar	sala_bombas.presion
	Caudal de agua	120 L/min	Sin alarma	sala_bombas.caudal
	Corriente de bombas	12-15 A	Sobre 16,5 A	sala_bombas.vdf_corriente
	Frecuencia de bombas	50 Hz	Sin alarma	sala_bombas.vdf_frecuencia
	Voltaje de bombas	380 V	Bajo 350 V Sobre 410 V	sala_bombas.vdf_voltaje
Sala de Caldera	Presión de Gas	28 mbar presión de red 18mbar	Bajo 18 mbar	sala_caldera.presion_gas
	Presión de Agua secundario	0-6 bar	Bajo 0,5 bar Sobre 2,5 bar	sala_caldera.presion_agua
	Temperatura	60 °C	Bajo de 45 °C	sala_caldera.temp_agua
	Corriente de bombas (BRCA)	6 A	Sobre 7 A	sala_caldera.amp_bomba
	Voltaje de entrada tablero	380 V	Bajo 350 V Sobre 410 V	sala_caldera.presion_agua
Sala de Ventilación	Voltaje	380 V	Bajo 350 V Sobre 410 V	sala_ventilacion.voltaje
	Corriente	4,5 A	Sobre 5,5 A	sala_ventilacion.corriente

Tabla 3.1: Variables críticas, rangos de medición y condiciones de alarma.

Construcción del Mensaje de Alerta (Template Node). Una vez que el nodo “Switch” valida que existe una anomalía, el mensaje pasa a un nodo “Template”. Su función es transformar el dato numérico crudo en un mensaje de texto comprensible para el operador. Se utilizó el motor de plantillas Mustache para insertar dinámicamente los valores del sistema dentro del texto. Este nodo reemplaza automáticamente `{{payload.sala_caldera.temp_agua}}` por el valor real registrado en el momento del evento (ej. “65 °C”), generando un reporte de incidente preciso.

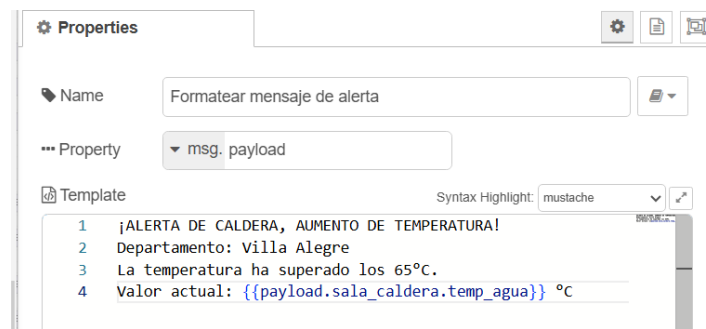


Figura 3.11: Formateo del mensaje de alerta usando plantillas Mustache.

Gestión de Frecuencia de Notificaciones (Delay Node). Para evitar la saturación del canal de comunicación (Spam) en caso de una falla persistente donde el sensor envía datos fuera de rango cada segundo, se implementó un nodo “Delay” en configuración avanzada.

- **Action:** “Rate Limit” (Límite de tasa).
- **Rate:** “1 mensaje cada 1 hora”.
- **Drop intermediate messages:** Activado. Esta configuración asegura que, aunque la falla persista, el operador solo reciba un recordatorio cada 10 minutos, evitando el bloqueo del bot de Telegram por exceso de peticiones.

The image shows the 'Properties' panel for a Node-RED node. The 'Action' dropdown is set to 'Rate Limit'. Below it, the 'All messages' dropdown is set to 'All messages'. The 'Rate' section has two input fields: '1' for 'msg(s) per' and '1' for 'Hour'. There is an unchecked checkbox labeled 'allow msg.rate (in ms) to override rate'. Below that, the 'Drop intermediate messages' dropdown is set to 'Drop intermediate messages'. At the bottom, the 'Name' field contains the text '1 hrs/ sin alarma'.

Figura 3.12: Configuración del límite de tasa para evitar spam de alertas.

3.2.5. Sistema de Notificación (Telegram Bot y Notify Node)

Para la salida de alertas hacia los dispositivos móviles de los operadores, se integró la API de Telegram.

Creación del Bot. Se utilizó el bot maestro de Telegram llamado @BotFather. Mediante el comando “/newbot”, se asignó el nombre y usuario al bot del proyecto, obteniendo como respuesta el API Token (clave alfanumérica única).

Obtención del Chat ID. Se utilizó el bot @userinfobot desde la cuenta del administrador para obtener el número identificador único (Chat ID) al cual se deben enviar los mensajes.

Configuración del Nodo Telegram Sender. En Node-RED, se arrastró el nodo de envío de Telegram. En su configuración interna (Bot Configuration), se pegó el API Token obtenido. Finalmente, en el campo “Chat Id”, se ingresó el identificador del administrador. Este nodo recibe el texto formateado del nodo Template y lo despacha inmediatamente a la aplicación móvil.

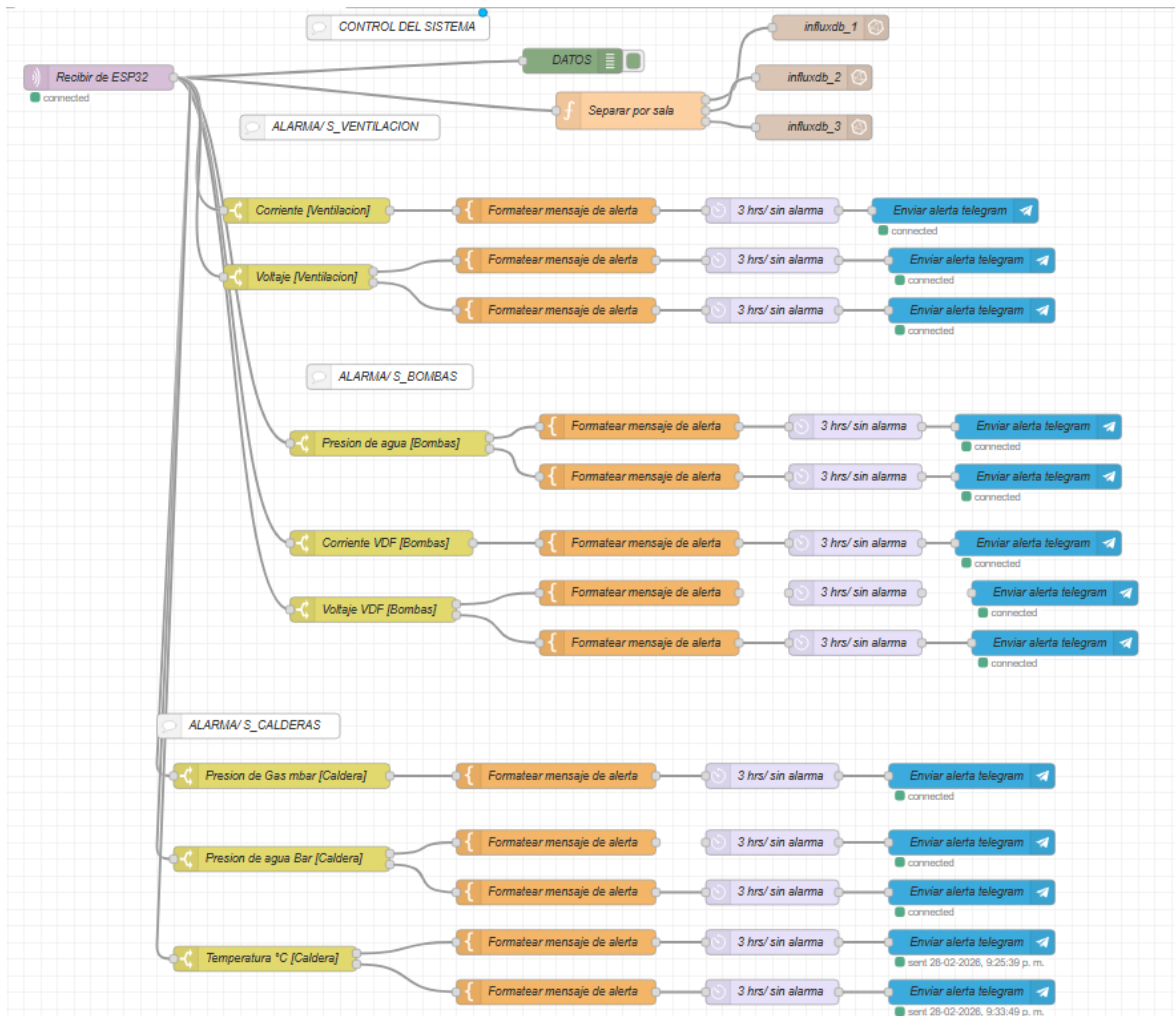


Figura 3.13: Flujo completo de procesamiento en Node-RED.

3.3. Gestión de Base de Datos de Series de Tiempo (InfluxDB)

El almacenamiento histórico se confió a InfluxDB debido a su alto rendimiento en la escritura de datos indexados por tiempo (Time-Series), fundamental para el análisis de tendencias industriales, además de sus óptimas condiciones en plan gratuito.

3.3.1. Configuración de la Organización y Buckets

En la interfaz de administración de InfluxDB, se estableció la estructura lógica de almacenamiento:

- **Organización:** Se definió una organización raíz para el proyecto (USM).
- **Buckets (Cubos):** A diferencia de las bases de datos SQL tradicionales, InfluxDB utiliza Buckets. Se creó un Bucket llamado “Tesis-ESP32” con una política de retención (Retention Policy) configurada a “Infinite” (o 30 días según el caso), asegurando que los datos históricos no sean borrados automáticamente.

3.3.2. Generación de Token de Seguridad (API Token)

Para permitir que Node-RED escriba datos en la base de datos sin comprometer la cuenta de administrador completa, se generó un “API Token” con permisos específicos de lectura/escritura (Read/Write) limitados exclusivamente al Bucket “Tesis-ESP32”. Este Token es la llave que se utiliza en la configuración de Node-RED.

3.3.3. Integración de Nodos de Salida (InfluxDB Out)

En Node-RED, se utilizaron tres nodos “influxdb out” independientes, conectados a las salidas del nodo Switch descrito en la sección.

- **Configuración del Servidor:** Se creó una nueva configuración de servidor apuntando a localhost de Node-RED y puerto 8086, utilizando la Organización definida y el Token generado anteriormente “Tesis-ESP32”.
- **Measurement (Medición):** En cada uno de los tres nodos, se especificó un nombre de medición distinto para etiquetar los datos en la base de datos:
 - Nodo 1: Measurement. sala_bombas.
 - Nodo 2: Measurement sala_caldera.
 - Nodo 3: Measurement sala_ventilacion.

Esta separación final asegura que, al momento de graficar en el siguiente capítulo, las consultas a la base de datos sean rápidas y específicas para cada área del proceso.

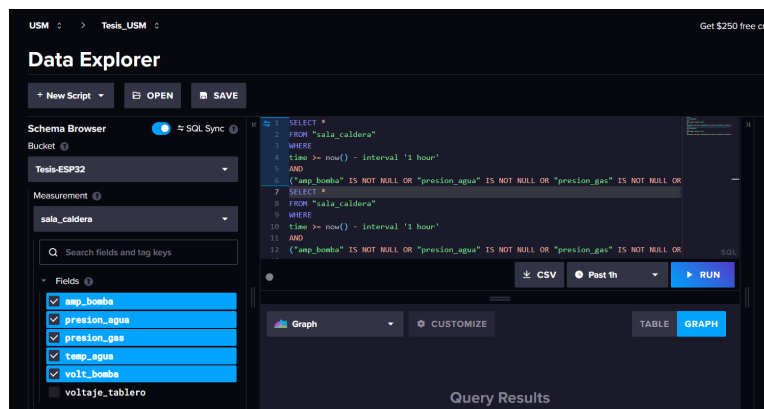


Figura 3.14: Explorador de datos en InfluxDB mostrando las mediciones ingresadas.

4. Implementación de la Plataforma de Visualización y Supervisión (Grafana)

Este capítulo describe el desarrollo de la Interfaz Hombre-Máquina (HMI) web. Se detalla el despliegue del servicio en la nube utilizando el plan gratuito de Grafana, la vinculación segura con la base de datos y la construcción paso a paso de los tableros de control (Dashboards), basado en las tres salas de proceso y, fundamentalmente, la ingeniería de software aplicada a las consultas de datos (Queries) en lenguaje Flux, diferenciando las estrategias para visualización en tiempo real frente al análisis histórico.

4.1. Creación de Instancia (Free Tier)

Se procedió al registro en la plataforma oficial de Grafana seleccionando el plan “Grafana Cloud Free”. Este plan proporciona:

- Retención de datos suficiente para el alcance del proyecto (14 días de historial).
- Acceso multiusuario (hasta 3 miembros).
- Disponibilidad 99,9 % garantizada por el proveedor.

Una vez activada la cuenta, se generó una instancia dedicada con una URL única (monitoreovariablescriticas.grafana.net), la cual sirve como portal de acceso para los operadores del sistema.



Figura 4.1: Portal de administración de Grafana Cloud.

4.2. Vinculación de la Fuente de Datos (Data Source)

4.2.1. Configuración del Conector Flux

Dentro del panel de administración de Grafana, se navegó a “Connections / Data Sources” y se seleccionó el motor InfluxDB. Dado que se utiliza la versión 2.x de la base de datos, la configuración requirió el uso del lenguaje de consulta Flux.

- **URL:** Se ingresó la dirección de la base de datos (la URL de InfluxDB Cloud).
- **InfluxDB Details:**
 - **Organization:** Se ingresó la organización definida “USM”
 - **Token:** Se pegó el API Token de lectura generado previamente.
 - **Default Bucket:** Se estableció “Tesis-ESP32” como el cubo de datos por defecto.

Database settings

Provide the necessary database connection details based on your selected InfluxDB product and query language.

Organization * USM

Default Bucket * Tesis-ESP32

Token * configured Reset

Advanced Database Settings ☐

Figura 4.2: Configuración de los parámetros de conexión a InfluxDB.

Se finalizó el proceso con la prueba “Save & Test”, confirmando que Grafana tiene acceso de lectura a los registros históricos.

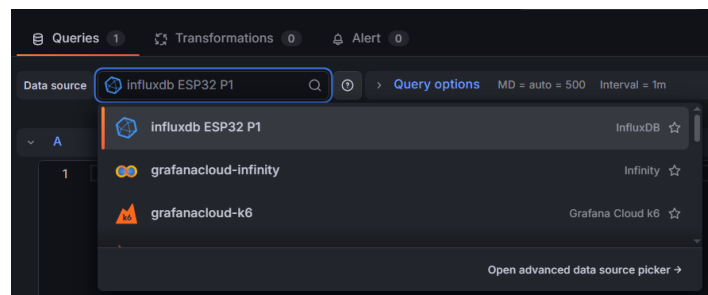


Figura 4.3: Selección y prueba del Data Source.

4.3. Diseño y Construcción de Dashboards

El diseño visual se estructuró en un panel único denominado “Monitor de Planta”, organizado en tres filas lógicas (Rows) que corresponden a la distribución física del recinto, para observar todas las variables críticas en una sola pantalla, de querer analizar el comportamiento de una variable en específico solo se debe ingresar directamente a ese gráfico.

- **Fila Superior (Sala de Bombas):** Dedicada al monitoreo hidráulico y del Variador de Frecuencia. Incluye indicadores numéricos de gran tamaño y gráficas de caudal/presión.
- **Fila Central (Sala de Calderas):** Enfocada en la termodinámica. Incluye gauge de arco para temperatura y presión, además del monitoreo eléctrico de la bomba auxiliar.
- **Fila Inferior (Sala de Ventilación):** Destinada exclusivamente al historial de consumo eléctrico (Corriente y Voltaje) de los extractores.

4.3.1. Implementación de Umbrales Visuales (Thresholds)

Para facilitar la supervisión rápida, se optó por una lógica binaria estricta de Permitido/No Permitido, implementada mediante la función “Thresholds” de Grafana.

Lógica de Color:

- **Verde (Zona Segura):** El valor se encuentra dentro de los parámetros operativos nominales.
- **Rojo (Zona Crítica):** El valor excede el límite de seguridad establecido, requiriendo intervención inmediata.

Caso de Aplicación:

- Temperatura caldera.

Configuración:

- Se estableció el límite en $< 65\text{ }^{\circ}\text{C}$ y $> 40\text{ }^{\circ}\text{C}$.

Comportamiento:

- Si Temperatura $< 65\text{ }^{\circ}\text{C}$ y Temperatura $> 40\text{ }^{\circ}\text{C}$: El arco y el texto se renderizan en verde.
- Si Temperatura $> 65\text{ }^{\circ}\text{C}$ o Temperatura $< 40\text{ }^{\circ}\text{C}$: El arco y el texto se renderizan en rojo.

4.4. Consultas Flux (Query Scripting)

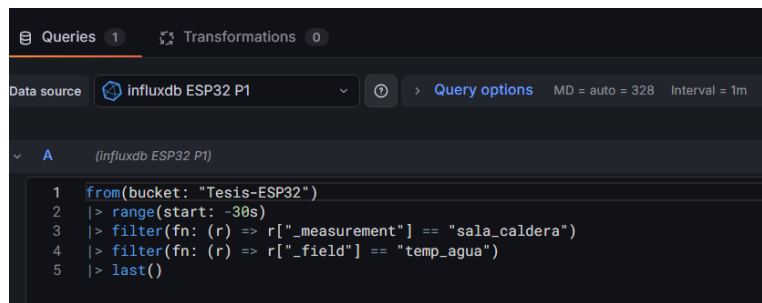
Este es el componente técnico más crítico. Para visualizar los datos correctamente, se desarrollaron dos tipos de algoritmos de consulta distintos: uno para Indicadores de Tiempo Real (Gauges) y otro para Gráficas Históricas (Time Series).

4.4.1. Código para Indicadores de Tiempo Real (Gauge y Stat)

Para los paneles tipo “Gauge” (Manómetro circular) y “Stat” (Texto grande), el objetivo es mostrar el último estado conocido del sistema.

Problema Técnico: Si se solicita a la base de datos traer el historial de “los últimos 3 días” para un manómetro, el sistema intentará promediar miles de datos, entregando un valor que no representa la realidad actual.

Solución Implementada: Se programó una consulta con una ventana de tiempo estática y reducida, forzando la obtención del último registro único.



```

1 from(bucket: "Tesis-ESP32")
2 |> range(start: -30s)
3 |> filter(fn: (r) => r["_measurement"] == "sala_caldera")
4 |> filter(fn: (r) => r["_field"] == "temp_agua")
5 |> last()
  
```

Figura 4.4: Script Flux para obtener el último valor registrado.

Análisis del Script:

- `range(start: -5m)`: Esta línea es crítica. Fija la búsqueda de datos estrictamente a los últimos 5 minutos, independientemente de lo que seleccione el usuario en el calendario de Grafana.
- `last()`: Esta función selecciona exclusivamente el último dato ingresado. No promedia ni suma.

Comportamiento “No Data”. Si el sistema deja de recibir datos por más de 5 minutos (por ejemplo, corte de energía o desconexión del ESP32), la consulta devuelve “vacío” y el panel muestra “No Data”. Esto actúa como una alerta implícita de “Pérdida de Comunicación”.

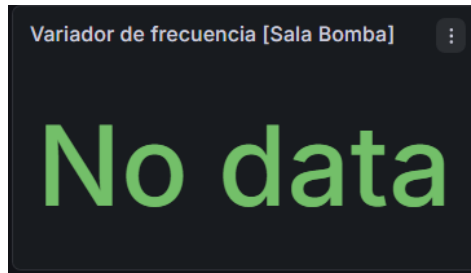


Figura 4.5: Indicador visual de pérdida de comunicación (No Data).

4.4.2. Código para Gráficas Históricas (Time Series)

Para los gráficos de línea (como el “Dato histórico de Corriente” en Sala Ventilación), el objetivo es visualizar tendencias a lo largo del tiempo (horas o días).

Problema Técnico. Cargar cada segundo de datos de un mes saturaría la memoria del navegador y haría la gráfica ilegible.

Solución Implementada. Se utilizó una consulta con rango dinámico y agregación de datos (promedio por ventanas).

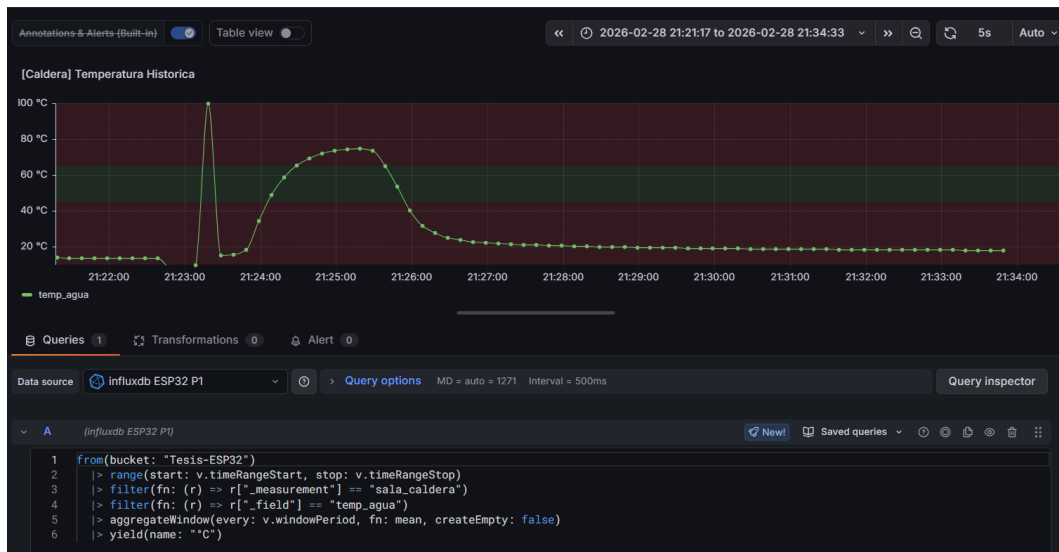


Figura 4.6: Script Flux con agregación dinámica para series de tiempo.

Análisis del Script:

- `Range(start: v.timeRangeStart...)`: A diferencia del código anterior, aquí se usan variables dinámicas. Si el usuario selecciona “Últimos 3 días” en Grafana, la gráfica obedecerá y mostrará ese periodo.
- `AggregateWindow(...,fn: mean)`: Esta función toma grupos de datos y calcula su promedio (mean). Grafana ajusta automáticamente el tamaño de la ventana (`v.windowPeriod`) según el zoom. Por ejemplo, si se ven 30 días, cada punto en la gráfica puede ser el promedio de 1 hora; si se ven los últimos 15 minutos, cada punto es el dato real.

4.5. Análisis de Resultados y Validación de la Integridad de Datos

Tras la configuración de los paneles y la inyección de la trama de datos de prueba (JSON), se procedió a validar la fiabilidad de la información desplegada. Este análisis se centra en tres aspectos críticos para la operación: la concordancia de los datos, el comportamiento ante pérdida de señal y la respuesta visual de las alarmas.

4.5.1. Verificación de Concordancia (Input vs Output)

Se realizó una prueba de integridad comparando los valores numéricos inyectados mediante el protocolo MQTT contra los valores renderizados en el Dashboard final.

Prueba: Se envió el vector de datos simulado `{“sala_bombas”:{“presion”: 8.2, ... }}`

Resultado en Grafana: El panel de presión indicó exactamente 8.2 Bar.

Análisis: Esto valida que la cadena completa de procesamiento (Decodificación JSON en Node-RED / Escritura en InfluxDB / Consulta Flux en Grafana) no introduce errores de redondeo, latencia excesiva ni pérdida de decimales. La información que ve el operador es una representación fiel de la variable física medida.



Figura 4.7: Validación de lectura correcta de presión (8.20 bar).

4.5.2. Análisis del Comportamiento ante Pérdida de Señal (Estado “No Data”)

Uno de los resultados más relevantes de la configuración de consultas (Flux Queries) fue la respuesta del sistema ante la ausencia de datos recientes.

Observación. Como se evidencia en los paneles de la Sala de Calderas, los indicadores de Temperatura y Nivel de Agua muestran el mensaje “No data”

Justificación Técnica. Este comportamiento no es un error, sino una validación exitosa del código `range(start: -5m)`. Al restringir la ventana de tiempo a los últimos 5 minutos, el sistema garantiza que nunca se muestre un dato obsoleto como si fuera actual.

Implicancia Operativa. En sistemas SCADA tradicionales, a veces un sensor “congelado” sigue mostrando el último valor (ej. 90 °C), engañando al operador. En esta propuesta, si el ESP32 deja de transmitir, el operador recibe una retroalimentación visual inmediata (“No Data”) indicando una falla de comunicación, lo cual es crítico para la seguridad.

4.5.3. Validación de la Lógica de Umbrales (Thresholds)

Se evaluó la eficacia de la semaforización binaria (Rojo/Verde) para la detección rápida de anomalías.

Escenario de Prueba. La presión de agua en la sala de caldera se elevó a 6,50 bar, superando el setpoint de seguridad definido en 6,0 bar.

Respuesta del Sistema:

- El panel “Presión de agua” cambió su color base a Rojo Sólido.
- El gráfico histórico adyacente registró la subida, pero mantuvo la línea de umbral visible para referencia.



Figura 4.8: Validación de cambio de color por superación de umbral.

Análisis. La transición de color fue instantánea al refresco del panel. Esto demuestra que la lógica de supervisión visual es efectiva, permitiendo al operador identificar una condición de riesgo (sobrepresión) en menos de 1 segundo mediante un simple escaneo visual, sin necesidad de leer o interpretar el valor numérico exacto.

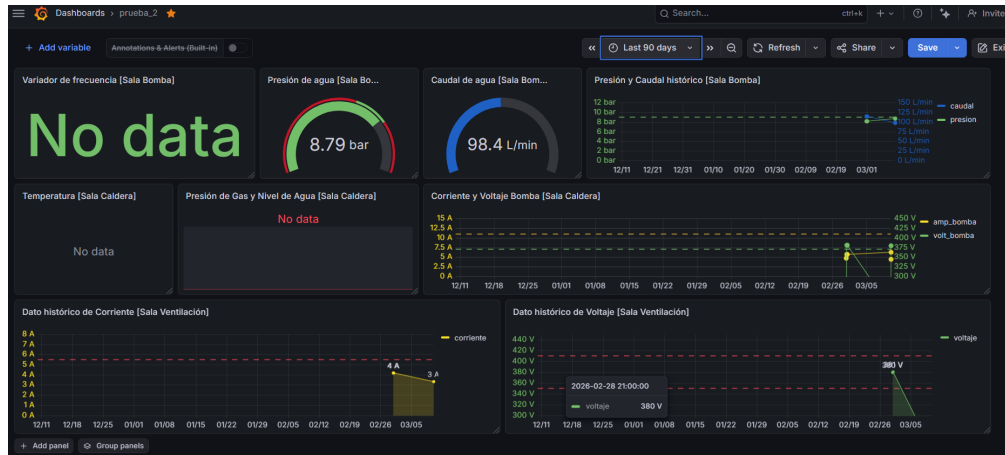


Figura 4.9: Estado del Dashboard ante pérdida de comunicación (No Data).

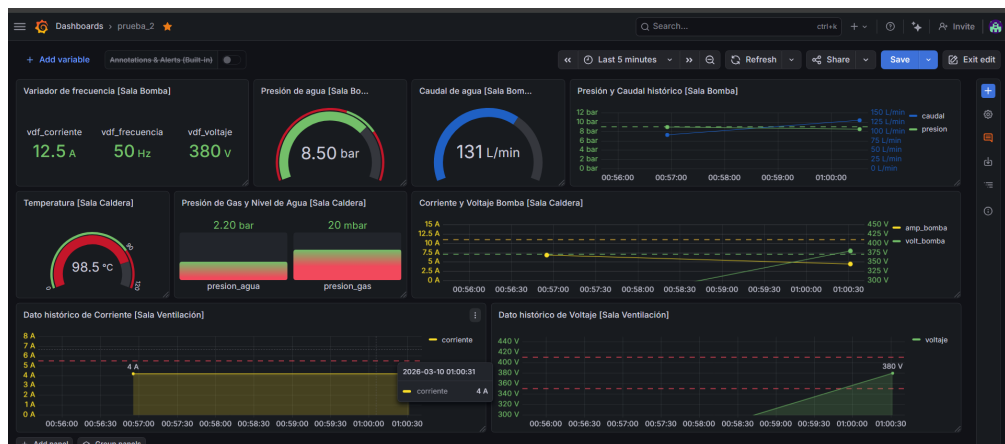


Figura 4.10: Estado operativo del Dashboard con visualización en tiempo real.

5. CONCLUSIONES Y TRABAJOS FUTUROS

5.1. Conclusiones

El desarrollo de esta tesis ha permitido validar exitosamente una arquitectura de bajo costo para la telemetría de variables críticas en sistemas de climatización. La implementación del prototipo demostró que es posible adquirir datos de campo y transmitirlos a la nube utilizando internet, logrando una visualización efectiva tanto en tiempo real como en registros históricos, accesibles desde cualquier ubicación geográfica.

Si bien la integración con protocolos industriales (Modbus RTU) se abordó desde un enfoque teórico para establecer las bases de la conexión con variadores de frecuencia, la parte experimental confirmó la solidez de la cadena de comunicación: Sensor - Microcontrolador (ESP32) - Broker MQTT - Base de Datos y Dashboard.

Se concluye que el uso de herramientas Open Source (Node-RED, Grafana, InfluxDB) democratiza el acceso a sistemas de monitoreo avanzados, permitiendo a pequeñas empresas o administradores de edificios tener un control que antes era exclusivo de grandes industrias con sistemas SCADA costosos.

5.2. Análisis de Variables Críticas y Diagnóstico Operativo

Para garantizar la continuidad operacional de una sala de operación, el sistema propone el monitoreo de múltiples variables. A continuación, se presenta un diagnóstico técnico de lo que implican las desviaciones en estas variables:

5.2.1. Temperatura de Agua

- **Rango Crítico Bajo ($< 45^{\circ}\text{C}$):** Indica falla en la transferencia térmica. En sistemas de acumulación, esto genera un riesgo sanitario crítico por proliferación de Legionella.
- **Rango Crítico Alto ($> 65^{\circ}\text{C}$):** Riesgo inminente de quemaduras a usuarios y estrés térmico en tuberías de polímero (PPR), acortando su vida útil.

5.2.2. Presión de Impulsión (Variable Hidráulica)

- **Baja Presión (Posible “Pozo Seco”):** Si la presión cae cerca de 0 bar mientras la bomba está energizada, indica falta de agua en la succión. Esto es letal para la bomba. Sin agua, el sello mecánico pierde lubricación y refrigeración, destruyéndose por fricción en cuestión de minutos. El sistema debe detener la bomba inmediatamente.
- **Alta Presión:** Indica obstrucción en la línea de descarga (válvula cerrada). Provoca fatiga mecánica en el eje de la bomba y riesgo de reventón en uniones débiles.

5.2.3. Corriente Eléctrica / Amperaje (Variable Eléctrica)

- **Sobrecorriente (Sobrecarga):** Si el consumo en Amperios supera el valor nominal de placa del motor (FLA). Puede indicar un problema mecánico (rodamientos granados, eje atascado) o que la bomba está moviendo un fluido más denso/sucio de lo previsto. Requiere inspección inmediata para evitar quemar el bobinado.
- **Baja Corriente (Subcarga):** Confirma el diagnóstico de “Pozo Seco” o indica una rotura en el acople entre el motor y la bomba (el motor gira “en vacío”).

5.2.4. Voltaje de Alimentación (Variable de Red)

- **Desbalance de Fases o Bajo Voltaje:** Si el voltaje cae más del 10 %, el motor intentará compensar subiendo la corriente (Amperaje), lo que elevará la temperatura interna del motor. Como consecuencia, la degradación prematura del aislamiento del motor y disparos intempestivos de las protecciones térmicas.

5.3. Trabajos Futuros

Considerando los resultados obtenidos y la validación del prototipo en un entorno controlado, se proponen las siguientes líneas de desarrollo para evolucionar el proyecto hacia un producto comercial:

5.3.1. Implementación en Empresa de Mantenimiento (Escalabilidad)

El siguiente paso lógico es la implementación de esta arquitectura en una empresa de servicios de climatización real. Se proyecta escalar el sistema para monitorear un edificio con 30 departamentos. Gracias a la arquitectura MQTT y el uso de un Broker en la nube (HiveMQ), el sistema es fácilmente escalable; solo se requiere replicar los nodos de lectura (ESP32) sin necesidad de modificar la infraestructura del servidor central.

5.3.2. Encapsulado y Grado de Protección Industrial (Clasificación IP)

Si bien en el prototipo actual se logró integrar con éxito instrumentación de clase industrial (sensor PT-100 y lazo de 4-20 mA), la circuitería de adquisición y procesamiento (ESP32, multiplexor CD74HC4067 y módulo step-down) aún se mantiene dispuesta sobre una placa de pruebas (Protoboard). Para su implementación definitiva en las salas de calderas y bombas, se requiere el diseño de una placa de circuito impreso (PCB) dedicada, seguida de su instalación al interior de un tablero eléctrico con certificación IP65. Esta medida garantizará la robustez mecánica y la inmunidad del sistema frente al polvo, la humedad ambiental y el ruido electromagnético.

5.3.3. Implementación de Control Bidireccional (Telecontrol)

Actualmente, el sistema es de monitoreo (lectura). Un trabajo futuro crucial es cerrar el lazo de control, permitiendo enviar comandos desde el Dashboard en la nube hacia el terreno. Por ejemplo, agregar un botón en la interfaz web que permite “Resetear” una alarma o “Detener” una bomba de emergencia remotamente, escribiendo en los registros del PLC o ESP32.

5.3.4. Algoritmos de Mantenimiento Predictivo

Con la base de datos histórica (InfluxDB) acumulando datos de 30 departamentos, se propone a futuro integrar algoritmos de Machine Learning en Node-RED. Esto permitiría predecir fallas antes de que ocurran (detectar la degradación del rendimiento de la bomba meses antes de que falle), ofreciendo un servicio de valor agregado premium a los clientes.

6. Anexos

6.1. Anexo A: ESPECIFICACIONES TÉCNICAS (DATASHEETS)

6.1.1. Microcontrolador ESP32 (Espressif Systems)

Hoja de datos técnica del módulo ESP-WROOM-32, utilizado como unidad central de procesamiento y comunicación Wi-Fi.



Figura 6.1: Portada Datasheet ESP32 Series.

6.1.2. Variador de Frecuencia Mitsubishi Electric Serie FR-F800

Especificaciones generales y diagrama de bloque del variador de frecuencia utilizado teóricamente para el control de bombas.



INVERTER FR-F800 INSTRUCTION MANUAL (DETAILED)

Inverter for fans and pumps

FR-F820-00046(0.75K) to 04750(110K)
FR-F840-00023(0.75K) to 06830(315K)
FR-F842-07700(355K) to 12120(560K)
FR-F846-00023(0.75K) to 03610(160K)

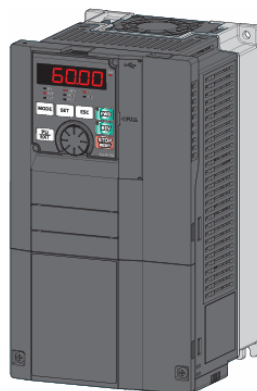


Figura 6.2: Portada Manual de Instrucciones Inverter FR-F800.

6.1.3. Módulo Conversor TTL a RS-485 (MAX485)

Circuito integrado para la conversión de niveles lógicos de la UART del ESP32 al estándar diferencial RS-485.

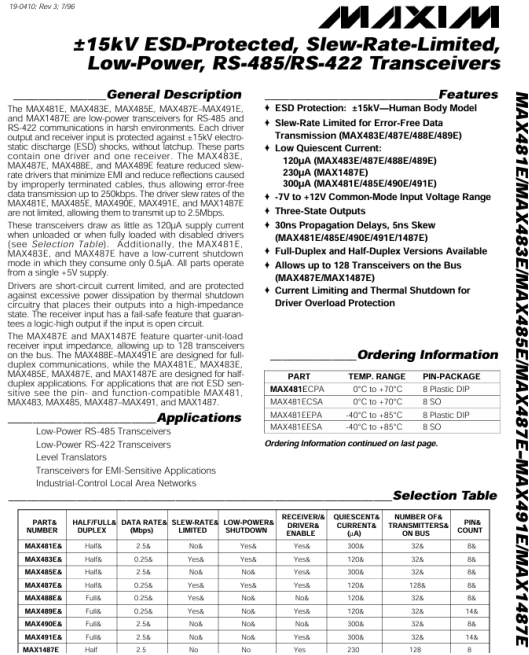


Figura 6.3: Portada Datasheet Maxim Integrated MAX485.

6.2. Anexo B: PLANOS Y ESQUEMAS DE CONEXIÓN

6.2.1. B.1. Diagrama de Conexión Prototipo (Maqueta)

Esquema de conexionado real utilizado durante las pruebas experimentales, incluyendo el sensor de temperatura y el módulo ESP32.

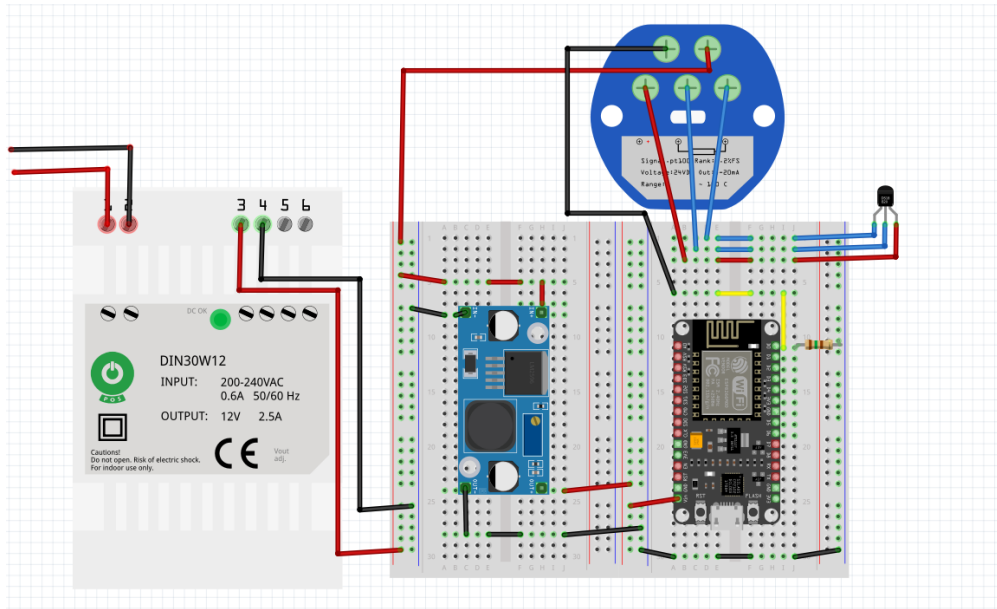


Figura 6.4: Conexionado, maqueta prototipo.

6.2.2. B.2. Esquema de Conexión Industrial Teórico (Modbus)

Diagrama proyectado para la integración industrial entre el ESP32, el módulo RS-485 y los terminales de comunicación del Variador Mitsubishi F800.

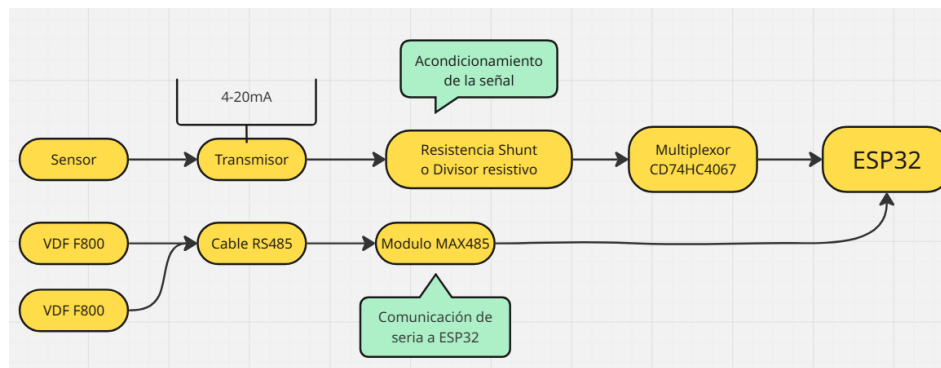


Figura 6.5: Diagrama de flujo de datos y conexión del sistema.

6.3. Bibliografía

- **Espressif Systems.** (2024). *ESP32 Series Datasheet: 2.4 GHz Wi-Fi + Bluetooth + Bluetooth LE SoC*. Documentación oficial del microcontrolador. https://documentation.espressif.com/esp32_datasheet_en.pdf
- **Mitsubishi Electric.** (2020). *Inverter FR-F800 Instruction Manual (Detailed)*. Manual técnico para configuración de parámetros y comunicación Modbus. https://usmcl-my.sharepoint.com/:b:/g/personal/ignacio_castilloh_usm_cl/IQAPs3v6JzRARrKKDe=oxq1Wf
- **HiveMQ.** (2025). *The Essentials of MQTT Protocol*. Guía fundamental sobre el funcionamiento del protocolo Pub/Sub utilizado en el proyecto. <https://www.hivemq.com/>
- **Node-RED.** (2024). *Node-RED Documentation: Core Nodes*. Documentación sobre los nodos de función, MQTT in/out y depuración. <https://nodered.org/docs/user-guide/nodes>
- **InfluxData.** (2024). *InfluxDB OSS 2.0 Documentation*. Guía para la escritura y consulta de datos en series temporales usando lenguaje Flux. <https://docs.influxdata.com/influxdb/v2/>
- **Grafana Labs.** (2024). *Grafana Visualization Guide*. Manual para la creación de Dashboards, Paneles de estado (Gauge) y Series de tiempo. <https://grafana.com/docs/grafana/latest/visualizations/>
- **Maxim Integrated (Analog Devices).** (2019). *MAX485/MAX490 Datasheet*. Especificaciones del transceptor RS-485 utilizado para la interfaz física. <https://www.analog.com/media/en/technical-documentation/data-sheets/MAX1487-MAX491.pdf>
- **Modbus.** <https://www.cursosaula21.com/modbus-que-es-y-como-funciona>
- **Espressif Systems.** (2024). *Wi-Fi API - Arduino-ESP32 Documentation*. Librería oficial para la gestión de conectividad inalámbrica y seguridad TLS/SSL en el microcontrolador ESP32. <https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/wifi.html>
- **O’Leary, N.** (2024). *PubSubClient API Documentation*. Librería cliente diseñada para mensajería simple de publicación/suscripción con un servidor compatible con MQTT. <https://pubsubclient.knolleary.net/api>
- **Blanchon, B.** (2024). *ArduinoJson: C++ JSON library for IoT*. Biblioteca eficiente para la serialización y deserialización de archivos JSON en sistemas embebidos con recursos limitados. <https://arduinojson.org/>
- **Walker, D. (4-20ma).** (2024). *ModbusMaster Library*. Librería de Arduino para la implementación del protocolo Modbus RTU (Maestro) a través de RS232/485. <https://github.com/4-20ma/ModbusMaster>

- **Espressif Systems.** (2024). *Arduino core for the ESP32*. Repositorio oficial del soporte de hardware y herramientas de desarrollo para la programación del ESP32 en Arduino IDE. <https://github.com/espressif/arduino-esp32>