Sistema IIoT para monitorización de la estación de radioastronomía CASIRI usando el protocolo MQTT Sparkplug B

Andrés Felipe Rubio Toloza

Trabajo de Grado para optar al título de Magíster en Ingeniería de Telecomunicaciones

Director

Dr. Juan Manuel Rey López

Co-Directores

Dr. Julián Rodríguez Ferreira

Dr. German Osma Pinto

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones

Bucaramanga

2025

Agradecimientos

El presente trabajo de grado de maestría fue desarrollado gracias al apoyo de la Universidad Industrial de Santander mediante el proyecto de investigación titulado “Plataforma IIoT para formación de profesionales en tecnologías de la cuarta revolución industrial” VIE-UIS 2824 y gracias al apoyo de MinCiencias mediante el proyecto de investigación titulado “Diseño, desarrollo e implementación de una plataforma IIoT para formación de profesionales en tecnologías de la cuarta revolución industrial” código 82273, contrato 719-2022.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Lista de Apéndices

pág.

Resumen

Título: Sistema IIoT para monitorización de la estación de radioastronomía CASIRI usando el protocolo MQTT Sparkplug B.[1]

Autor: Andrés Felipe Rubio Toloza [2]

Palabras Clave: IIoT, MQTT, Sparkplug B, Radioastronomía.

Descripción: Este trabajo de investigación presenta el diseño e implementación de un sistema de Internet Industrial de las Cosas (Industrial Internet of Things - IIoT) para la monitorización de la estación de sitios candidatos a radio observatorios CASIRI [3], desarrollado por el Grupo de Investigación en Control, Electrónica, Modelado y Simulación CEMOS de la Universidad Industrial de Santander (UIS). El sistema fue verificado de manera progresiva a través de prácticas experimentales, lo que permitió validar la funcionalidad y fiabilidad de los dispositivos y protocolos de comunicación utilizados, como el protocolo de transporte por telemetría de cola de mensajes (Message Queuing Telemetry Transport - MQTT) y Sparkplug B. Además, se integró exitosamente con el software Ignition, lo que garantiza una gestión eficiente y estructurada de los datos en tiempo real. El proyecto tiene un enfoque pedagógico, donde se demuestra el potencial del sistema diseñado para la enseñanza de Industria 4.0. A través de la creación de guías educativas, se facilita el aprendizaje de tecnologías clave como MQTT, Sparkplug B, y la plataforma Ignition, así como la operación de una estación de radioastronomía. Estas guías están diseñadas para ser herramientas prácticas y aplicadas, permitiendo que estudiantes y profesionales adquieran habilidades relevantes para la industria y la investigación.

---

[1] Trabajo de investigación de maestría en Ingeniería de Telecomunicaciones

[2] Facultad de Ingenierías Físicomecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Director: Juan Manuel Rey López. Co-directores: German Osma Pinto, Julián Rodríguez Ferreira

[3] CASIRI corresponde a CAracterización de SItios en Radio Interferencias

Abstract

Title: IIoT system for monitoring the CASIRI radio astronomy station using the MQTT Sparkplug B protocol[4]

Author: Andrés Felipe Rubio Toloza[5]

Keywords: IIoT, MQTT, Sparkplug B, Radioastronomy

Description: This research work presents the design and implementation of an Industrial Internet of Things (IIoT) system for monitoring candidate sites for radio observatories CASIRI [6] station developed by the Control, Electronics, Modelling and Simulation CEMOS Research Group from the Universidad Industrial de Santander (UIS). The system was progressively verified through experimental practices, allowing the validation of the functionality and reliability of the devices and communication protocols used, such as Message Queuing Telemetry Transport (MQTT) and Sparkplug B. Additionally, it was successfully integrated with Ignition software, ensuring efficient and structured real-time data management. Furthermore, the project has a pedagogical focus, demonstrating the system's potential for teaching Industry 4.0. The creation of educational guides facilitates the learning of critical technologies such as MQTT, Sparkplug B, and the Ignition platform, as well as the operation of a radio astronomy station. These guides are designed to be practical and applied tools, enabling students and professionals to acquire skills relevant to the industry and research.

---

[4] Master's thesis in Telecommunications Engineering

[5] Faculty of Physical-Mechanical Engineering. School of Electrical Engineering, Electronics and Telecommunications. Advisor: Juan Manuel Rey López. Co-advisors: German Osma Pinto, Julián Rodríguez Ferreira

[6] CASIRI corresponds to CAracterización de SItios en Radio Interferencias

1. Introducción

1.1. Introducción

En el contexto colombiano, la industria santandereana enfrenta dificultades para encontrar talento con todas las competencias necesarias para enfrentar los retos de la era moderna, especialmente, por el rápido avance de la tecnología industrial (Solano et al. (2021)). Ante esto, las universidades deben adaptar sus enfoques pedagógicos para satisfacer las demandas de la Industria 4.0, que implica la integración de tecnologías como la inteligencia artificial, el Internet de las Cosas (IoT) y la automatización en la producción. Desarrollar estas competencias es esencial para que los profesionales, especialmente los recién egresados, estén preparados para entrar en el mercado laboral nacional y ser líderes de las transformaciones que requiere un sector industrial en constante evolución.

La disponibilidad de la información para analizar, monitorear y controlar los procesos industriales es fundamental para la toma de decisiones efectivas que optimicen la producción y la adaptabilidad. Ante esta realidad, las plataformas IIoT emergen como alternativas viables debido a su adaptabilidad, disponibilidad continua, costos reducidos por usuario y su capacidad para integrarse fácilmente con otras tecnologías.

El Internet de las Cosas (Internet of Things, IoT) se refiere a la interconexión de dispositivos y sistemas a través de internet, permitiendo la comunicación y el intercambio de datos en tiempo real (Xu et al. (2014)). El IoT aplicado al ámbito industrial se denomina Internet Industrial de las Cosas (Industrial Internet of Things, IIoT), el cual se enfoca en la integración de senso-

res, dispositivos y sistemas en un entorno industrial para mejorar la eficiencia, productividad y seguridad (Piccialli et al. (2021)). Para abordar la problemática de búsqueda de talentos con competencias de alto nivel tecnológico que enfrenta la industria santandereana, se propone el desarrollo de una plataforma IIoT enfocada en tres casos de estudio, uno de los cuales está relacionado con la Radioastronomía, como se muestra en la Figura 1. Este caso, denominado CASIRI, incluye una estación de caracterización de sitios candidatos a radio observatorios equipada con una estación meteorológica, una cámara y un dispositivo USRP para capturar las señales de radio circundantes. El proyecto se enfoca específicamente en este caso de estudio.

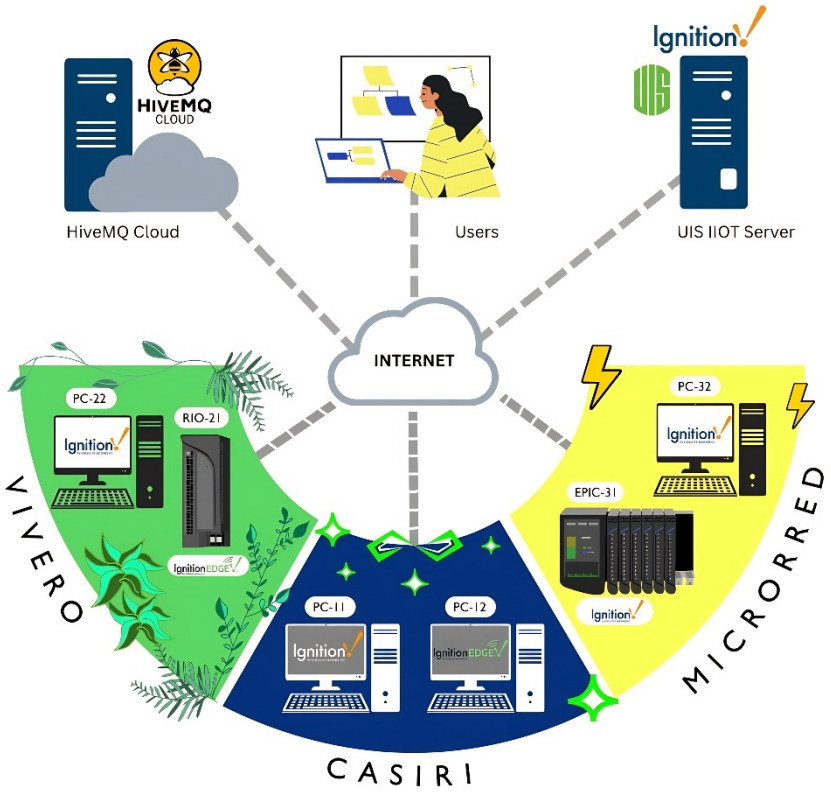


Figura 1. Plataforma IIoT basada en 3 casos de estudio

La Radioastronomía tiene como objetivo explorar el universo a partir de señales interestelares de radio. Para lograrlo, se necesita construir radio-observatorios en lugares con cielos silenciosos (Wilson et al. (2013)), donde las señales de radio de origen humano sean mínimas o inexistentes, y donde existan condiciones atmosféricas óptimas, como cielos despejados, altos, con pocas nubes y una humedad muy baja (Censier et al. (2021)).

Aunque en Colombia ya existen iniciativas de radiotelescopios, no hay un radio-observatorio profesional de alta montaña. A esto apunta la segunda fase del proyecto CASIRI, liderado por el grupo CEMOS y apoyado por la convocatoria 890 de Minciencias, que busca no solo la construcción de radiotelescopios profesionales, sino también avanzar en la creación de un radio-observatorio. Debido a esto, es crucial identificar y caracterizar con precisión los sitios óptimos para establecer el radio-observatorio, esencial para captar señales espaciales con mayor precisión y fortalecer la ciencia en el país.

Para identificar los lugares óptimos es fundamental contar con un sistema integrado que incluya sensores para medir variables atmosféricas, una cámara para evaluar la densidad de nubes y un receptor de radio para detectar señales. Sin embargo, dado que estos sitios suelen ser apartados y de difícil acceso, es necesario implementar un sistema que permita la visualización remota de los datos recolectados.

Una posible solución a esta necesidad es el desarrollo de un sistema IIoT, el cual posibilita la visualización remota de los datos medidos por CASIRI, que actualmente carece de esta capacidad. Este sistema proporciona una infraestructura robusta para monitorear en tiempo real las condiciones atmosféricas, la actividad celeste y las señales de radio desde cualquier ubicación. Al

permitir la recolección, procesamiento y visualización de datos de manera remota, se mejoran las capacidades de uso de CASIRI, pues puede reducirse la necesidad de desplazamientos físicos y se asegura la continuidad de la recolección de la información requerida.

De esta manera podemos identificar que el diseño del sistema IIoT requerido por CASIRI aborda dos enfoques: técnico (resolución de un problema específico relacionado con la estación CASIRI) y educativo (al ser CASIRI uno de los casos de estudio que serán usados en el marco de una plataforma IIoT de formación en competencias de la industria 4.0). El desafío principal radica en integrar estos enfoques en el diseño para resolver el problema específico de visualización remota de datos y garantizar que el sistema pueda ser aprovechado para realizar prácticas para la formación en industria 4.0.

1.2. Planteamiento del problema

Este proyecto surge como respuesta a un reto técnico concreto: poner en operación una estación de caracterización de sitios candidatos a radio observatorios, desde la cual es necesario recolectar y visualizar datos de forma remota, confiable y en tiempo real. Actualmente, la estación CASIRI está compuesta por una estación meteorológica, una cámara para observación del cielo y un receptor para detección de interferencia por radiofrecuencia (RFI).

Frente a esta necesidad, se requiere el diseño e implementación de un sistema IIoT que permita la adquisición, transmisión, almacenamiento y visualización de variables ambientales recolectadas por la estación meteorológica y la cámara para observación del cielo. Además, el sistema a diseñar debe permitir la integración futura del receptor de radio para la detección de interferencias por radiofrecuencia (RFI), que no ha sido considerado en el alcance de este proyecto. El sistema debe integrar tecnologías representativas del entorno industrial, como protocolos de comunicación, plataformas de visualización, servidores en la nube y bases de datos, orientadas a garantizar la escalabilidad y confiabilidad de la solución.

De forma complementaria, se busca que el desarrollo del sistema IIoT sirva como base para una plataforma educativa, en la cual se utilicen guías de aprendizaje prácticas y material audiovisual que demuestre y valide el correcto funcionamiento del sistema por etapas. Esta estrategia pedagógica no solo debe permitir verificar la funcionalidad del sistema implementado, sino que también debe facilitar la comprensión de conceptos técnicos complejos por parte de estudiantes, a través de actividades alineadas mediante guías formativas.

Así, el proyecto no solo busca resolver un problema técnico de monitoreo remoto en un entorno científico especializado, sino también consolidarse como una herramienta educativa adaptable y validada que fortalezca la formación en tecnologías asociadas a la Industria 4.0.

1.3. Objetivos

Objetivo general

Diseñar e implementar un sistema IIoT para monitorización de la estación de radioastronomía CASIRI, que pueda ser usado como caso de estudio en una plataforma educativa para la formación de profesionales en competencias de industria 4.0.

Objetivos específicos

1. Implementar el sistema IIoT usando la especificación Sparkplug del protocolo MQTT.
2. Usar el sistema IIoT como caso de estudio, integrándolo a una plataforma educativa para la formación de profesionales en competencias de industria 4.0.
3. Validar el funcionamiento del sistema mediante una práctica de laboratorio que utilice el software y hardware que compone el sistema IIoT.

1.4. Estructura del documento

Este trabajo de investigación está organizado de la siguiente manera:

- **Capítulo 1:** Presenta una contextualización de las necesidades por las cuales surge este trabajo de investigación. En este capítulo encontramos la introducción, la justificación y los objetivos del trabajo realizado.
- **Capítulo 2:** Expone el estado del arte de las plataformas IIoT usadas en el contexto de la formación. En este capítulo vemos los avances recientes en el campo de plataformas IIoT educativas.
- **Capítulo 3:** Exhibe los requerimientos del sistema IIoT. En este capítulo encontramos los retos a solucionar de cada uno de los interesados.
- **Capítulo 4:** Plantea el diseño del sistema IIoT. En este capítulo encontramos una descripción del software y el hardware, así como de su funcionamiento.
- **Capítulo 5:** Presenta la integración del sistema. En este capítulo se detalla cómo se interconectaron el hardware y el software para lograr el correcto funcionamiento del sistema.
- **Capítulo 6:** Muestra la verificación del sistema. En este capítulo se describen las pruebas utilizadas para la validación del correcto funcionamiento del sistema IIoT.
- **Capítulo 7:** Introduce brevemente las guías pedagógicas. En este capítulo se encuentra la introducción y las actividades a desarrollar en cada una de las guías.

- **Capítulo 8:** Presenta las conclusiones del proyecto y se identifican áreas clave para la continuación de investigaciones futuras.

2. Estado del arte

El acelerado desarrollo de las Tecnologías de la Información y Comunicación (TIC) ha incrementado significativamente la conectividad global, generando la necesidad de definir nuevos conceptos para referirse a la interconexión de objetos a través de Internet (Drath and Horch (2014)). Este fenómeno se denomina Internet de las Cosas (IoT, por sus siglas en inglés), un término introducido por Kevin Ashton. El IoT describe una red de objetos físicos que incorporan sensores, software y otras tecnologías para conectarse e intercambiar datos con otros dispositivos y sistemas a través de Internet (Liao et al. (2018)),(Dorsemaine et al. (2015)),(Xu et al. (2014)).

El Internet Industrial de las Cosas (IIoT) representa una evolución significativa del IoT, centrada en la autonomía de las máquinas dentro de entornos industriales. El IIoT integra sensores, instrumentos y dispositivos autónomos conectados a través de Internet para aplicaciones específicas en la industria (Mumtaz et al. (2017)). Su principal enfoque es la comunicación máquina a máquina (M2M) y las tecnologías de comunicación industrial, con el objetivo de aumentar la automatización y la autogestión de las máquinas industriales (Zhang et al. (2017)). Esta integración tecnológica permite una comprensión más profunda y detallada de los procesos de fabricación, resultando en una producción más eficiente y sostenible (Sisinni et al. (2018)).

La Figura 2 presenta una comparación entre el IoT y el IIoT, destacando algunas de las aplicaciones más comunes de cada uno.

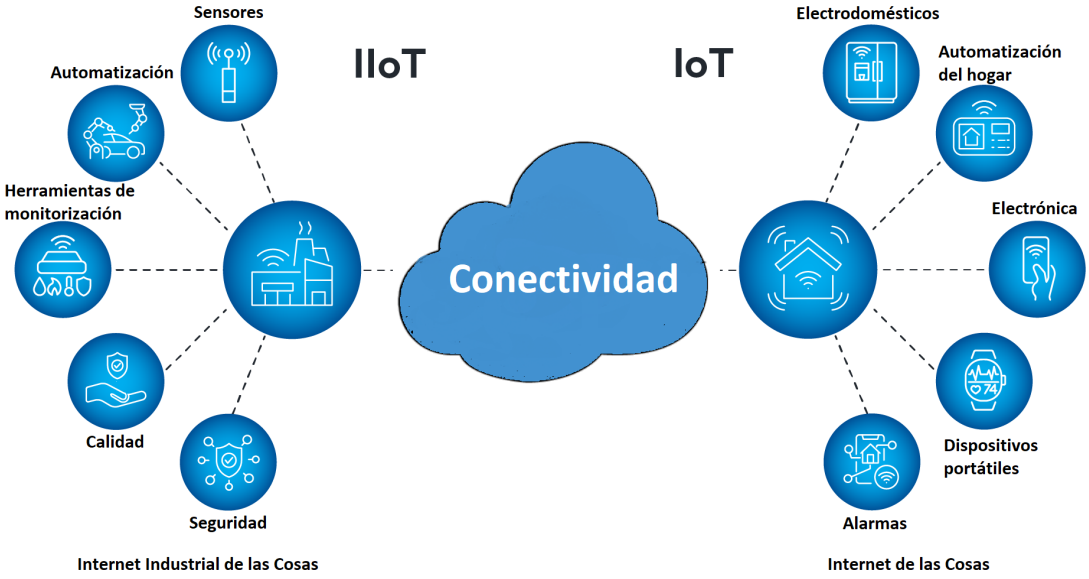


Figura 2. Diferencias entre IoT e IIoT. Adaptado de (Genetec (2024))

El IoT ha experimentado un uso creciente en una variedad cada vez mayor de aplicaciones. Las investigaciones abarcan desde el control remoto y el seguimiento por Global Position System (GPS) utilizando el protocolo MQTT (Aroon (2016)), hasta el diseño de variadores de frecuencia controlados remotamente basados en MQTT (Yang et al. (2021)), así como su implementación en ciudades inteligentes (Kaushik and Bagga (2021)) y estaciones meteorológicas (Mohapatra and Subudhi (2022)), (Amelia et al. (2020)). El protocolo MQTT utiliza una arquitectura de publicación-suscripción y permite comunicaciones múltiples entre dispositivos (Kaskatiiski and Boyanov (2021)). La relevancia de este protocolo ha aumentado significativamente debido a su escalabilidad, eficiencia en el uso de recursos, facilidad de implementación, además de que es muy ligero (Bender et al. (2021)). También destaca por su bajo consumo de energía y una distribución de información bien organizada para los receptores (Amjad et al. (2021)). La Figura 3 muestra la

arquitectura del protocolo MQTT. En este ejemplo, el cliente MQTT publica datos de temperatura (25º) que el servidor (broker) transmite a otros dispositivos suscritos, como una aplicación móvil o un cliente en una computadora.

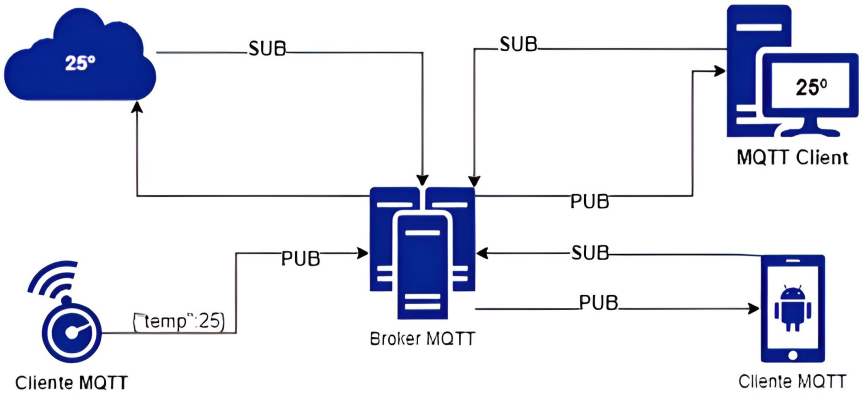


Figura 3. Arquitectura del protocolo MQTT (InfoPLC (2024))

En el ámbito educativo, el IoT ha encontrado aplicaciones significativas, ganando relevancia en los últimos años por su potencial para mejorar la enseñanza de tecnologías avanzadas (Embong et al. (2019)). Diversos proyectos y estudios han abordado la creación y aplicación de estas plataformas, proporcionando tanto infraestructura como contenido educativo adaptado a distintos niveles y contextos educativos (Cai et al. (2015))(Muhammad et al. (2018))(Rodríguez-Calderón and Belmonte-Izquierdo (2021)).

Uno de los enfoques más comunes es el desarrollo de sistemas de bajo costo que permitan la enseñanza práctica del IoT. Un ejemplo notable es un sistema que facilita la conexión remota de los estudiantes, enfocándose en sistemas de distribución CA/CC, lenguajes de programación como C y Python, y diseño de hardware y software (Cheng et al. (2020)).

Otra iniciativa destacada es la construcción de kits IoT basados en LoRa (RFID) y Rasp-

berry Pi, utilizados en diversos cursos en la Universidad de Ciencias de Hue, en Vietnam. Estos kits permiten la implementación de proyectos prácticos con metas de aprendizaje específicas, proporcionando a los estudiantes una experiencia práctica (tipo *hands-on*) esencial para la comprensión de las tecnologías IoT (Phan et al. (2021)).

La configuración de laboratorios a escala que emulan plantas industriales es otro método empleado para enseñar principios de IoT. Un ejemplo es la creación de una planta de Industria 4.0 a escala de laboratorio, que utiliza tecnologías avanzadas para replicar un entorno industrial realista (Krushnan and Schrödel (2022)).

Proyectos como Tiphys han desarrollado plataformas abiertas en red con el objetivo de facilitar el aprendizaje de temas relacionados con la Industria 4.0. Estas plataformas permiten a los estudiantes interactuar con tecnologías de vanguardia en un entorno colaborativo y accesible (Antonelli et al. (2019)).

La Universidad de Zhejiang en China implementó un curso MOOC (Massive Open Online Course) titulado "DIY Smart House", donde los estudiantes aprenden sobre IoT a través del control de una casa miniatura con un Arduino. Este curso de ocho semanas aborda temas como control de temperatura, riego y detección de fugas de gas (Shi et al. (2020)). [7]

El uso de sistemas de aprendizaje basados en tecnología es también una estrategia eficaz. Los autores en (Akbar et al. (2018)) diseñan un sistema para el laboratorio de control, utilizando un

---

[7] Disponible en la siguiente URL: `https://www.icourse163.org/course/ZJU-1206632831?from=searchPage&outVendor=zw_mooc_pcssjg_`. Consultado el 10/08/2024

controlador estándar industrial adecuado para la creación de prototipos de aplicaciones industriales y empresariales, destacando su bajo costo y accesibilidad.

Las plataformas basadas en la nube, como la propuesta que utiliza IBM Cloud junto con Raspberry Pi como dispositivo de frontera (edge device - hace referencia a un equipo que realiza procesamiento de datos cerca de la fuente de generación, reduciendo la necesidad de enviar toda la información a un servidor central o la nube), ofrecen experiencias prácticas con plataformas IoT reales. Estas experiencias proporcionan un entorno de aprendizaje más realista y aplicable (Nykyri et al. (2019)).

La plataforma IoT PEER (Plataforma de IoT para la Educación e Investigación en Ingeniería) es otro ejemplo de cómo las herramientas educativas pueden ser diseñadas y utilizadas. Esta plataforma muestra los parámetros de diseño y presenta casos de estudio específicos en los que se ha implementado, facilitando tanto la educación como la investigación en ingeniería (Guo et al. (2018)).

El desarrollo de laboratorios remotos de bajo costo también es una tendencia emergente. Un estudio describe la arquitectura de un sistema de laboratorio remoto tanto en hardware como en software, y presenta varios casos de estudio implementados a través de este sistema (Pirrone et al. (2021)).

El proyecto SUCRE (Sense yoUr Context and REact) se enfoca en fomentar la vocación por la ciencia mediante el IoT. Este proyecto describe tanto la parte de hardware (SucreCore) como la parte de software (SucreCode), comparando sus características con otras plataformas y presentando experimentos y montajes realizados con el sistema (Trilles et al. (2022)).

Un escenario innovador es el IoT aplicado a la agricultura, donde diversos elementos han sido acondicionados para enseñar sobre automatización y control, enfocándose principalmente en ingenieros agrícolas (Loukatos et al. (2022)).

En la enseñanza de microprocesadores, se ha implementado un laboratorio que incluye un módulo desde donde el docente puede monitorear la actividad del estudiante, observando el estado de algunos pines y la ejecución del código. Esta configuración mejora el control y seguimiento del aprendizaje de los estudiantes (Jacko et al. (2022)).

Finalmente, la plataforma Labs of Things at UNED (LoT@UNED) ha sido utilizada en la asignatura "Cloud Computing and Network Service Management" en la Universidad Nacional de Educación a Distancia en España. Esta plataforma facilita la enseñanza y la investigación, permitiendo a los estudiantes interactuar con entornos IoT en sus cursos (Pastor-Vargas et al. (2020)).

En la era de la cuarta revolución industrial, donde el IoT tiene un rol crucial, la adopción de tecnologías IIoT ha dejado de ser opcional para convertirse en una necesidad para las empresas que desean seguir siendo competitivas. La evolución del sector industrial exige la convergencia de sistemas físicos y digitales, generando procesos inteligentes, flexibles y eficientes. La academia, por lo tanto, debe alinearse con estas exigencias, proporcionando plataformas que permitan a los estudiantes desarrollar las habilidades prácticas necesarias para enfrentar los desafíos de la Industria 4.0.

Estas investigaciones reflejan algunos de los avances más recientes en el desarrollo de plataformas educativas basadas en IoT e IIoT, que han demostrado su eficacia en diversos campos. Estas plataformas han facilitado la adquisición de habilidades prácticas y el monitoreo del aprendi-

zaje de los estudiantes, estableciendo un vínculo sólido entre la teoría y su aplicación en el mundo real. En este contexto, la plataforma educativa IIoT propuesta en este proyecto se alinea con estas iniciativas, buscando dar respuesta a las necesidades específicas de formación en habilidades requeridas por la industria 4.0 de la región y del país, con el objetivo de generar un impacto positivo directo en los programas educativos de la UIS y la formación de profesionales de Santander.

El proyecto de creación de una plataforma educativa IIoT responde directamente a esta necesidad emergente. Su objetivo principal es construir una herramienta que le permita a los estudiantes de la UIS desarrollar competencias técnicas para diseñar, implementar y operar sistemas IIoT. Actualmente, aunque los programas de Ingeniería Eléctrica y Electrónica abordan conceptos clave de tecnologías emergentes, no cuentan con una plataforma de este tipo que permita aplicar estos conceptos en un entorno práctico e integrado. Este proyecto pretende llenar ese vacío, ofreciendo un entorno donde los estudiantes trabajen con sensores conectados, dispositivos inteligentes y tecnologías de comunicación industrial.

El impacto de esta plataforma no se limitará solo a los programas mencionados; también se puede extender a otras disciplinas afines, como Ingeniería de Sistemas, permitiendo el desarrollo de proyectos multidisciplinarios y colaborativos. Además, al adaptar sus contenidos a las demandas actuales del mercado laboral regional, la plataforma tendrá el potencial de ser usada como herramienta para la formación de profesionales externos a la universidad a través de posibles programas de educación continua (extensión).

3. Requerimientos del sistema

Para establecer los requerimientos necesarios en el diseño del sistema IIoT de la estación de radioastronomía CASIRI, es fundamental llevar a cabo una serie de procesos preliminares que permitan comprender plenamente el entorno en el cual se implementarán estos desarrollos. En este contexto, este capítulo se enfoca en describir la estación CASIRI, detallar las particularidades de su operación y, finalmente, presentar las necesidades y requerimientos indicados por los principales usuarios del sistema IIoT a desarrollar.

3.1. Contexto y objetivo de la estación CASIRI

Colombia está en camino de integrar su primera infraestructura dedicada a la radioastronomía, con el desarrollo del proyecto CASIRI. Este esfuerzo es liderado por la UIS y el Grupo de Investigación CEMOS, y tiene como objetivo identificar y caracterizar ubicaciones óptimas para la instalación de radio observatorios en el país. La radioastronomía permite estudiar el universo a través de señales de radio provenientes de cuerpos celestes y eventos cósmicos, complementando la astronomía óptica tradicional.

CASIRI se origina gracias al proyecto “Estudio para proponer recomendaciones de política para la gestión de espectro en servicios científicos enfocados en radioastronomía” (2019-2020), financiado por la Agencia Nacional del Espectro (ANE) y liderado por el profesor Julián Rodríguez Ferreira. También se fundamenta en el proyecto “Estudio RFI entre la banda de 50 a 250 MHz para evaluar la calidad del cielo en el Páramo de Berlín (Santander) como observatorio de bajas frecuencias para aplicaciones en radioastronomía y eventos en la alta atmósfera” (2019-2020),

cuyo investigador principal fue el profesor Óscar Reyes. Una versión adaptada de este trabajo se implementó en el proyecto "Caracterización de emisiones electromagnéticas de interferencia de radiofrecuencias y pruebas de un radiotelescopio (100 MHz) como insumos para la validación del sitio de montaje de una base radioastronómica en la Península Antártica", en el marco de las IX y X expediciones científicas colombianas a la Antártida del Programa Antártico Colombiano en 2023 y 2024, también liderado por Julián Rodríguez Ferreira.

Los resultados del desarrollo del sistema IIoT que aborda este proyecto serán incluidos en la XI expedición científica a la Antártida y en el proyecto "Desarrollo de un arreglo interferométrico de radiotelescopios para establecer una estación de radioastronomía de la UIS en el Páramo de Berlín (Santander)", dirigido también por Julián Rodríguez Ferreira en el marco de la convocatoria 890 de Minciencias, así como en el proyecto liderado por el profesor Juan Manuel Rey, "Diseño, desarrollo e implementación de una plataforma IIoT para formación de profesionales en tecnologías de la cuarta revolución industrial", financiado en la misma convocatoria.

3.2. Características de la estación CASIRI

CASIRI es un laboratorio móvil diseñado para ser transportado fácilmente mediante un remolque. Está conformado por:

1. Antenas y Rotor: Equipadas para captar señales de radio de servicios comerciales en diversas direcciones.

2. Cámara de Cielo Completo: Capta imágenes del cielo las 24 horas.

3. Estación Meteorológica: Monitorea variables atmosféricas como temperatura, presión, hu-

medad, velocidad y dirección del viento.

Además, cuenta con una versión portátil para realizar sondeos rápidos en sitios potenciales. Las dos versiones de la estación CASIRI se muestran en la Figura 4. La versión portátil incluye una estación meteorológica Vantage Pro 2 de la marca DAVIS Instruments y una cámara Oculus All Sky de la marca Starlight Xpress Inc. Esta estación portátil fue la utilizada en el desarrollo de este proyecto. La Figura 5 muestra un diagrama esquemático del funcionamiento de la estación.

(a) Remolque estación CASIRI (b) Versión laboratorio CASIRI

Figura 4. Remolque de la estación CASIRI y versión de pruebas de laboratorio

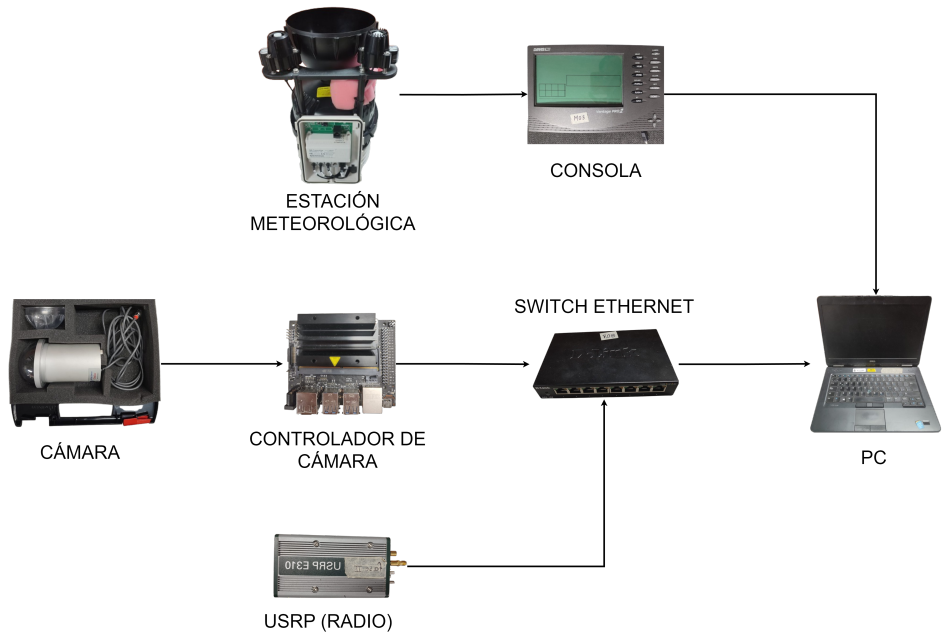


Figura 5. Diagrama de la estación CASIRI

3.3. Funcionamiento y proceso de selección de sitios

El proceso para determinar la viabilidad de un sitio como radio observatorio sigue estos pasos:

1. Sondeo Inicial: Utilizando la versión portátil del sistema, se realiza un primer análisis rápido del sitio. Este análisis incluye la medición preliminar de las condiciones atmosféricas y la identificación de posibles interferencias de señales de radio comerciales, permitiendo descartar rápidamente sitios no aptos.

2. Instalación Completa: Si el sitio cumple con los requisitos básicos, se instala el sistema completo en un tráiler y se deja recolectando datos durante varios días. Estos datos incluyen mediciones de interferencias de radiofrecuencia (RFI), condiciones meteorológicas como

temperatura, presión, humedad, velocidad y dirección del viento, y capturas de imágenes del cielo.

3. Análisis de Datos: Los datos son analizados para determinar si un sitio cumple con las condiciones óptimas. Un aspecto fundamental es la evaluación de los "cielos silenciosos", es decir, áreas con niveles mínimos de interferencia de radiofrecuencia (RFI). Esta interferencia, medida en dBm, representa el ruido producido por señales de origen humano. Para garantizar la detección de señales astronómicas débiles, se establece un umbral máximo de ruido, generalmente en torno a -100 dBm.

3.4. Travesía científica y desafíos

Previo a la pandemia del COVID-19, el equipo de CASIRI realizó expediciones a potenciales sitios como el Páramo de Santurbán, el Parque Nacional Nevado del Cocuy y la Laguna de Tota. Estas expediciones implicaron desafíos logísticos y físicos significativos, desde condiciones meteorológicas extremas hasta la necesidad de transportar equipos a través de terrenos difíciles. Durante la pandemia las actividades de campo fueron suspendidas temporalmente. No obstante, desde que se superó la emergencia sanitaria, el proyecto ha realizado dos expediciones a la Antártida.

3.5. Colaboración y apoyo

El proyecto CASIRI es un esfuerzo colaborativo que involucra a varios actores:

- Profesores y Grupos de Investigación: El proyecto es dirigido por el profesor Julián Rodríguez, y cuenta con el apoyo de profesores y grupos de investigación de la Escuela de

Ingenierías Eléctrica, Electrónica y de Telecomunicaciones (E3T) de la UIS, incluyendo a Homero Ortega Boada, Óscar Reyes, Edison Andrés Soto Ríos, Efrén Acevedo, y Germán Chaparro Molano de la Universidad de Antioquia.

- Colaboración Internacional: Científicos como Ricardo Bustos de la Universidad Católica de la Santísima Concepción de Chile, Oscar Restrepo de la Universidad de Chile, y Joan Montanyà de la Universidad Politécnica de Catalunya también participan en el proyecto.

- Apoyo Institucional y Logístico: La Agencia Nacional del Espectro y la Universidad Industrial de Santander, a través de la Vicerrectoría de Investigación y Extensión, han brindado recursos y apoyo logístico cruciales para el desarrollo del proyecto.

3.6. Impacto y futuro

El éxito de CASIRI posicionaría a Colombia como un nuevo actor en el campo de la radioastronomía, proporcionando una herramienta esencial para la observación y estudio del universo. La identificación de un "cielo silencioso" permitiría la construcción del primer observatorio radioastronómico profesional del país, abriendo nuevas oportunidades para la investigación científica y la formación de futuros astrónomos en Colombia.

3.7. Identificación de necesidades

Una vez comprendido el funcionamiento de la estación de radioastronomía CASIRI, junto con sus componentes y el estado actual de su implementación, se llevaron a cabo reuniones con cada uno de los interesados en la ejecución del proyecto. Estas reuniones tuvieron como objetivo recopilar los requerimientos necesarios para el diseño del sistema IIoT. A continuación, se

presentan los requerimientos identificados durante este proceso.

3.7.1. Equipo de trabajo CEMOS. El grupo de investigación CEMOS, liderado por el profesor Julián Rodríguez, quien también dirige la iniciativa de CASIRI, ha identificado una serie de necesidades específicas para el funcionamiento y optimización del sistema. Entre estas, destaca la capacidad de enviar y visualizar de forma remota los datos recolectados por la estación CASIRI, incluyendo la información de la estación meteorológica y las imágenes captadas por la cámara de cielo completo. Para satisfacer esta necesidad, es fundamental desarrollar una interfaz gráfica intuitiva y segura que permita a los investigadores acceder a los datos recolectados desde cualquier ubicación.

3.7.2. GISEL. El grupo de investigación GISEL está trabajando en un proyecto financiado por la convocatoria 890 del Ministerio de Ciencia, Tecnología e Innovación. El objetivo principal de este proyecto es la creación de una plataforma IIoT destinada a la formación de profesionales en la industria 4.0. En este contexto, la estación CASIRI ha sido seleccionada como un caso de estudio. El grupo de trabajo de este proyecto ha identificado una serie de temáticas clave sobre las cuales se deben realizar unas prácticas pedagógicas con el sistema IIoT que se va a implementar. Estas temáticas, MQTT, Ignition 8.1 y Sparkplug B, son fundamentales para el desarrollo y la operación efectiva de la plataforma. Estas necesidades identificadas deben ser consideradas con atención durante todo el proceso de diseño y desarrollo del sistema IIoT para garantizar su funcionalidad y efectividad en la formación de profesionales en la industria 4.0.

3.7.3. DAUTOM S.A.S. Como parte del equipo de trabajo para el desarrollo de la plataforma de enseñanza sobre IIoT mencionada anteriormente, DAUTOM S.A.S. plantea una

serie de requerimientos que deben ser considerados en el diseño del sistema IIoT. Esta empresa proporcionará asesoría en la selección de dispositivos y software adecuados, enfocándose en la utilización de elementos utilizados en la industria nacional. Esto permitirá posteriormente abordar temáticas de actualidad en el país dentro del ámbito de la enseñanza de IIoT.

3.7.4. Tabla de requerimientos. Después de concluir las reuniones con cada uno de los interesados en el desarrollo del proyecto y de identificar sus intereses y necesidades, se elaboró la Tabla 1. Esta tabla presenta a los interesados y los requerimientos específicos que debe cumplir el sistema IIoT para satisfacer sus necesidades.

Interesado	**Requerimiento**
CEMOS	Enviar y visualizar de forma remota los datos recolectados por la estación CASIRI.
CEMOS	Desarrollar una interfaz gráfica intuitiva y segura para acceder a los datos recolectados desde cualquier ubicación.
GISEL	Implementar prácticas pedagógicas con el sistema IIoT basadas en MQTT, Ignition 8.1, y Sparkplug B.
DAUTOM	Utilizar elementos (hardware y software) empleados en la industria nacional para la enseñanza de temáticas actuales en IIoT.

Tabla 1
Tabla de interesados y requerimientos

4. Diseño del sistema IIoT

El diseño del sistema IIoT para la estación CASIRI se realizó partiendo de la infraestructura existente, que ya incluía la estación meteorológica Davis, el controlador Jetson Nano, la cámara Oculus All-Sky, el switch Ethernet y el radio Ettus E310. El principal desafío del proyecto fue diseñar e integrar un sistema IIoT capaz de adquirir, gestionar y visualizar de manera remota los datos generados por estos dispositivos, ampliando sus capacidades. La Figura 6 presenta un diagrama de bloques que ilustra la infraestructura inicial de CASIRI y su integración con el sistema IIoT.

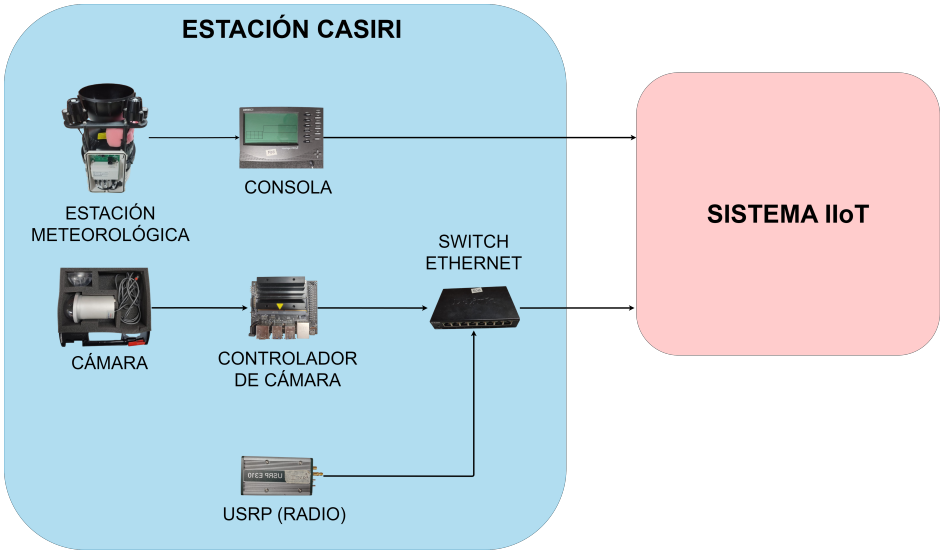


Figura 6. Infraestructura inicial de CASIRI e integración a el sistema IIoT

A continuación, se presenta una breve descripción de los componentes clave que conforman la estación CASIRI.

- **Estación meteorológica Davis:** Esta estación proporciona mediciones clave de las condi-

ciones climáticas, como temperatura, humedad, presión atmosférica y velocidad del viento. Su robustez y precisión en la recolección de datos la hacen una opción ideal para operar en condiciones climáticas adversas como las que enfrenta CASIRI.

- **Jetson Nano:** El controlador Jetson Nano es una computadora de desarrollo con capacidades avanzadas de procesamiento en tiempo real, esencial para gestionar y procesar las imágenes capturadas por la cámara Oculus All-Sky. Su bajo consumo energético y la posibilidad de ejecutar algoritmos de inteligencia artificial lo convierten en una pieza fundamental para el análisis de datos en entornos exigentes.

- **Cámara Oculus All-Sky:** Esta cámara captura imágenes del cielo en 360 grados, proporcionando una visión completa de las condiciones atmosféricas en todo momento.

- **USRP Ettus E310 (Radio):** El USRP Ettus E310 destaca por su capacidad para operar eficientemente en entornos remotos, gracias a su bajo consumo energético. Además, ofrece una gran versatilidad para analizar señales de radiofrecuencia en un amplio rango de frecuencias, lo que lo convierte en una herramienta fundamental para la detección y análisis de interferencias de radiofrecuencia (RFI).

El diseño del sistema IIoT de CASIRI se centró en integrar los datos recolectados por los dispositivos de monitoreo de la estación, con el fin de crear una infraestructura que permita la visualización remota y la gestión de datos en tiempo real.

Para lograr esto de manera eficiente, se seleccionó la arquitectura basada en el protocolo MQTT, reconocido por su capacidad de operar de forma eficiente en entornos con ancho de banda

limitado y con baja latencia. MQTT emplea un modelo de publicación-suscripción, lo que facilita la transmisión de datos desde los sensores y dispositivos de CASIRI hacia un servidor central, optimizando así el flujo de información en un sistema distribuido.

Esta arquitectura es particularmente adecuada para CASIRI, dado que el monitoreo continuo de las condiciones meteorológicas y la captura de imágenes del cielo requieren una comunicación fiable y sin interrupciones, incluso en ubicaciones remotas. Además, la arquitectura MQTT destaca por su escalabilidad y bajo consumo de recursos, superando a otras alternativas como HTTP, que resultan menos eficientes en cuanto a consumo energético y tiempo de respuesta. La Figura 7 presenta la arquitectura del sistema IIoT integrado a CASIRI.

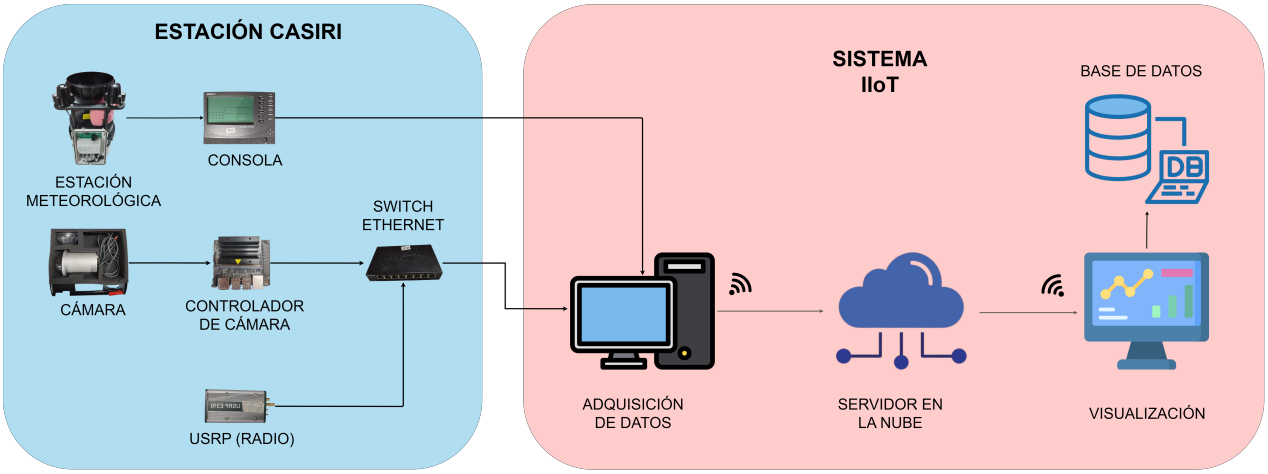


Figura 7. Arquitectura del sistema IIoT integrado a CASIRI

Para implementar este sistema, se seleccionaron componentes clave en tres categorías principales: hardware, software y comunicaciones. Estos elementos fueron elegidos para asegurar la interoperabilidad y el rendimiento del sistema en su conjunto. A continuación, se describen en

detalle cada uno de los componentes que conforman el sistema. Es importante mencionar que el subsistema relacionado con el análisis de interferencias de radiofrecuencia (RFI), mediante el USRP, no se incluyó en el alcance de este proyecto.

4.1. Hardware

4.1.1. PC OnLogic ML100G-40. Este computador industrial fue seleccionado para gestionar el sistema IIoT debido a sus características técnicas, evaluadas en conjunto con el equipo de DAUTOM para cumplir con los requisitos del proyecto. La Tabla 2 detalla sus especificaciones clave.

Categoría	Descripción
Procesador	Intel® Core™ i3-6100U / i5-6300U: El ML100G-40 está disponible con procesadores Intel® Core™ de 6ª generación, ofreciendo un equilibrio entre rendimiento y eficiencia energética. Arquitectura Skylake: Basado en la arquitectura Skylake de Intel, proporciona capacidades avanzadas de procesamiento y gráficos.
RAM	Capacidad de RAM: Soporta hasta 32 GB de memoria DDR4, ofreciendo un amplio margen para aplicaciones que requieren un procesamiento intensivo de datos. Ranuras SODIMM: Utiliza ranuras SODIMM de 260 pines, permitiendo la expansión de memoria según las necesidades del usuario.
Almacenamiento	Opciones de Almacenamiento: Equipado con una variedad de opciones de almacenamiento, incluyendo unidades SSD M.2 y 2.5"SATA, proporcionando alta velocidad de lectura/escritura y fiabilidad. Capacidad de Expansión: Ofrece la posibilidad de ampliar la capacidad de almacenamiento según las demandas específicas del proyecto.
Conectividad	Puertos USB: Dispone de múltiples puertos USB 3.0 y USB 2.0 para la conexión de periféricos y dispositivos externos. Ethernet: Equipado con dos puertos Gigabit Ethernet, facilitando la conectividad de red redundante y segura. Wi-Fi y Bluetooth: Opcionalmente, puede incluir módulos de conectividad inalámbrica Wi-Fi y Bluetooth.
Video y Audio	Salidas de Video: Proporciona salidas HDMI y DisplayPort, soportando configuraciones de monitores múltiples y resoluciones de hasta 4K. Audio: Incluye una salida de audio de alta definición para aplicaciones que requieren capacidades de sonido.
Sistema Operativo	Compatibilidad: Compatible con una variedad de sistemas operativos, incluyendo Windows 10, Linux y otros sistemas operativos embebidos. Configuraciones Personalizadas: Puede ser preconfigurado con el sistema operativo y software específico según los requisitos del cliente.
Diseño y Construcción	Chasis Sin Ventiladores: El diseño sin ventiladores reduce el ruido y el mantenimiento, aumentando la fiabilidad en entornos industriales. Construcción Robusta: Fabricado con materiales duraderos y de alta calidad, ideal para operar en condiciones ambientales adversas. Dimensiones Compactas: Su tamaño compacto permite una fácil integración en espacios reducidos y gabinetes industriales.

Tabla 2
Características técnicas del PC OnLogic

Ventajas de utilizar el PC OnLogic ML100G-40

1. **Robustez y Durabilidad**: El PC OnLogic ML100G-40 está diseñado para operar en entornos industriales y condiciones ambientales adversas. Su construcción robusta y sin ventilador lo hace resistente al polvo, a las vibraciones y a las temperaturas extremas, lo que es esencial para el proyecto. Este proyecto implica la instalación y operación de equipos en ubicaciones remotas y desafiantes, donde la fiabilidad y durabilidad del hardware son críticas.

2. **Tamaño Compacto y Montaje Flexible**: El diseño compacto del ML100G-40 facilita su instalación en espacios reducidos y en configuraciones móviles. Su flexibilidad de montaje permite adaptarse a diferentes entornos y configuraciones, asegurando que pueda integrarse fácilmente en la infraestructura existente sin requerir modificaciones significativas.

3. **Capacidades de Procesamiento Potente**: Equipado con procesadores Intel de bajo consumo pero alta eficiencia, el ML100G-40 ofrece un rendimiento de procesamiento robusto para manejar las tareas intensivas de datos necesarias en el proyecto. Esto incluye la recopilación, almacenamiento, procesamiento y análisis de grandes volúmenes de datos, asegurando un rendimiento fiable y eficiente en todo momento.

4. **Conectividad Avanzada**: El PC OnLogic ML100G-40 proporciona una amplia gama de opciones de conectividad, incluyendo múltiples puertos USB, Ethernet y opciones de comunicación inalámbrica. Esta conectividad avanzada es vital para integrar diversos dispositivos y sensores del sistema IIoT del proyecto, permitiendo una comunicación fluida y eficiente entre los componentes del sistema y facilitando el acceso remoto a los datos recolectados.

5. **Capacidad de Almacenamiento y Expansión**: El ML100G-40 ofrece opciones de almacenamiento flexible y expansión, lo que permite adaptarse a las crecientes necesidades de almacenamiento de datos del proyecto. La capacidad de agregar discos duros adicionales o módulos de memoria asegura que el sistema pueda crecer y adaptarse a medida que aumentan las demandas de datos, proporcionando una solución escalable y duradera.

6. **Bajo Consumo de Energía**: El diseño eficiente en términos de energía del ML100G-40 es particularmente importante para el proyecto, donde la operación en ubicaciones remotas puede limitar el acceso a fuentes de energía constantes. El bajo consumo de energía asegura que el sistema pueda operar de manera eficiente durante largos períodos, incluso en condiciones de suministro de energía limitadas, optimizando la sostenibilidad del proyecto.

7. **Fiabilidad y Mantenimiento Reducido**: La construcción sin ventilador del ML100G-40 reduce significativamente la necesidad de mantenimiento, minimizando los puntos de falla potenciales y aumentando la fiabilidad del sistema. Esto es crucial para el proyecto, que requiere un funcionamiento continuo y sin interrupciones en entornos difíciles, donde el acceso para el mantenimiento puede ser limitado.

(a) Vista frontal (b) Vista posterior

Figura 8. PC OnLogic ML100G-40

4.2. Software

4.2.1. Visual Studio Code & Python. La implementación del sistema IIoT de la estación CASIRI se llevó a cabo utilizando Visual Studio Code y Python debido a sus características y ventajas específicas. Visual Studio Code, con su interfaz intuitiva y extensibilidad, permitió un desarrollo ágil y eficiente del proyecto. Python, por su parte, fue seleccionado por su extensa variedad de librerías preexistentes para la transmisión de datos mediante MQTT y la integración del protocolo Sparkplug B, sobre el cual se presenta más información en las siguientes secciones. Estas librerías facilitaron la implementación de las funcionalidades necesarias sin necesidad de desarrollar soluciones desde cero, optimizando tanto el tiempo como los recursos del proyecto. En la Figura 9 se observa la interfaz gráfica de Visual Studio Code utilizando Python.

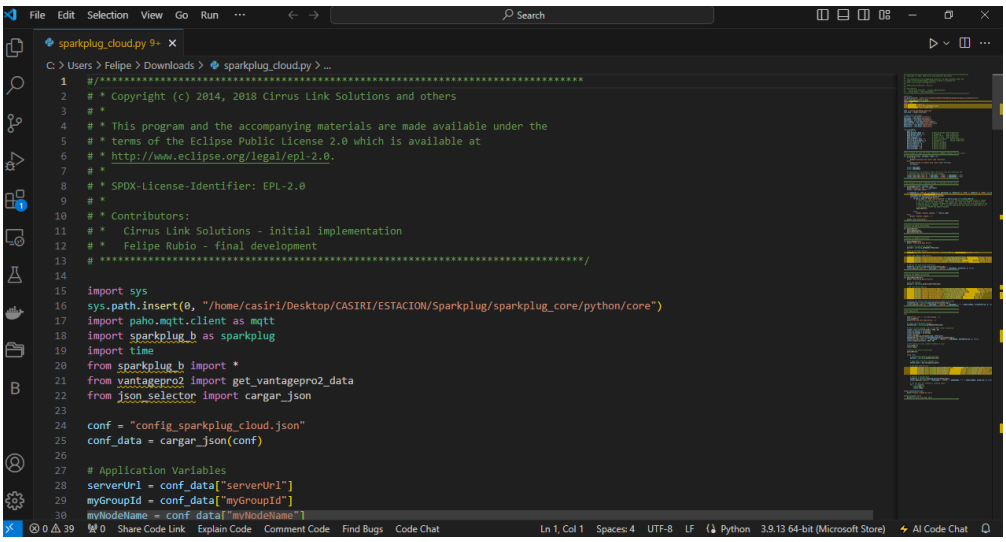


Figura 9. VS Code & Python

4.2.2. Ignition 8.1. La implementación del sistema IIoT de la estación CASIRI requiere un entorno robusto y ampliamente utilizado en la industria para automatización, control y

monitoreo. Se seleccionó Ignition 8.1 de Inductive Automation debido a sus numerosas ventajas. Este software es utilizado por grandes empresas como Coca-Cola, Shell, Johnson & Johnson, Starbucks, entre otros, y DAUTOM S.A.S actúa como integrador en Colombia, garantizando soporte local (Automation (2024b)). Ignition 8.1 permite la creación ilimitada de clientes, pantallas y variables, y soporta lenguajes como SQL, Python y el protocolo de comunicación MQTT, esenciales para la industria 4.0. Su capacidad para integrarse con diversas plataformas y dispositivos facilita la interoperabilidad y escalabilidad del sistema. Además, Ignition 8.1 proporciona acceso a recursos educativos como “Inductive University”, que ofrece módulos detallados para comprender y maximizar el uso de sus componentes (Automation (2024a)). Su arquitectura web-based permite una administración centralizada y acceso remoto seguro, lo cual es crucial para el monitoreo y control continuo de la estación CASIRI. La Figura 10 muestra la interfaz gráfica del software Ignition de Inductive Automation.

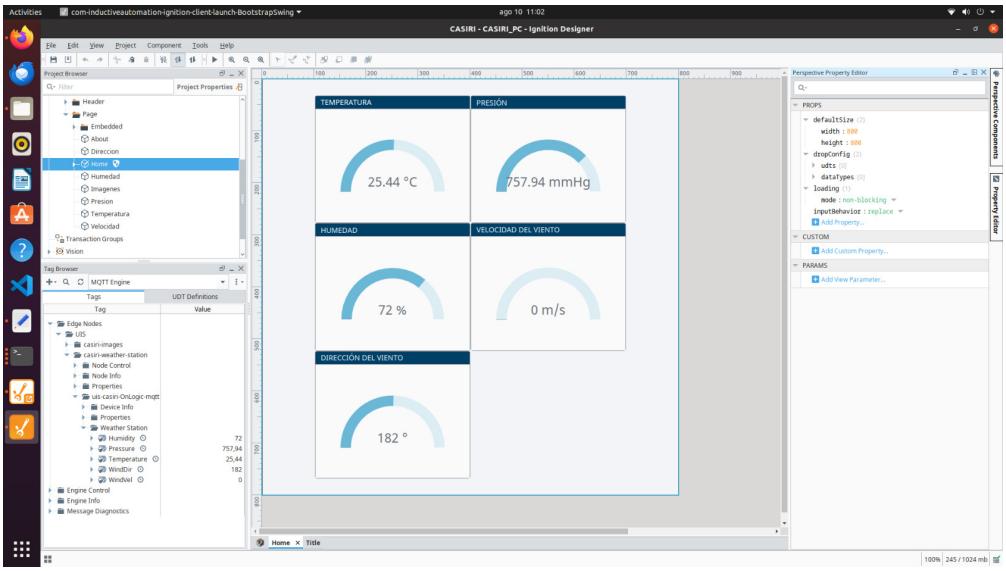


Figura 10. Software Ignition

4.2.3. MySQL. Para el manejo de bases de datos en la implementación del sistema IIoT de la estación CASIRI, se seleccionó MySQL debido a su robustez, escalabilidad y compatibilidad con Ignition. MySQL es un sistema de gestión de bases de datos relacional de código abierto ampliamente utilizado en la industria, lo que garantiza estabilidad y soporte continuo. Su integración con Ignition permite el almacenamiento eficiente y la gestión de grandes volúmenes de datos generados por el sistema IIoT. Además, MySQL ofrece un rendimiento sólido en consultas y transacciones, facilitando el análisis de datos en tiempo real y la generación de informes. Su capacidad para manejar múltiples usuarios simultáneamente y su flexibilidad en la configuración aseguran que pueda adaptarse a las necesidades específicas del proyecto CASIRI, optimizando la administración y acceso a la información crítica. La Figura 11 muestra la interfaz gráfica de MySQL Workbench, una herramienta visual para bases de datos.

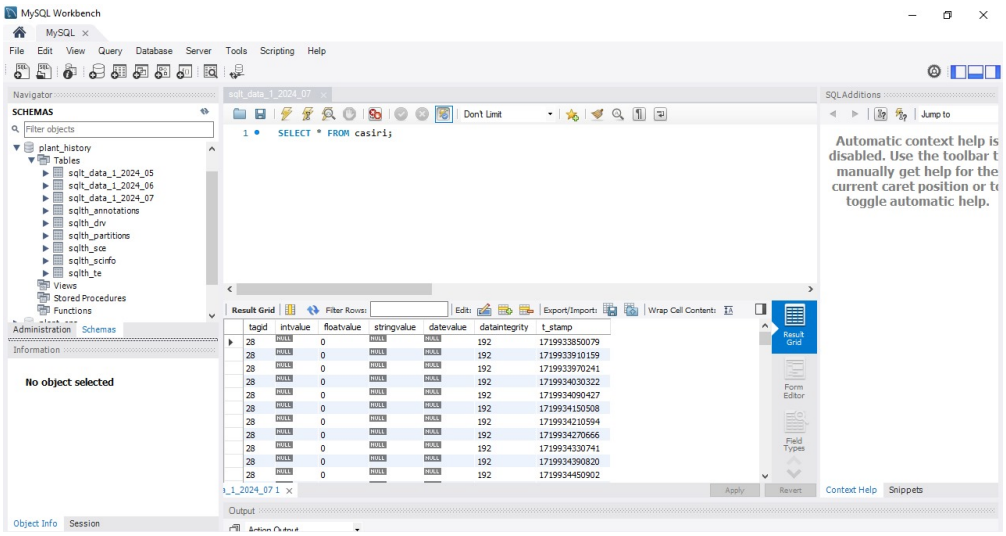


Figura 11. MySQL Workbench

4.3. Sistema de comunicaciones

El sistema de comunicaciones se basa en una arquitectura robusta y eficiente para la transmisión y monitoreo de datos. La estación meteorológica Davis Vantage Pro 2 se comunica mediante conexión serial con el PC principal. Simultáneamente, la cámara Oculus All Sky se conecta por serial a la Jetson Nano, que actúa como su controlador. La Jetson Nano se comunica por Ethernet con el PC principal, asegurando una transmisión de datos rápida y fiable.

Los datos capturados por ambos subsistemas, la estación meteorológica y la cámara, son recolectados por el PC principal. Posteriormente, estos datos se envían a través de MQTT utilizando la especificación Sparkplug B. Este protocolo garantiza que cualquier cliente, en cualquier ubicación, pueda acceder a los datos en tiempo real. La plataforma HiveMQ Cloud facilita la gestión de la mensajería MQTT, asegurando la fiabilidad y escalabilidad del sistema de comunicaciones. Así, los usuarios pueden visualizar los datos de forma remota y segura. La Figura 12 muestra la arquitectura para el sistema de comunicaciones.

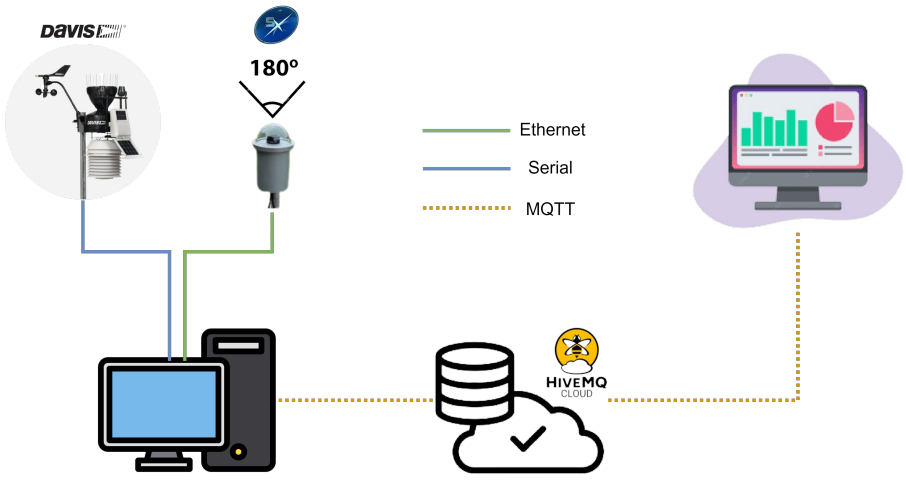


Figura 12. Diagrama de comunicaciones del sistema IIoT

4.3.1. MQTT. El protocolo MQTT (Message Queuing Telemetry Transport) es un protocolo de mensajería ligero y eficiente, diseñado específicamente para la comunicación de dispositivos en redes con recursos limitados y conexiones inestables. Utiliza un modelo de publicación/suscripción, donde los dispositivos (publicadores) envían mensajes a un servidor central denominado servidor MQTT. Los clientes interesados (suscriptores) pueden recibir estos mensajes a través del servidor, lo que permite una comunicación flexible y escalable entre múltiples dispositivos y sistemas.

En el sistema de comunicaciones de CASIRI, el uso de MQTT es fundamental para garantizar la transmisión de datos en tiempo real entre los diferentes componentes del sistema. La Figura 13 ilustra cómo Ignition, utilizado para la visualización, interactúa con un servidor MQTT central. Este servidor actúa como intermediario entre los dispositivos que capturan los datos (como la estación meteorológica y la cámara) y los sistemas que los procesan y visualizan.

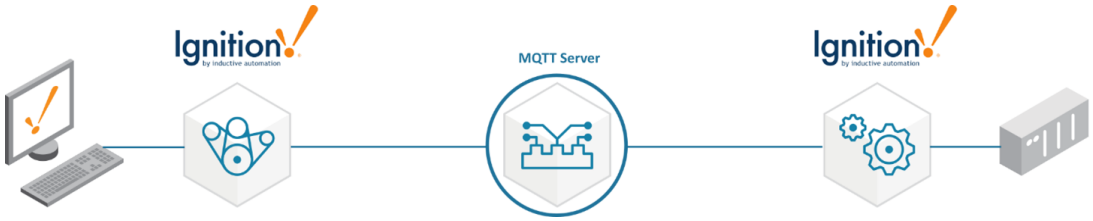


Figura 13. Diagrama del protocolo MQTT

4.3.2. Sparkplug B. La inclusión de Sparkplug B en el protocolo MQTT es clave para el sistema IIoT de la estación CASIRI, integrado con Ignition, debido a su capacidad para estandarizar y estructurar la comunicación entre dispositivos. Mientras que MQTT por sí solo ofrece una forma ligera de transmitir mensajes entre clientes y servidores, Sparkplug B añade una capa de

modelado de datos unificado y jerárquico que facilita que todos los dispositivos intercambien información de manera coherente y organizada. Esto asegura que cualquier dispositivo o aplicación conectada dentro del sistema hable el mismo "lenguaje", permitiendo una integración más fluida y eficiente.

Sparkplug B

Sparkplug B es una especificación que define cómo deben formatearse los datos y cómo deben gestionarse los eventos en un sistema basado en MQTT. En un entorno industrial, los dispositivos generan muchos datos que deben ser interpretados correctamente por diversas aplicaciones. Sparkplug B ofrece un modelo de datos que organiza esta información de manera jerárquica y que incluye:

- Nodos (dispositivos o sistemas físicos como sensores, controladores, etc.).
- Tags o puntos de datos (variables medidas, como la temperatura, presión, o datos de estado).

Este formato estructurado permite a los usuarios crear un árbol de datos en el que cada dispositivo y punto de datos está claramente definido. Por ejemplo, si un sensor en la estación CASIRI mide la temperatura, Sparkplug B etiquetaría estos datos como parte del "nodo" al que pertenece el sensor y lo categorizaría como un tag dentro de esa estructura.

Ejemplo

Supongamos que tenemos un sensor de temperatura. A través de Sparkplug B, los datos enviados por este sensor seguirían una estructura similar a:

- Estación_CASIRI
  - Sensores
    - Sensor_de_Temperatura
      - Valor_Temperatura = 22.5°C
      - Estado = Activo

Esta organización asegura que los datos sean fácilmente interpretables y procesables, tanto por Ignition como por otros dispositivos o sistemas que accedan a la información.

Beneficios

1. Interoperabilidad: Al unificar el formato de datos, Sparkplug B simplifica la comunicación entre dispositivos heterogéneos, como la estación meteorológica, la cámara All Sky y otros componentes.
2. Detección automática de dispositivos: Cuando un dispositivo se conecta o desconecta del sistema, Sparkplug B lo registra automáticamente, eliminando la necesidad de reconfiguración manual.
3. Recuperación tras fallos: Si la conexión MQTT se interrumpe, Sparkplug B garantiza que, una vez restablecida, los dispositivos puedan enviar los datos que se perdieron durante la interrupción, lo que es crucial para un monitoreo constante en entornos remotos.

4.3.3. Servidor HiveMQ. Se eligió el servidor HiveMQ como servidor MQTT para la implementación del sistema IIoT de la estación CASIRI debido a su compatibilidad con Spark-

plug B y su integración fluida con Ignition. HiveMQ ofrece una infraestructura robusta y escalable que asegura una transmisión eficiente y confiable de datos, lo cual es esencial para un sistema de monitoreo en tiempo real como el de CASIRI. Además, su soporte nativo para Sparkplug B facilita la interoperabilidad y el manejo estructurado de los datos, garantizando una comunicación eficiente y segura entre los dispositivos y las aplicaciones del sistema. La Figura 14 muestra la interfaz gráfica del servidor HiveMQ donde se configura la conexión y se muestran los detalles de la misma.

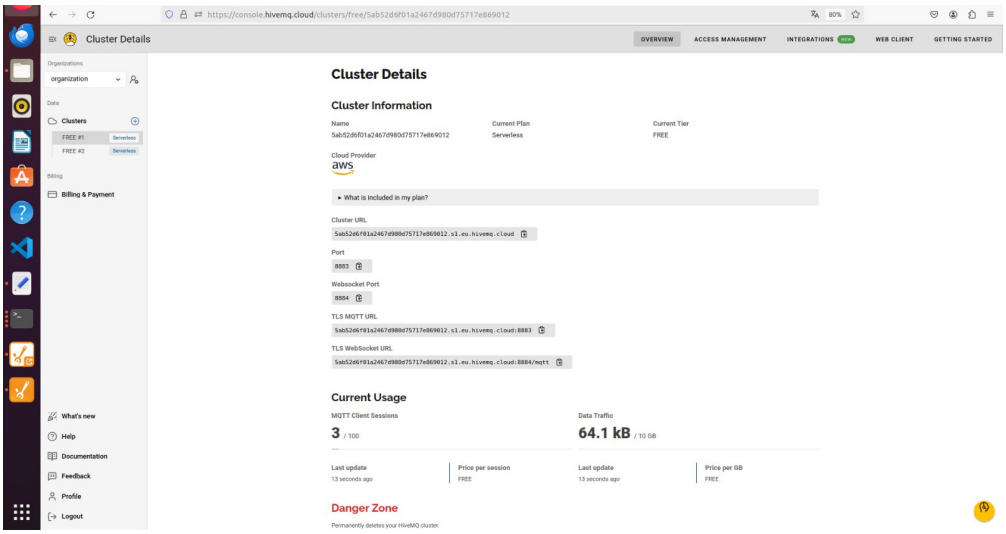


Figura 14. Interfaz gráfica HiveMQ

El diagrama final de la arquitectura del sistema IIoT integrado en CASIRI, con cada uno de sus componentes seleccionados, se presenta en la Figura 15. Esta representación gráfica ilustra la interacción entre el hardware, software y las comunicaciones, proporcionando una visión clara de cómo se organiza y fluye la información dentro del sistema. La integración de estos elementos asegura que el monitoreo remoto y la gestión de datos en tiempo real se realicen de manera eficiente

y confiable, cumpliendo con los objetivos de CASIRI en el contexto del proyecto.

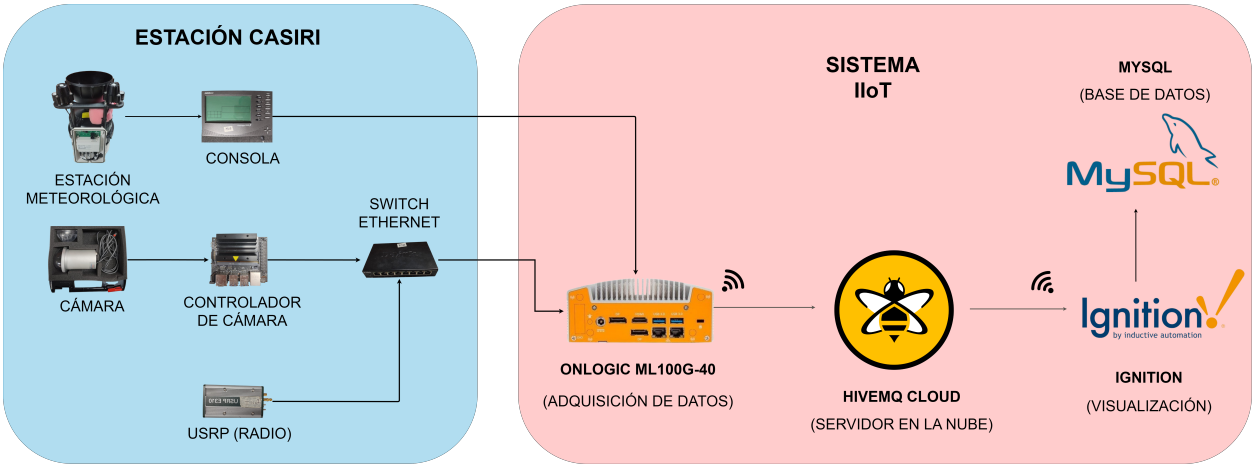


Figura 15. Diagrama final de la integración de CASIRI y el sistema IIoT

5. Integración del sistema IIoT

La integracion del sistema IIoT a CASIRI implica la interconexión, configuración y puesta en marcha de los componentes de hardware, software y de comunicaciones, para lograr que funcionen correctamente con el objetivo de capturar, procesar y transmitir datos en tiempo real. Este capítulo detalla el proceso de integración, haciendo énfasis en los aspectos clave de hardware, software, y cómo estos subsistemas están interconectados para garantizar el funcionamiento continuo y fiable de CASIRI.

5.1. Hardware

CASIRI integra la estación meteorológica Davis Vantage Pro 2 al sistema de adquisición de datos mediante USB. Para gestionar la adquisición de estos datos, se desarrolló un script en Python que los recoge y los envía posteriormente a través del protocolo MQTT Sparkplug B.

Además, la cámara Oculus All-Sky se conecta al controlador Jetson Nano también mediante USB. La Jetson Nano se comunica con el sistema de adquisición de datos a través de Ethernet, mientras que la conexión entre ambos dispositivos se gestiona mediante SSH. Los scripts desarrollados para la captura de imágenes y su envío a través de MQTT Sparkplug B se ejecutan de manera remota desde el sistema de adquisición de datos.

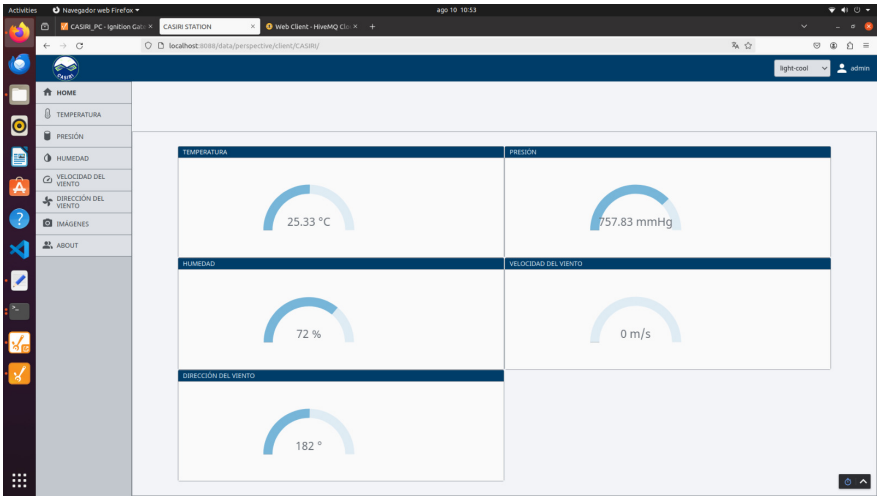
5.2. Software

El software Ignition es el componente central de la plataforma de monitoreo de CASIRI. Este software integra todos los dispositivos y subsistemas, proporcionando una plataforma unificada para la gestión y visualización de datos.

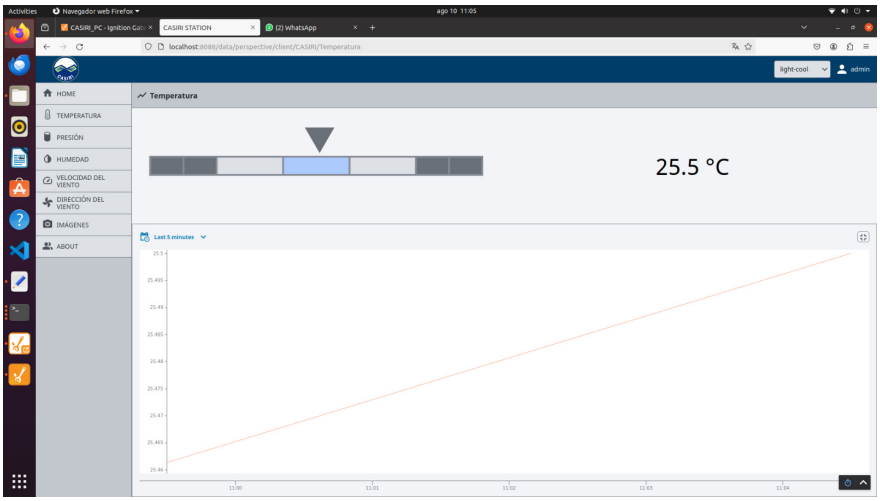
Ignition permite la creación de interfaces gráficas de usuario (HMI) personalizadas, a través de las cuales se puede monitorear en tiempo real las condiciones atmosféricas y las imágenes capturadas por la cámara. La plataforma centraliza la gestión de datos, permitiendo no solo ver los datos en tiempo real, sino que también se puedan almacenar y analizar los datos históricos.

La plataforma implementada en Ignition consta principalmente de la vista principal (Home), la vista por variable y la vista de la cámara. En la vista principal, se presenta un resumen general de las variables atmosféricas, permitiendo acceder rápidamente a cada una de éstas. La vista por variable ofrece una representación detallada de cada variable monitoreada, facilitando su seguimiento individual y análisis específico. Finalmente, la vista de la cámara proporciona una visualización en tiempo real de las imágenes captadas.

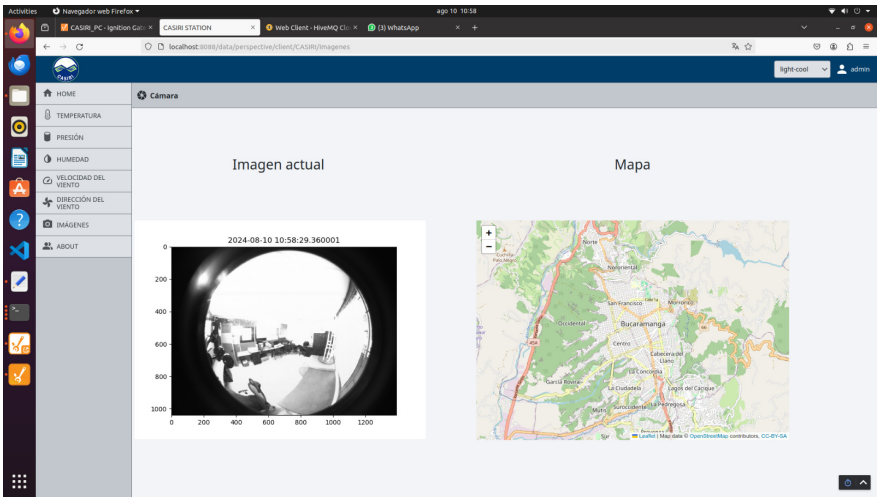
Esta configuración permite centralizar toda la información relevante en una interfaz intuitiva y de fácil acceso. La Figura 16 muestra la interfaz gráfica de la plataforma implementada en Ignition.



(a) Vista principal (Home)



(b) Vista de una variable



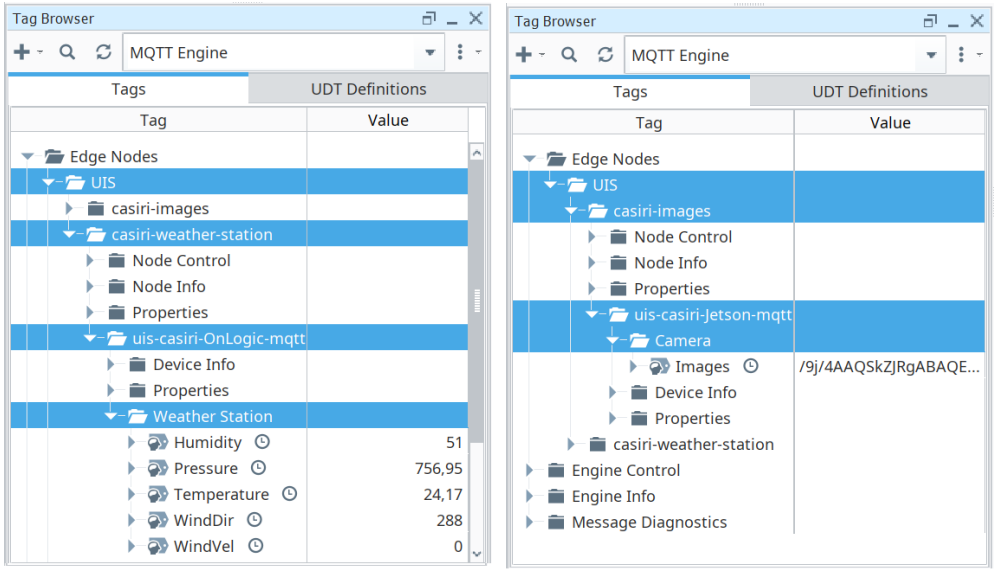
(c) Vista de la cámara

Figura 16. Interfaz gráfica desarrollada en Ignition

Para asegurar que todos los dispositivos del sistema “hablen el mismo lenguaje” y envíen datos de manera estructurada, se implementó la especificación Sparkplug B sobre el protocolo MQTT. Sparkplug B permite una estructura de datos jerárquica que facilita la identificación de cada dispositivo y los datos que genera.

Por ejemplo, la estación meteorológica y la cámara envían sus datos en una estructura de árbol, donde cada nodo representa un tipo de dato específico, ya sea temperatura, humedad, velocidad del viento, o imágenes del cielo. Esto asegura que los datos sean fácilmente identificables y organizados para su procesamiento y análisis.

Otra ventaja de Sparkplug B es su capacidad para manejar la detección automática de dispositivos y la recuperación de datos después de fallos o desconexiones. La Figura 17 presenta la estructura jerárquica de los datos recibidos en Ignition de la estación meteorológica y la cámara.



(a) Datos de la estación meteorológica (b) Datos de la cámara

Figura 17. Estructura jerárquica utilizada para la recepción de datos en Ignition de la estación meteorológica y la cámara

Los datos atmosféricos recolectados por la estación meteorológica y las imágenes capturadas por la cámara son enviados a un servidor HiveMQ en la nube para que el cliente Ignition se pueda conectar de manera remota y visualizar los datos.

Este proceso de sincronización asegura que todos los subsistemas operen en conjunto, permitiendo una visión completa y unificada de las condiciones en el sitio de observación.

El sistema IIoT está diseñado para permitir el monitoreo remoto desde cualquier ubicación con acceso a internet. Esto significa que se puede acceder al sistema a través de un navegador web y visualizar las variables medidas sin necesidad de estar físicamente en el sitio.

Este monitoreo remoto se realiza a través de la plataforma Ignition, que actúa como una interfaz entre los usuarios y los dispositivos en el campo. La capacidad de visualizar datos de manera remota es crucial para minimizar la necesidad de desplazamientos y facilitar el mantenimiento y la operación del sistema en sitios remotos.

6. Verificación del sistema

La metodología utilizada para la implementación del sistema IIoT en la estación CASIRI se basó en un enfoque progresivo, estructurado en una serie de pruebas experimentales que garantizaron la funcionalidad de cada uno de los componentes integrados. La secuencia de pruebas se desarrolló de la siguiente manera:

1. **Envío de Información a través de MQTT**: Se corroboró el correcto envío de datos mediante el protocolo MQTT.

2. **Visualización en Ignition con Datos Simulados**: Utilizando datos simulados en Python enviados a través de MQTT, se verificó la visualización en la plataforma Ignition.

3. **Configuración de la Estación Meteorológica**: Se configuró la estación meteorológica para visualizar los datos en Ignition usando MQTT.

4. **Envío de Datos usando Sparkplug B**: Se comprobó el envío de datos utilizando la especificación Sparkplug B y su visualización en Ignition.

5. **Respaldo y Almacenamiento de Datos**: Finalmente, se verificó el correcto funcionamiento del respaldo y almacenamiento de datos, así como su visualización en Ignition.

Las secciones siguientes detallan el proceso de validación de cada una de estas pruebas experimentales. Además, el material audiovisual de las pruebas está disponible a través de los enlaces proporcionados en cada sección.

6.1. Prueba 1: Publicación/Suscripción en MQTT

Descripción: En esta prueba se demuestra el funcionamiento de la publicación y suscripción utilizando el protocolo MQTT (Message Queuing Telemetry Transport). MQTT es un protocolo de mensajería ligero diseñado para la comunicación entre dispositivos en redes de baja potencia y ancho de banda limitado, lo que lo hace ideal para aplicaciones IoT. La Figura 18 presenta el diagrama del protocolo MQTT, utilizado para esta prueba.

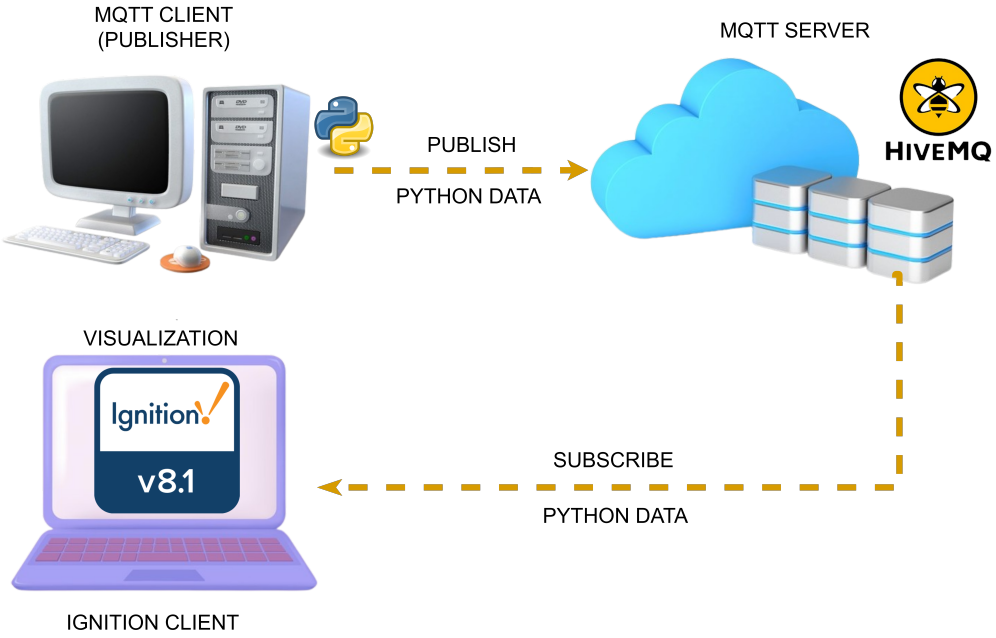


Figura 18. Diagrama prueba 1

Materiales:

- Computador con conexión a Internet.
- Servidor MQTT en la nube (En este caso, HiveMQ Cloud).
- PC portátil con cliente MQTT para Python instalado (En este caso, Paho MQTT).

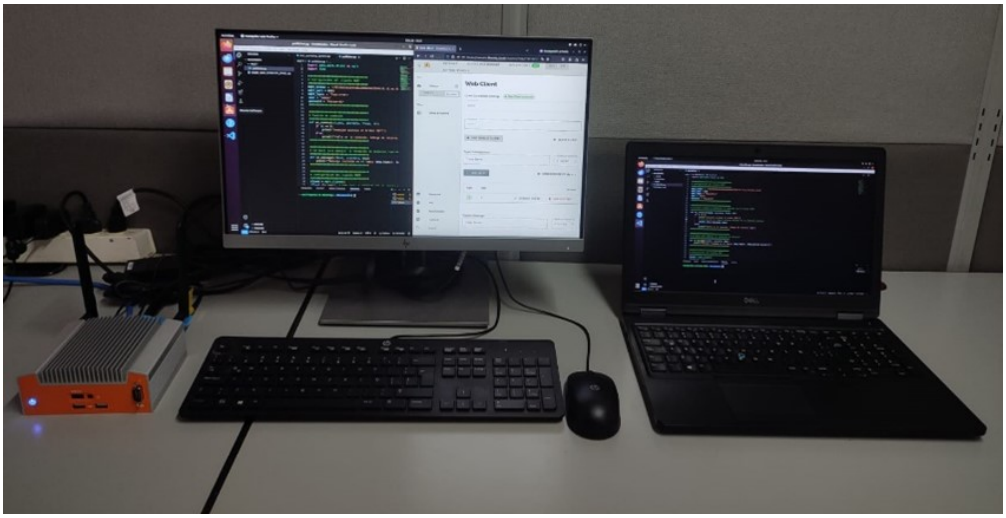


Figura 19. Materiales para la prueba 1

Procedimiento general:

1. Configurar el servidor MQTT (HiveMQ Cloud).
2. Configurar el cliente MQTT en el PC portátil.
3. Suscribirse al tema (topic) en el que se va a publicar.
4. Publicar el mensaje asegurándose que sea el mismo tema (topic) al que se suscribió en el paso anterior.
5. Verificar que el mensaje recibido corresponde al mensaje publicado.

Esta prueba demuestra el flujo básico de información en un sistema IoT utilizando el protocolo MQTT. La publicación y suscripción permiten que los dispositivos envíen y reciban datos de manera eficiente y asíncrona a través de un servidor centralizado. Al suscribirse a un tema específico, un dispositivo indica su interés en recibir mensajes relacionados con ese tema. Cuando

se publica un mensaje en ese tema, el servidor lo envía a todos los dispositivos suscritos, lo que permite la distribución eficiente de la información en toda la red IoT.

Enlace a video Prueba 1: `https://youtu.be/igaOg0mcEWo`

6.2. Prueba 2: Visualización en Ignition con datos simulados en Python usando MQTT

Descripción: Esta prueba demuestra cómo visualizar datos simulados en Ignition utilizando Python para generar datos y MQTT para la comunicación entre el generador de datos y la plataforma de visualización. Ignition es una plataforma de software de automatización industrial que proporciona herramientas para la visualización y control de procesos. La Figura 20 presenta el diagrama del protocolo MQTT conectándose a un cliente Ignition.

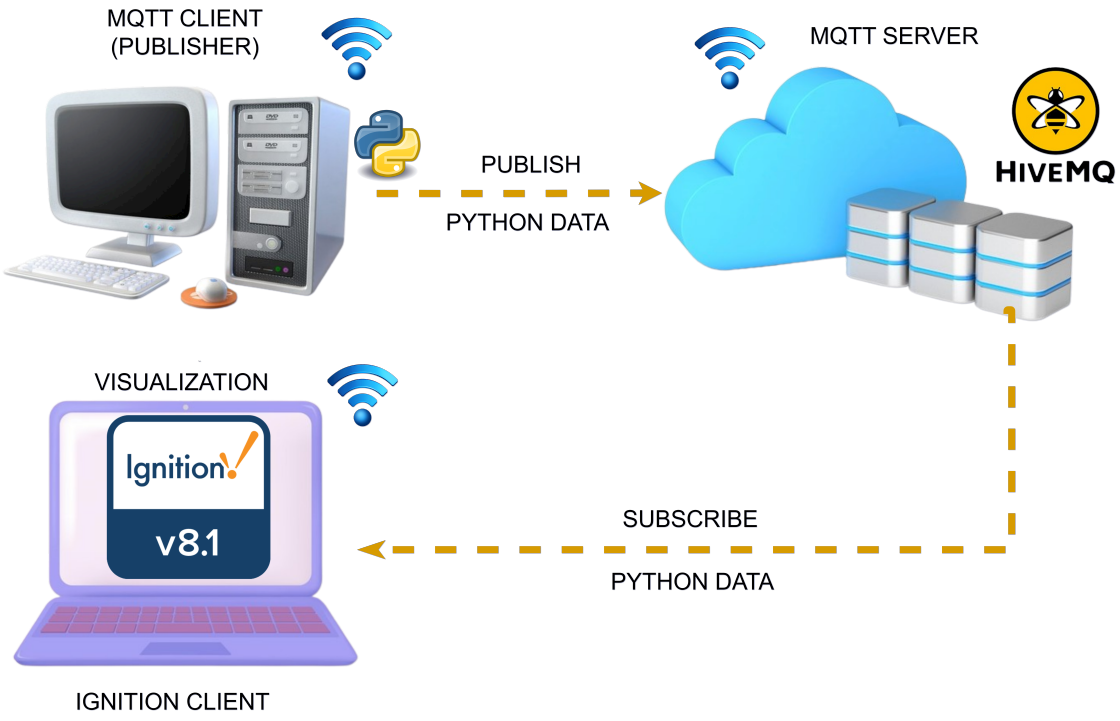


Figura 20. Diagrama prueba 2

Materiales:

- PC portátil con Ignition instalado.
- Servidor MQTT en la nube (En este caso, HiveMQ Cloud).
- Computador con cliente MQTT para Python instalado (En este caso, Paho MQTT).

Procedimiento general:

1. Configurar el generador de datos (Cliente MQTT - publicador).
2. Instalar módulo MQTT Engine.
3. Configurar Ignition (módulo MQTT Engine para hacer compatible con MQTT)
4. Diseñar la interfaz de visualización.
5. Vincular los datos simulados en Python a elementos gráficos de la interfaz.
6. Verificar que los datos simulados corresponden con los datos visualizados.

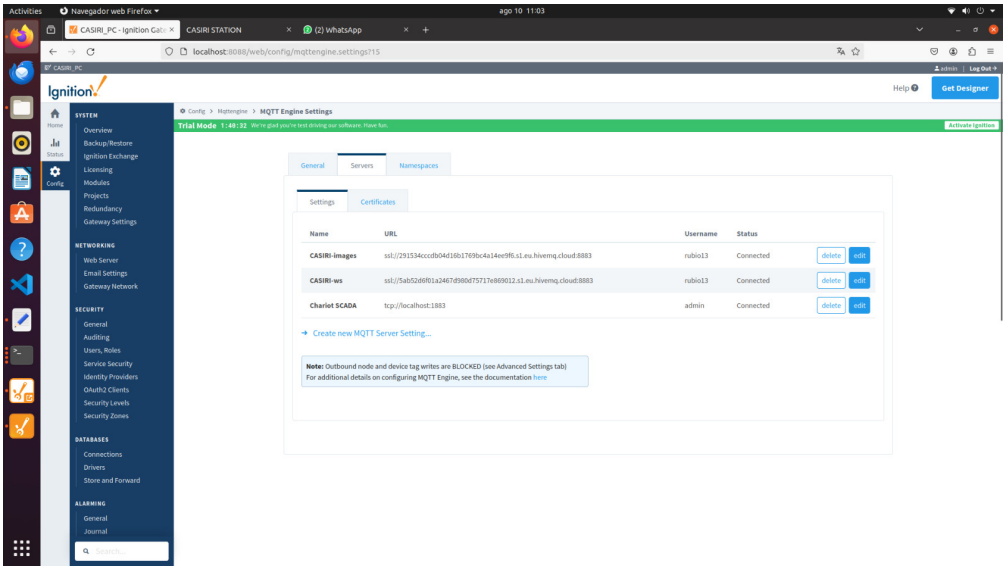


Figura 21. Configuración módulo MQTT Engine en Ignition

Esta prueba demuestra cómo integrar datos generados por un script Python utilizando MQTT en la plataforma de visualización Ignition. Al simular datos y publicarlos en un tema MQTT, se puede utilizar Ignition para suscribirse a estos datos y visualizarlos en tiempo real. Esto ilustra cómo Ignition puede conectarse y recibir datos de dispositivos externos a través de MQTT.

Enlace a video Prueba 2: `https://youtu.be/yi7MfaTczVw`

6.3. Prueba 3: Configuración de la estación meteorológica DAVIS Vantage Pro-2 y visualización en Ignition usando MQTT

Descripción: Esta prueba muestra cómo configurar la estación meteorológica Davis Vantage Pro 2 para recopilar datos ambientales y transmitirlos a través del protocolo MQTT. Luego, se utiliza Ignition para visualizar estos datos en tiempo real. La Figura 22 muestra la conexión de un cliente Ignition con la estación meteorológica mediante MQTT.

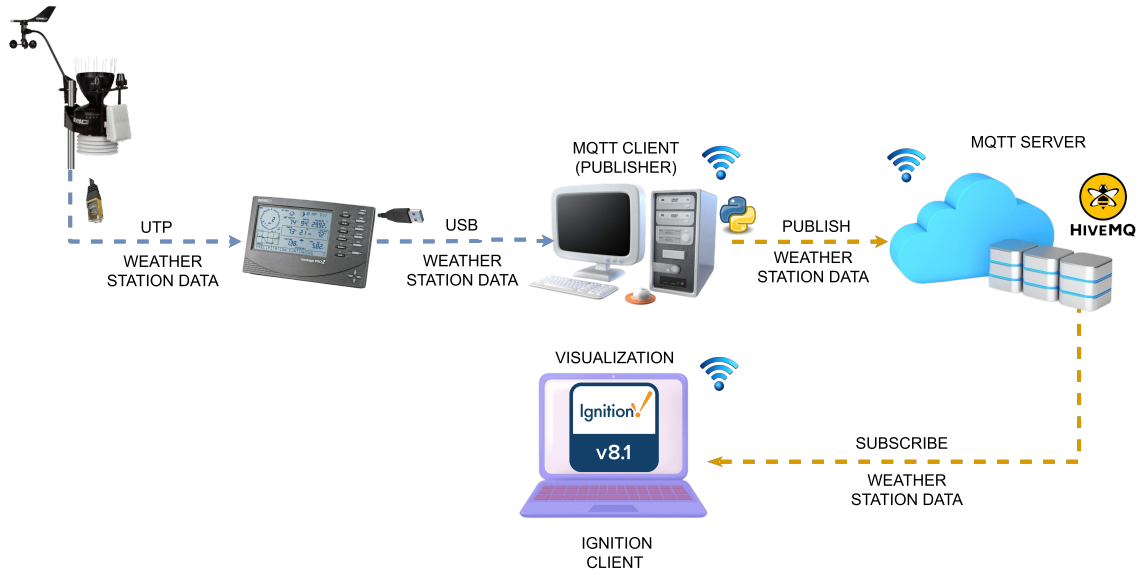


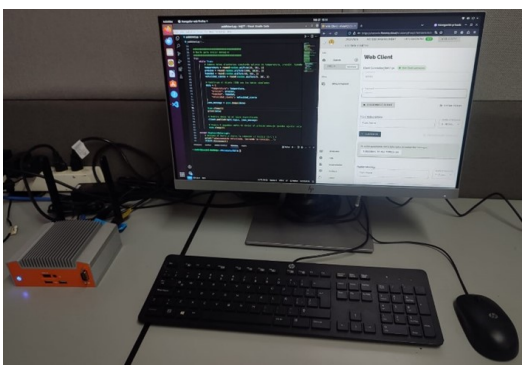
Figura 22. Diagrama prueba 3

Materiales:

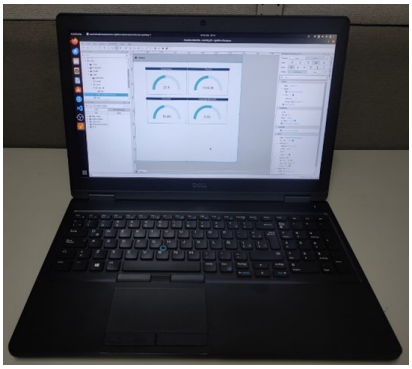
- Estación meteorológica Davis Vantage Pro-2.
- Consola de recepción de datos de la estación meteorológica.
- PC portátil con Ignition instalado.
- Servidor MQTT en la nube (En este caso, HiveMQ Cloud).
- Computador con cliente MQTT para Python instalado (En este caso, Paho MQTT).

Procedimiento general:

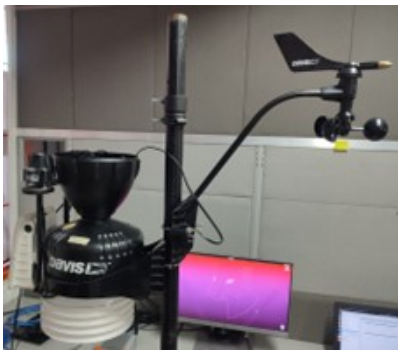
1. Configurar la estación meteorológica.
2. Configurar Ignition.
3. Vincular las variables de la estación a elementos gráficos de la interfaz.
4. Ejecutar el código Python del cliente MQTT (publicador).
5. Validar los datos que muestra la consola con los visualizados en Ignition.



(a) Cliente MQTT publicador



(b) Cliente Ignition suscriptor



(c) Estación meteorológica

(d) Consola para la estación meteorológica

Figura 23. Materiales para la prueba 3

Esta prueba demuestra cómo integrar una estación meteorológica Davis Vantage Pro 2 con la plataforma de visualización Ignition utilizando MQTT. Al transmitir datos de la estación meteorológica a través de MQTT, podemos utilizar Ignition para suscribirnos a estos datos y visualizarlos en tiempo real.

Enlace a video Prueba 3: `https://youtu.be/bGrvgkFBvNs`

6.4. Prueba 4: Envío de datos usando la especificación Sparkplug B y visualización en Ignition

Descripción: En esta prueba se demuestra cómo implementar el envío de datos utilizando la especificación Sparkplug B, un protocolo de comunicación MQTT optimizado para la industria

IoT. Luego, se utiliza Ignition para visualizar estos datos en tiempo real, demostrando la capacidad de visualización de la plataforma. La Figura 24 muestra el diagrama del protocolo MQTT conectándose a un cliente Ignition, implementando Sparkplug B.

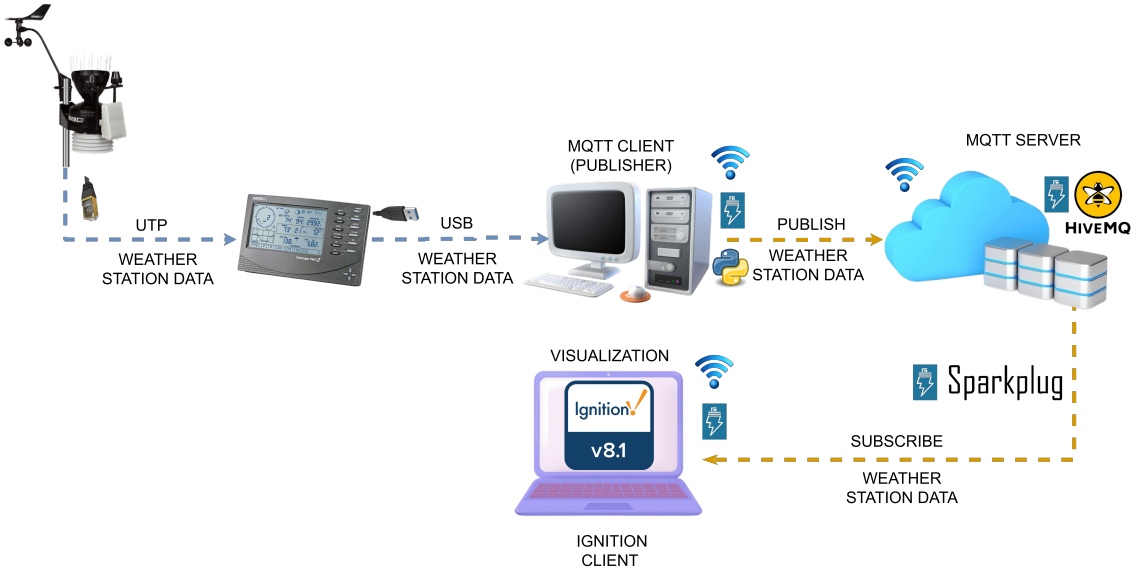


Figura 24. Diagrama prueba 4

Materiales:

- Estación meteorológica Davis Vantage Pro-2.
- Consola de recepción de datos de la estación meteorológica.
- PC portátil con Ignition instalado.
- Servidor MQTT en la nube (En este caso, HiveMQ Cloud).
- Computador con cliente MQTT compatible con Sparkplug B.

Procedimiento general:

1. Vincular las variables de la estación a elementos gráficos de la interfaz.

2. Ejecutar el código Python del cliente MQTT publicador usando Sparkplug B.

3. Validar los datos que muestra la consola con los visualizados en Ignition.

4. Verificar que los datos se están enviando en el formato de Sparkplug B en Ignition.

Esta prueba demuestra cómo utilizar la especificación Sparkplug B para enviar datos de manera eficiente y estructurada a través de MQTT, y luego visualizar estos datos en Ignition. Al adoptar Sparkplug B, podemos mejorar la interoperabilidad entre dispositivos y simplificar la integración de datos en la plataforma de visualización. Esto proporciona una solución robusta y escalable para la implementación de sistemas de monitoreo y control en entornos industriales y de IoT, aprovechando las capacidades de comunicación y visualización de Ignition junto con las ventajas de la especificación Sparkplug B.

Enlace a video Prueba 4: `https://youtu.be/6r9AY_KxCUc`

6.5. Prueba 5: Respaldo, almacenamiento de datos y visualización en Ignition

Descripción: En esta prueba se demuestra cómo configurar un sistema para respaldar y almacenar datos en una base de datos utilizando Ignition. Luego, se utiliza Ignition para visualizar estos datos almacenados en tiempo real, demostrando la capacidad de la plataforma para gestionar datos históricos y en tiempo real. La Figura 25 muestra el diagrama del protocolo MQTT conectándose a un cliente Ignition, implementando Sparkplug B y respaldando los datos en una base de datos.

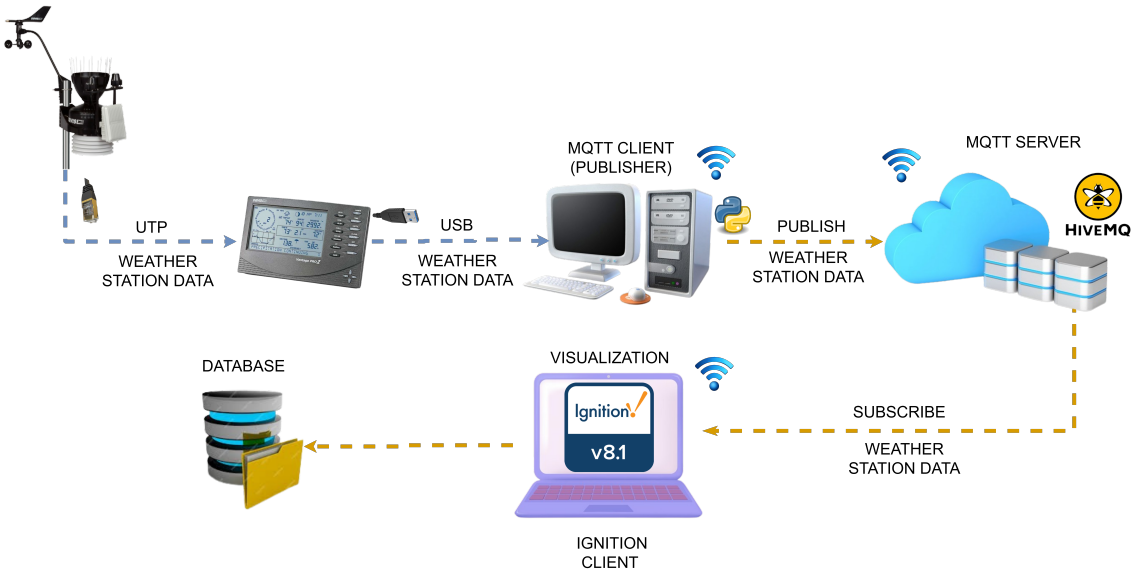


Figura 25. Diagrama prueba 5

Materiales:

- Estación meteorológica Davis Vantage Pro-2.
- Consola de recepción de datos de la estación meteorológica.
- PC portátil con Ignition instalado.
- Servidor MQTT en la nube (En este caso, HiveMQ Cloud).
- Gestor de base de datos (En este caso Workbench de MySQL)
- Computador con cliente MQTT para Python instalado (En este caso, Paho MQTT).

Procedimiento general:

1. Configurar la conexión entre Ignition y la base de datos.

2. Configurar Ignition para poder almacenar los datos.

3. Ejecutar el código Python del cliente MQTT publicador.

4. Validar los datos que muestra la consola con los visualizados en Ignition.

5. Comprobar que los datos históricos corresponden a los enviados previamente.

Esta prueba demuestra cómo configurar Ignition para respaldar y almacenar datos en una base de datos, así como visualizar estos datos almacenados en tiempo real. Al respaldar los datos en una base de datos, se puede mantener un registro histórico de los datos recopilados. Además, al visualizar estos datos en Ignition, se pueden aprovechar las capacidades de visualización de la plataforma para representar los datos de manera significativa y comprensible.

Enlace a video Prueba 5: `https://youtu.be/ZCaMGB6DTVM`

Además, se incluye el enlace al video de validación del sistema completo: `https://youtu.be/4RvGn-hWm80`

Este proceso metódico y secuencial permitió asegurar que cada componente, y el sistema como un todo, funcionaran de manera correcta antes de su implementación definitiva en la estación CASIRI. A través de una integración cuidadosamente validada, se logró no solo garantizar la operatividad y confiabilidad del sistema, sino también sentar las bases para futuras expansiones, como la incorporación de nuevos sensores y mejoras en la gestión de datos. Este enfoque asegura que la estación CASIRI esté preparada para adaptarse a las necesidades cambiantes, posicionándose co-

mo una plataforma flexible y escalable, capaz de evolucionar junto con los avances tecnológicos.

7. Guías pedagógicas

Al concluir el proceso de verificación del sistema IIoT para el monitoreo de la estación CASIRI, se diseñó una estrategia pedagógica para destacar la finalidad educativa del sistema, orientada a la enseñanza de la Industria 4.0. Considerando las opciones presentadas en la propuesta de investigación y tras varias reuniones con el equipo de trabajo de GISEL, se decidió elaborar guías educativas. Tras esta elección, se estableció un formato estándar para las guías que se desarrollarían a partir de los distintos casos de estudio. Estas guías se componen de las siguientes secciones:

- Título
- Autor y revisor
- Objetivos de aprendizaje
- Conocimientos previos requeridos
- Introducción
- Materiales
- Actividades preliminares
- Actividades a desarrollar
- Actividad de cierre
- Conceptos para revisar

- Referencias bibliográficas

En las siguientes secciones se presenta un breve resumen de cada una de las guías diseñadas, mientras que en los apéndices A, B y C se encuentran los documentos completos correspondientes a cada una. Las guías educativas proponen un enfoque de aprendizaje por etapas, comenzando con procedimientos básicos como el envío de datos usando MQTT, hasta llegar a la interacción de Ignition con el sistema completo de la estación CASIRI.

7.1. Explorando el Internet de las cosas (IoT) con Python y MQTT

7.1.1. Introducción. En esta guía se explora cómo utilizar Python junto con el protocolo MQTT para desarrollar proyectos de IoT. El Internet de las Cosas permite la conexión y comunicación entre dispositivos, y MQTT es un protocolo ligero y eficiente que facilita esta comunicación. Al finalizar la guía se espera que el estudiante implemente proyectos simples de IoT usando MQTT.

7.1.2. Actividades a desarrollar.

1. Familiarización de conceptos.
2. Configuración del entorno.
3. Envío de datos usando MQTT.
4. Envío de datos usando MQTT y sensores.

7.2. Desarrollo de un sistema IoT con Ignition

7.2.1. Introducción. En esta guía se exploran los conceptos básicos de Ignition para el desarrollo de sistemas IoT. El objetivo es que, al finalizar, el estudiante pueda crear una aplicación funcional que interactúe con dispositivos IoT simulados.

7.2.2. Actividades a desarrollar.

1. Comprensión del concepto de tag en Ignition.
2. Asociación de tags a valores simulados en Python.
3. Desarrollo de una interfaz gráfica en Ignition.
4. Conexión de Ignition con bases de datos para almacenamiento de información.
5. Implementación de un proyecto IoT que involucra sensores.

7.3. Visualización remota de datos usando el protocolo MQTT Sparkplug B

7.3.1. Introducción. En esta guía se explora cómo visualizar de forma remota datos adquiridos de la estación CASIRI utilizando el protocolo MQTT Sparkplug B. El objetivo es aprender a adquirir y visualizar variables ambientales y fotos del cielo a través de una interfaz desarrollada en Ignition.

7.3.2. Actividades a desarrollar.

1. Adquisición de variables de la estación CASIRI.
2. Implementación de Sparkplug B.

3. Implementación del sistema completo.

8. Conclusiones

8.1. Conclusiones

1. Este proyecto demuestra la factibilidad de implementar un sistema IIoT funcional y escalable en un entorno real, mediante la integración de tecnologías industriales modernas como el protocolo MQTT Sparkplug B, el software Ignition, bases de datos MySQL y servicios en la nube como HiveMQ Cloud. Esta solución técnica permite la adquisición, transmisión, almacenamiento y visualización remota de datos ambientales y de imágenes del cielo en tiempo real.

2. La implementación del sistema responde de manera efectiva a las necesidades de observación remota de la estación CASIRI, integrando dispositivos heterogéneos como la estación meteorológica y la cámara All Sky bajo una arquitectura coherente basada en tecnologías modernas. Por su parte, dadas las características del sistema implementado para la estación meteorológica y la cámara, se garantiza la posibilidad de integración futura del receptor de radio para la detección de interferencias por radiofrecuencia (RFI). Esta tarea será realizada en proyectos futuros de los grupos de investigación de interés.

3. Desde el punto de vista educativo, este proyecto presenta una propuesta pedagógica innovadora, ya que no solo entrega un sistema tecnológico funcional, sino que lo acompaña con un conjunto de guías de aprendizaje estructuradas que permiten abordar el desarrollo por etapas. Esta segmentación facilita la comprensión gradual de conceptos clave del IIoT y permite que

los estudiantes desarrollen competencias técnicas alineadas con los estándares de la industria 4.0.

4. La inclusión de material audiovisual como recurso de validación y apoyo didáctico fortalece la propuesta educativa del proyecto, permitiendo que los estudiantes verifiquen el funcionamiento correcto de cada etapa del sistema, refuercen su aprendizaje de forma visual y logren una mejor conexión entre teoría y práctica.

5. En conjunto, el proyecto resuelve un problema técnico concreto y un desafío educativo, aportando una herramienta didáctica y una solución tecnológica aplicable a escenarios reales de monitoreo ambiental y desarrollo de habilidades para la industria digital.

8.2. Trabajo futuro y recomendaciones

El trabajo futuro en la estación CASIRI se centrará en ampliar sus capacidades mediante la integración de un sistema de medición de interferencias de radiofrecuencia (RFI) utilizando el dispositivo USRP. La infraestructura ya implementada garantiza la posibilidad de integración futura del receptor de radio para la detección de interferencias por radiofrecuencia (RFI) ya que el sistema está basado en un protocolo de comunicación escalable como MQTT Sparkplug B, que permite añadir nuevos nodos de manera sencilla sin modificar la arquitectura existente. La capacidad de detectar y analizar RFI permitirá a CASIRI contribuir a la identificación de “cielos silenciosos”, complementando la evaluación para la caracterización de sitios candidatos para la implementación de radio observatorios.

En paralelo, el crecimiento tecnológico de la estación irá acompañado del desarrollo de

nuevas estrategias pedagógicas, orientadas a enriquecer la formación en el ámbito de la Industria 4.0. Estas iniciativas educativas seguirán siendo fundamentales para mantener actualizados tanto a estudiantes como a profesionales, brindándoles experiencias prácticas que faciliten su adaptación a las demandas tecnológicas del entorno académico y profesional actual.

Un aspecto relevante a considerar en futuras fases del proyecto es la validación del impacto pedagógico de las guías y el material audiovisual desarrollado. Su implementación en contextos reales de formación académica o técnica permitirá evaluar su efectividad como herramienta de enseñanza, identificar oportunidades de mejora y avanzar hacia una aplicación a mayor escala.

Finalmente, se recomienda documentar y sistematizar todo el proceso de implementación, con el objetivo de facilitar su replicabilidad en otros escenarios de monitoreo ambiental o formación en Industria 4.0. Esto consolidaría a CASIRI no solo como una estación de observación avanzada, sino también como un modelo pedagógico y tecnológico transferible a diversos contextos educativos y de investigación.

Referencias Bibliográficas

Akbar, M. A., Rashid, M. M., and Embong, A. H. (2018). Technology based learning system in internet of things (IoT) education. In *2018 7th International Conference on Computer and Communication Engineering (ICCCE)*, pages 192–197.

Amelia, A., Roslina, Fahmi, N., Pranoto, H., Sundawa, B. V., Hutauruk, I. S., and Arief, A. (2020). MQTT protocol implementation for monitoring of environmental based on IoT. In *2020 International Conference on Applied Science and Technology (iCAST)*, pages 700–703.

Amjad, A., Azam, F., Anwar, M. W., and Butt, W. H. (2021). A systematic review on the data interoperability of application layer protocols in industrial IoT. *IEEE Access*, 9:96528–96545.

Antonelli, D., D'Addona, D. M., Maffei, A., Modrak, V., Putnik, G., Stadnicka, D., and Stylios, C. (2019). Tiphys: An open networked platform for higher education on industry 4.0. *Procedia CIRP*, 79:706–711. 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 18-20 July 2018, Gulf of Naples, Italy.

Aroon, N. (2016). Study of using MQTT cloud platform for remotely control robot and GPS tracking. In *2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 1–6.

Automation, I. (2024a). Inductive University course list - Learn Ignition Free. https://inductiveuniversity.com/courses/ignition/whats-new-in-ignition-81/8.1.

Automation, I. (2024b). Industrial Automation Software Solutions by Inductive Automation. https://inductiveautomation.com/.

Bender, M., Kirdan, E., Pahl, M.-O., and Carle, G. (2021). Open-source MQTT evaluation. In *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*, pages 1–4.

Cai, K., Tie, F., Huang, H., Lin, H., and Chen, H. (2015). Innovative experiment platform design and teaching application of the internet of things. *International Journal of Online and Biomedical Engineering (iJOE)*, 11(6):pp. 28–32.

Censier, B., Thomas, I., Auxepaules, G., Flouret, B., and Renaud, P. (2021). Sky modeling for correction and calibration of a single VHF antenna system for interferences monitoring. In *2021 XXXIVth General Assembly and Scientific Symposium of the International Union of Radio Science (URSI GASS)*, pages 1–3.

Cheng, J.-H., Lin, H.-H., Shen, J.-H., Chen, B.-C., and He, Z.-L. (2020). IoT training system for smart manufacturing education. In *2020 3rd IEEE International Conference on Knowledge Innovation and Invention (ICKII)*, pages 182–184.

Dorsemaine, B., Gaulier, J.-P., Wary, J.-P., Kheir, N., and Urien, P. (2015). Internet of things: A definition taxonomy. In *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*, pages 72–77.

Drath, R. and Horch, A. (2014). Industry 4.0: Hit or hype? [industry forum]. *IEEE Industrial Electronics Magazine*, 8(2):56–58.

Embong, A. H. B., Akbar, M. A., and Rashid, M. M. (2019). Design and development of multi-purpose educational and research platform (merp) for learning control and IoT technologies. In *Web of Science*.

Genetec (2024). ¿Por qué el IoT es importante en la seguridad electrónica?. https://www.genetec.com/es/blog/productos/por-que-el-iot-es-importante-en-la-seguridad-electronica. Accedido: 2024-07-19.

Guo, T., Khoo, D., Coultis, M., Pazos-Revilla, M., and Siraj, A. (2018). Poster abstract: IoT platform for engineering education and research (IoT peer)–applications in secure and smart manufacturing. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 277–278.

InfoPLC (2024). Protocolos iiot - infoplc. https://www.infoplc.net/blogs-automatizacion/item/113434-protocolos-iiot. Accedido: 2024-10-16.

Jacko, P., Bereš, M., Kováčová, I., Molnár, J., Vince, T., Dziak, J., Fecko, B., Gans, , and Kováč, D. (2022). Remote IoT education laboratory for microcontrollers based on the stm32 chips. *Sensors*, 22(4).

Kaskatiiski, N. and Boyanov, L. (2021). Efficiency of data exchange of IoT communication protocols. In *2021 International Conference Automatics and Informatics (ICAI)*, pages 358–361.

Kaushik, N. and Bagga, T. (2021). Smart cities using IoT. In *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pages 1–6.

Krushnan, J. and Schrödel, F. (2022). Development of a modern, low cost, lab scale industry 4.0 plant for education. *IFAC-PapersOnLine*, 55(17):156–161. 13th IFAC Symposium on Advances in Control Education ACE 2022.

Liao, Y., de Freitas Rocha Loures, E., and Deschamps, F. (2018). Industrial internet of things: A systematic literature review and insights. *IEEE Internet of Things Journal*, 5(6):4515–4525.

Loukatos, D., Androulidakis, N., Arvanitis, K. G., Peppas, K. P., and Chondrogiannis, E. (2022). Using open tools to transform retired equipment into powerful engineering education instruments: A smart agri-IoT control example. *Electronics*, 11(6).

Mohapatra, D. and Subudhi, B. (2022). Development of a cost-effective IoT-based weather monitoring system. *IEEE Consumer Electronics Magazine*, 11(5):81–86.

Muhammad, A., Nur, D., Zakiah, M., Amar, F., and Nooriwati, M. (2018). Computer assisted e-laboratory using labview and internet-of-things platform as teaching aids in the industrial instrumentation course. *International Journal of Online and Biomedical Engineering (iJOE)*, 14(12):pp. 26–42.

Mumtaz, S., Alsohaily, A., Pang, Z., Rayes, A., Tsang, K. F., and Rodriguez, J. (2017). Massive

internet of things for industrial applications: Addressing wireless IIoT connectivity challenges and ecosystem fragmentation. *IEEE Industrial Electronics Magazine*, 11(1):28–33.

Nykyri, M., Kuisma, M., Kärkkäinen, T. J., Hallikas, J., Jäppinen, J., Korpinen, K., and Silventoinen, P. (2019). IoT demonstration platform for education and research. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 1155–1162.

Pastor-Vargas, R., Tobarra, L., Robles-Gómez, A., Martin, S., Hernández, R., and Cano, J. (2020). A wot platform for supporting full-cycle iot solutions from edge to cloud infrastructures: A practical case. *Sensors*, 20(13).

Phan, H.-P., Vo, V.-D., Nguyen, Q.-H., Duong, V.-H., and Hoang, H.-H. (2021). Building lorawan IoT kits for stem education in interdisciplinary training programs. In *2021 6th International STEM Education Conference (iSTEM-Ed)*, pages 1–4.

Piccialli, F., Bessis, N., and Cambria, E. (2021). Guest editorial: Industrial internet of things: Where are we and what is next? *IEEE Transactions on Industrial Informatics*, 17(11):7700–7703.

Pirrone, D., Fornaro, C., and Assante, D. (2021). Open-source multi-purpose remote laboratory for IoT education. In *2021 IEEE Global Engineering Education Conference (EDUCON)*, pages 1462–1468.

Rodríguez-Calderón, R. and Belmonte-Izquierdo, R. (2021). Educational platform for the deve-

lopment of projects using internet of things. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 16(3):276–282.

Shi, Z., Chen, J., and He, S. (2020). Diy smart house : Exploration and practice of IoT MOOC education. In *2020 15th International Conference on Computer Science Education (ICCSE)*, pages 557–560.

Sisinni, E., Saifullah, A., Han, S., Jennehag, U., and Gidlund, M. (2018). Industrial internet of things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics*, 14(11):4724–4734.

Solano, J., Hernández, I., Duarte, N., and Rey, J. M. (2021). Diseño, desarrollo e implementación de una plataforma IIoT para formación de profesionales en tecnologías de la cuarta revolución industrial. *Encuentro Internacional de Educación en Ingeniería*.

Trilles, S., Monfort-Muriach, A., Gómez-Cambronero, , and Granell, C. (2022). Sucre4stem: Collaborative projects based on IoT devices for students in secondary and pre-university education. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 17(2):150–159.

Wilson, C., Storey, M., and Tzioumis, T. (2013). Measures for control of EMI and RFI at the murchison radioastronomy observatory, Australia. In *2013 Asia-Pacific Symposium on Electromagnetic Compatibility (APEMC)*, pages 1–4.

Xu, L. D., He, W., and Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243.

Yang, K., Zhang, B., Zhang, J., and Zhu, J. (2021). Design of remote control inverter based on mqtt communication protocol. In *2021 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1374–1378.

Zhang, Q., Zhu, C., Yang, L. T., Chen, Z., Zhao, L., and Li, P. (2017). An incremental cfs algorithm for clustering large data in industrial internet of things. *IEEE Transactions on Industrial Informatics*, 13(3):1193–1201.

Apéndices

Apéndice A. Guía #1

Enlace para acceder a la guía de aprendizaje: https://colab.research.google.com/drive/1fFgs2siSXeMg64CCgOaSlwxSPaQYCGPo?usp=drive_link

PRÁCTICA # 04

EXPLORANDO EL INTERNET DE LAS COSAS (IOT) CON PYTHON Y MQTT

Diseñada por: Felipe Rubio (E3T)
Revisada por: Juan Manuel Rey (E3T)

OBJETIVOS DE APRENDIZAJE

Al finalizar esta guía, el estudiante deberá ser capaz de:

- Comprender los principios fundamentales del Internet de las Cosas (IoT).
- Utilizar el protocolo MQTT para la comunicación en entornos IoT.
- Implementar proyectos simples de IoT con Python y MQTT.

CONOCIMIENTOS PREVIOS REQUERIDOS

Antes de comenzar esta guía, se espera que los estudiantes tengan conocimientos básicos en Python, incluyendo la capacidad de trabajar con variables, estructuras de control y funciones. Además, es beneficioso que tengan una comprensión general de los conceptos de redes y comunicación.

INTRODUCCIÓN

En esta guía, exploraremos cómo utilizar Python junto con el protocolo MQTT para desarrollar proyectos de IoT. El Internet de las Cosas permite la conexión y comunicación entre dispositivos, y MQTT es un protocolo ligero y eficiente que facilita esta comunicación.

MATERIALES

- Computadora con acceso a internet.
- Entorno de desarrollo integrado (IDE) para Python (como Visual Studio Code, PyCharm, o IDLE).
- Biblioteca Paho-MQTT para Python.

ACTIVIDADES PRELIMINARES

- **Instalación de Python:** Descarga e instala la última versión de Python desde el sitio oficial (https://www.python.org/). Asegúrate de tener Python instalado en tu computadora.
- **Instalación de un IDE:** Configura un entorno de desarrollo integrado según tu preferencia. Según la preferencia también se puede trabajar en entornos como Google Colab o Jupyter Notebooks.
- **Instalación de la Biblioteca MQTT:** Instala la biblioteca Paho-MQTT utilizando el gestor de paquetes de Python: "pip install paho-mqtt".

ACTIVIDAD 1: ENTENDIENDO IOT Y MQTT

1. Investiga y describe los conceptos fundamentales del Internet de las Cosas (IoT). Puedes visitar la siguiente página web:

- Conceptos fundamentales del IoT

2. Comprende cómo MQTT facilita la comunicación en entornos IoT. Puedes visitar las siguientes páginas web:

- Sitio oficial MQTT
- Conceptos fundamentales de MQTT

ACTIVIDAD 2: CONFIGURACIÓN DEL ENTORNO

1. Crear un programa Python simple que establezca una conexión MQTT con un servidor.

Programa simple para establecer conexión MQTT con un servidor

En esta instancia, emplearemos un servidor EMQX junto al puerto 1883 (sin seguridad) para lograr una conexión exitosa con el servidor MQTT. A continuación, se muestra un ejemplo del código utilizado para esta tarea.

```python
import paho.mqtt.client as mqtt

# Callback cuando se establece la conexión con el broker MQTT
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Conexión exitosa al broker MQTT")
    else:
        print(f"Fallo en la conexión. Código de retorno: {rc}")

# Configuración del cliente MQTT
client = mqtt.Client()
client.on_connect = on_connect

# Especifica el servidor MQTT y el puerto (por defecto es 1883)
mqtt_broker = "broker.emqx.io"
mqtt_port = 1883

# Conexión al servidor MQTT
client.connect(mqtt_broker, mqtt_port, 60)

# Inicia el bucle para mantener la conexión activa
client.loop_start()

# Aquí puedes agregar más funcionalidades o publicar/suscribir a temas MQTT

# Detener el bucle y cerrar la conexión
client.loop_stop()
client.disconnect()
```

2. Comprender los conceptos de publicación y suscripción en MQTT.

ACTIVIDAD 3: PROYECTO BÁSICO DE IOT

1. Ejercicio práctico: Proyecto sencillo de IoT utilizando Python y MQTT.

Para este ejercicio vamos a conectar dos dispositivos y permitir la comunicación entre ellos.

Para llevar a cabo esta tarea, en la Figura 1 se representa el esquema de comunicación del protocolo MQTT.

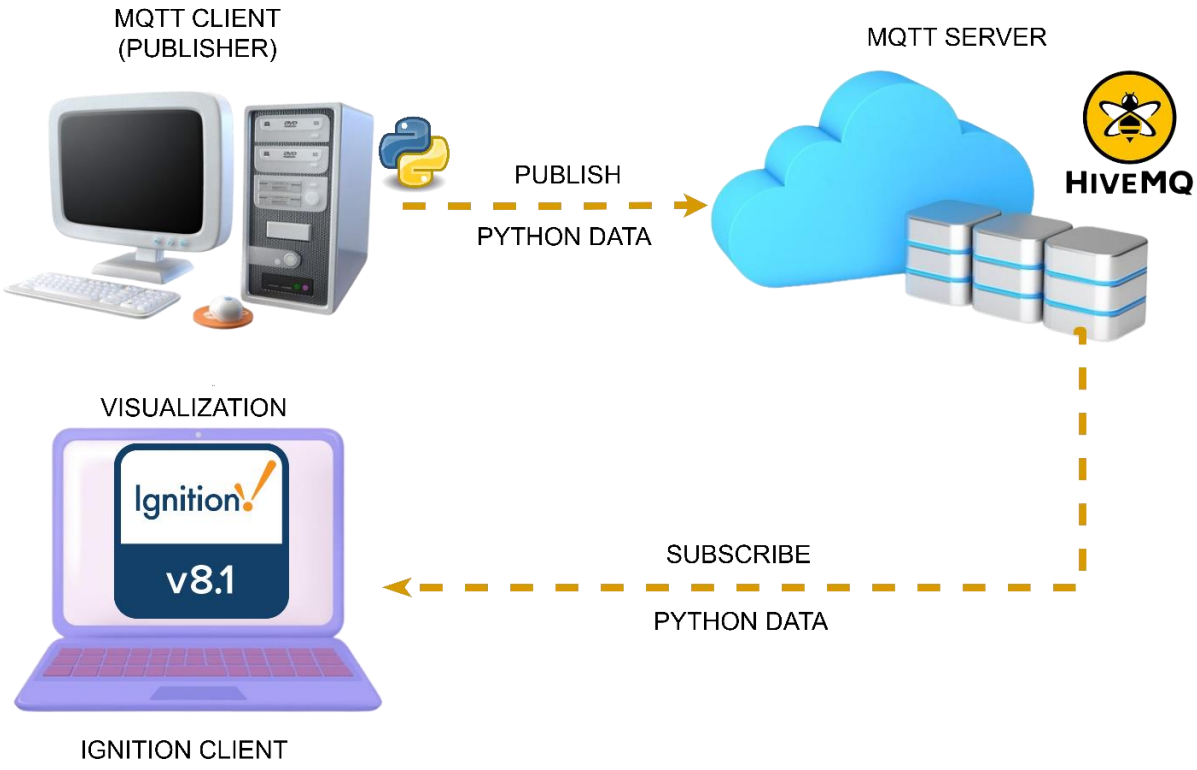


Figura 1. Esquema básico del protocolo MQTT

Conexión de dispositivos

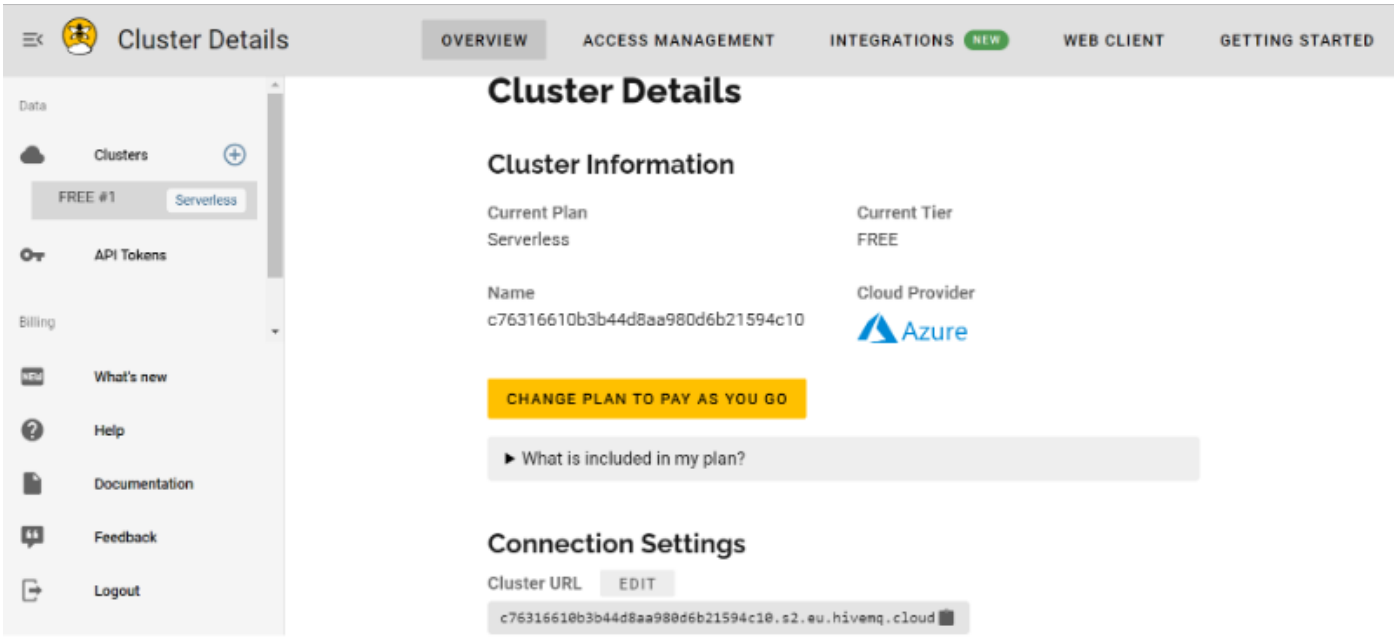
Con el objetivo de validar la capacidad de enviar y recibir mensajes mediante el protocolo MQTT, procederemos a implementar dos roles distintos: un cliente actuando como emisor de datos (publicador) y otro como receptor (suscriptor) utilizando Python. Además, configuraremos un servidor HiveMQ que implementa TLS, permitiendo así la utilización del puerto seguro 8883 para garantizar una conexión segura.

Configuración del servidor

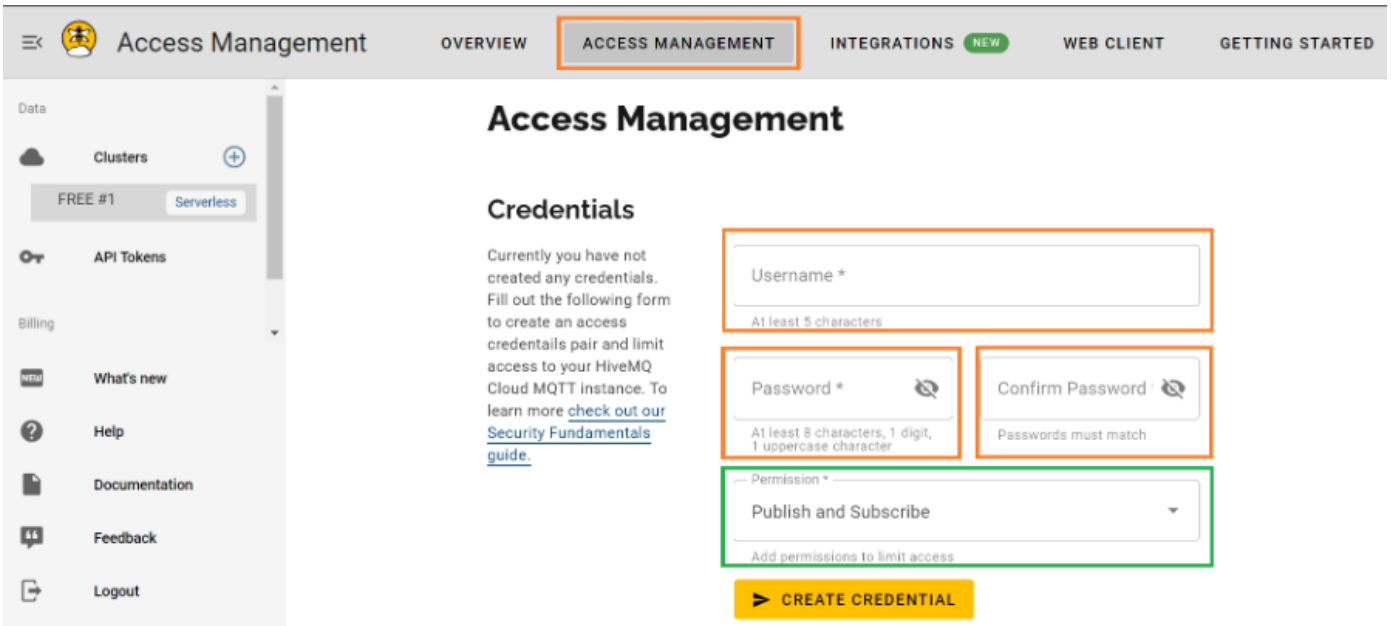
Para esta configuración, haremos uso del servidor HiveMQ Cloud, que nos brinda la capacidad de visualizar los mensajes a través de un cliente web. Para iniciar la configuración del servidor, por favor, siga estos pasos:

1. Acceda al siguiente enlace: HiveMQ Cloud Login
2. Cree una cuenta nueva o acceda utilizando sus credenciales de Google, GitHub o LinkedIn.

Después de completar estos pasos, se espera que aparezca la siguiente pantalla en su interfaz de HiveMQ Cloud.



3. Para iniciar, diríjase a la sección de "Access Management" en el servidor HiveMQ. Aquí, se procede a crear un nuevo usuario y asignarle una contraseña. En las áreas resaltadas en naranja, puede elegir sus propias credenciales de usuario y contraseña. Sin embargo, se recomienda dejar sin cambios la sección resaltada en verde, que está configurada como "Publish and Subscribe". Este paso es crucial, ya que las credenciales especificadas aquí serán la clave para acceder al servidor. Para finalizar dar click en Create Credential.



Creación del publicador

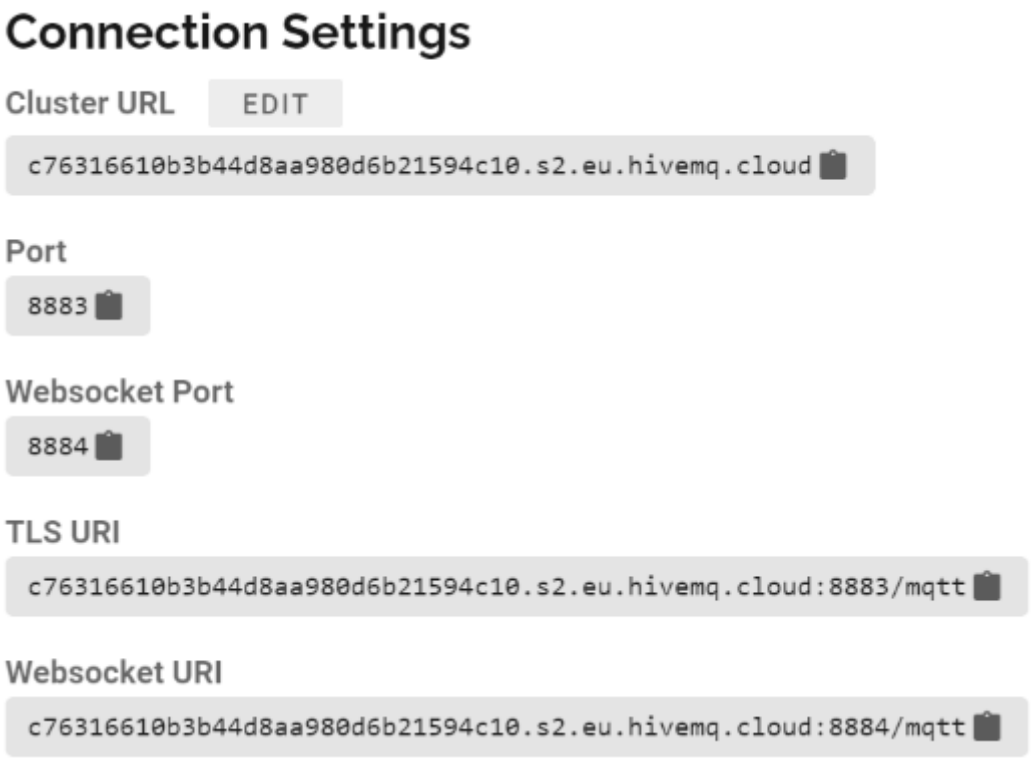
Para la creación del publicador hay varias cosas que debemos tener en cuenta:

- **Configuración del cliente MQTT**

Estos son los parámetros del código que se encuentra más adelante, en la implementación.

- **mqtt_broker**: Se refiere a la dirección del servidor recién creado en HiveMQ Cloud, identificado como la URL del clúster en el servidor.
- **mqtt_port**: Indica el puerto que emplearemos para la conexión; en este caso, el puerto seguro 8883.
- **mqtt_topic**: Representa el tópico al cual dirigiremos los datos para ser enviados al servidor HiveMQ.
- **user**: Corresponde al usuario que creamos durante la configuración del servidor HiveMQ.
- **password**: Es la contraseña generada durante la configuración del servidor HiveMQ.

Esta información está disponible en la sección "Overview" del servidor. Asegúrese de utilizar estos parámetros para establecer una conexión exitosa y comenzar a publicar datos en su servidor HiveMQ Cloud. En la sección "Overview" verá la siguiente información.



- **Implementación del publicador**

A continuación, se muestra un ejemplo del código utilizado para la implementación del publicador.

```
import paho.mqtt.client as mqtt

###################################
# Configuración del cliente MQTT
```

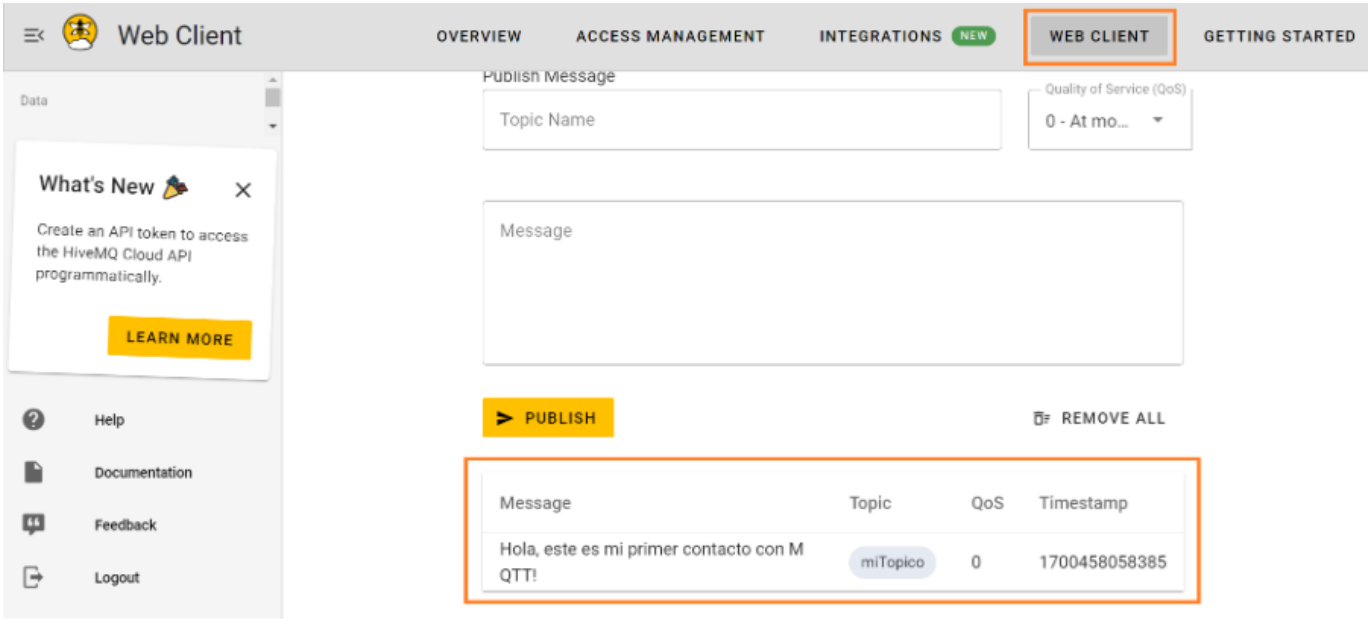
```
#####################################
mqtt_broker = "c76316610b3b44d8aa980d6b21594c10.s2.eu.hivemq.cloud"
mqtt_port = 8883
mqtt_topic = "miTopico"
user = "admin"
password = "Password1"
#####################################

#####################################
# Función de conexión
#####################################
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Conexión exitosa al broker MQTT")
    else:
        print(f"Fallo en la conexión. Código de retorno: {rc}")
#####################################

#####################################
# Callback para manejar la recepción de mensajes (opcional)
#####################################
def on_message(client, userdata, msg):
    print(f"Mensaje recibido en el topic {msg.topic}: {msg.payload.decode()}")
#####################################

#####################################
# Configuración del cliente MQTT
#####################################
client = mqtt.Client()
client.tls_set()  # habilita la conexión por el puerto seguro TLS
client.username_pw_set(user, password)  # establece las credenciales
client.on_connect = on_connect
client.on_message = on_message
#####################################

#####################################
# Conexión al servidor MQTT
#####################################
client.connect(mqtt_broker, mqtt_port, 60)
#####################################

#####################################
# Inicia el bucle para mantener la conexión activa
#####################################
client.loop_start()
#####################################

#####################################
```

```
# Bucle para enviar mensajes
####################################
try:
    # Publica datos en el topic especificado
    while True:
        message = input("Ingrese el mensaje a publicar: ")
        client.publish(mqtt_topic, message)
except KeyboardInterrupt:
    # Detiene el bucle y cierra la conexión al recibir Ctrl + C
    print("\nDesconexión solicitada. Cerrando la conexión...")
    client.disconnect()
####################################
```

En el entorno de HiveMQ, contamos con una herramienta integrada que nos permite acceder a un cliente web, facilitándonos la visualización de los mensajes enviados. Para lograrlo, procedemos de la siguiente manera:

1. Accedemos con las credenciales previamente creadas.
2. Nos suscribimos al tópico específico que deseamos observar.

Este cliente web se encuentra en la sección "Web Client" del servidor HiveMQ. En este contexto, podemos observar que el mensaje "Hola, ¡este es mi primer contacto con MQTT!" fue enviado al tópico "miTopico". Este resultado se evidencia como consecuencia de la ejecución del código anterior.



Creación del suscriptor

Usando los mismos parámetros del publicador, vamos a desarrollar el suscriptor.

- **Implementación del suscriptor**

A continuación, se muestra un ejemplo del código utilizado para la implementación del publicador.

```
import paho.mqtt.client as mqtt
```

```
#####################################
# Configuración del cliente MQTT
#####################################
mqtt_broker = "c76316610b3b44d8aa980d6b21594c10.s2.eu.hivemq.cloud"
mqtt_port = 8883
mqtt_topic = "miTopico"
user = "admin"
password = "Password1"
#####################################

#####################################
# Callback cuando se establece la conexión con el broker MQTT
#####################################
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Conexión exitosa al broker MQTT")
        # Suscripción al tópico deseado después de la conexión exitosa
        client.subscribe(mqtt_topic)
    else:
        print(f"Fallo en la conexión. Código de retorno: {rc}")
#####################################

#####################################
# Callback para manejar la recepción de mensajes
#####################################
def on_message(client, userdata, msg):
    print(f"Mensaje recibido en el topic {msg.topic}: {msg.payload.decode()}")
#####################################

#####################################
# Configuración del cliente MQTT
#####################################
client = mqtt.Client()
client.tls_set()  # Configuración de TLS
client.username_pw_set(user, password)  # Autenticación
client.on_connect = on_connect
client.on_message = on_message
#####################################

#####################################
# Conexión al servidor MQTT con TLS y autenticación
#####################################
client.connect(mqtt_broker, mqtt_port, 60)
#####################################

#####################################
# Bucle para enviar mensajes
```

```
####################################
try:
    # Inicia el bucle para mantener la conexión activa
    client.loop_forever()
except KeyboardInterrupt:
    # Detiene el bucle y cierra la conexión al recibir Ctrl + C
    print("\nDesconexión solicitada. Cerrando la conexión...")
    client.disconnect()
```

Para verificar el funcionamiento adecuado del suscriptor, se pueden realizar las siguientes acciones:

1. Enviar Datos desde el Cliente Web de HiveMQ:

- Utilice el cliente web integrado en el servidor HiveMQ para enviar datos.
- Visualice la recepción de estos datos en el cliente Python del código previamente proporcionado.

2. Utilizar el código del publicador anterior:

- Emplee el código del publicador desarrollado anteriormente para enviar datos desde el cliente Python.
- Observe la correcta visualización de la información en el cliente Python (suscriptor).

De cualquier manera, el resultado esperado es la correcta visualización de la información transmitida a través del servidor HiveMQ, utilizando el tópico especificado. Este proceso garantiza la coherencia y efectividad de la suscripción al servidor.

ACTIVIDAD DE CIERRE (OPCIONAL)

- Implementa un proyecto más avanzado que involucre el uso de sensores o actuadores controlados por MQTT.

CONCEPTOS PARA REVISAR

Antes de avanzar a la siguiente guía, revisa los siguientes conceptos:

- Calidad de servicio (QoS).
- Software Ignition.

REFERENCIAS BIBLIOGRÁFICAS

- Manual de HiveMQ Cloud. Manual HiveMQ Cloud
- Documentación de Paho-MQTT. Paho-MQTT.

ENLACE DE GOOGLE COLAB

https://colab.research.google.com/drive/1fFgs2siSXeMg64CCgOaSlwxSPaQYCGPo?usp=drive_link

Apéndice B. Guía #2

Enlace para acceder a la guía de aprendizaje: https://colab.research.google.com/drive/1f2Iu_5QbJcj_s9EG1_IbZKJTCPbQ0IzY?usp=drive_link

PRÁCTICA # 05

DESARROLLO DE UN SISTEMA IOT CON IGNITION

Diseñada por: Felipe Rubio (E3T)
Revisada por: Juan Manuel Rey (E3T)

OBJETIVOS DE APRENDIZAJE

Al finalizar la guía, se espera que el estudiante sea capaz de:

- Comprender el funcionamiento de los Tags en Ignition.
- Desarrollar una interfaz gráfica para interactuar con un sistema IoT.
- Asociar los Tags a valores simulados utilizando Python.
- Almacenar datos generados por el sistema IoT en una base de datos.

CONOCIMIENTOS PREVIOS REQUERIDOS

El estudiante debe tener conocimientos básicos de programación en Python, comprensión de conceptos de bases de datos y familiaridad con el entorno de desarrollo Ignition.

INTRODUCCIÓN

En esta guía, exploraremos los conceptos básicos de Ignition para el desarrollo de sistemas IoT. El objetivo es que, al finalizar, el estudiante pueda crear una aplicación funcional que interactúe con dispositivos IoT simulados.

MATERIALES

- Computadora con Ignition instalado.
- Acceso a una base de datos para almacenar los datos generados.
- Conexión a Internet.

ACTIVIDADES PRELIMINARES

- **Instalación de Ignition**: Antes de comenzar con las actividades prácticas, es esencial llevar a cabo la instalación de Ignition en tu entorno de desarrollo. Para ello, te recomendamos seguir estos pasos:
    1. Crea tu cuenta en Inductive University: Regístrate en Inductive University para acceder a una variedad de recursos, incluidos videos explicativos sobre las diversas funcionalidades de Ignition.
    2. Accede a los videos informativos: Una vez creada tu cuenta, explora la extensa biblioteca de videos explicativos disponibles en Inductive University. Estos videos te ofrecerán una comprensión más profunda de las capacidades de Ignition.
    3. Inicia la instalación: Para comenzar con la instalación de Ignition, te recomendamos seguir los pasos detallados en el siguiente video: Instalación de Ignition. Este video proporciona instrucciones visuales claras y concisas, asegurando una instalación sin contratiempos. Es fundamental seguir cada paso atentamente para configurar Ignition de manera óptima en tu sistema. Una instalación exitosa establecerá las bases necesarias para el desarrollo efectivo de las actividades siguientes.
    Además, es necesario instalar el Designer Launcher, una herramienta de Ignition que facilita la creación de interfaces gráficas. Puedes encontrarlo en la sección "Home" una vez que hayas completado la instalación de Ignition. Para esto te recomendamos seguir los pasos del siguiente video: Designer Launcher.

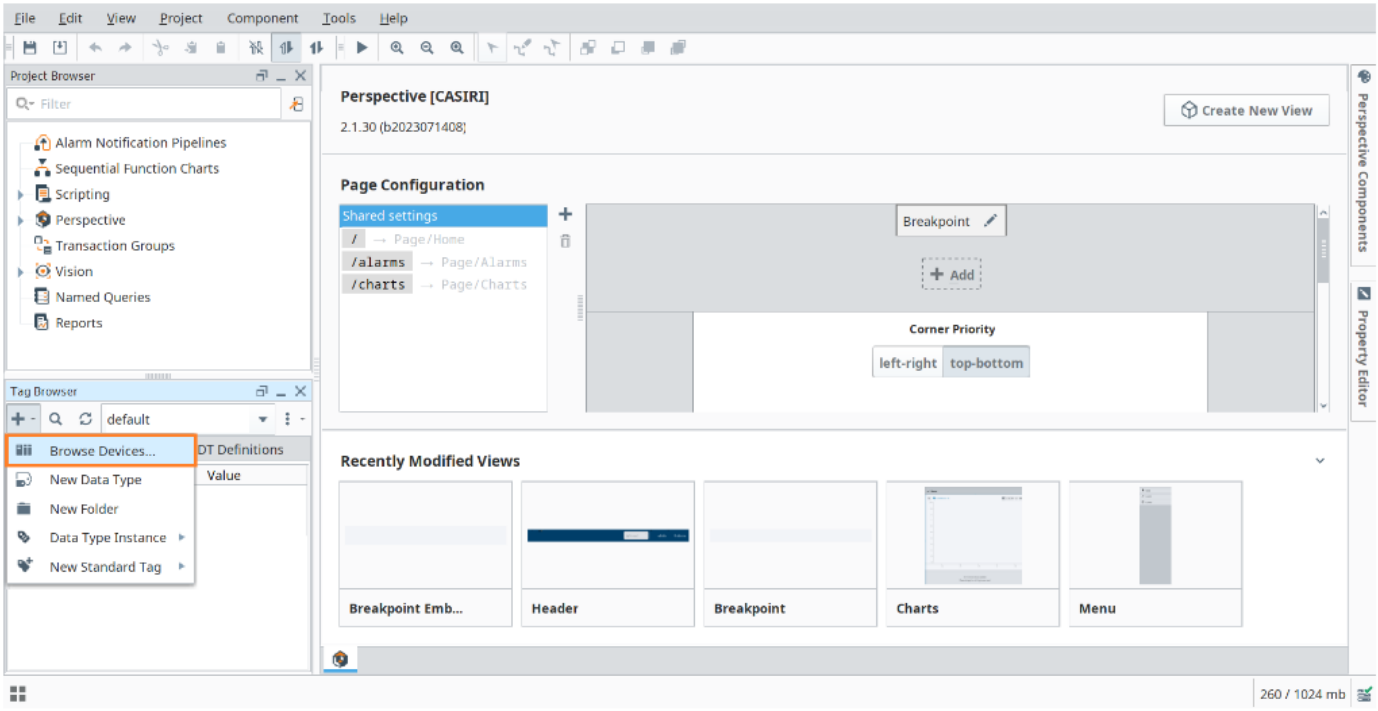
- **Preparación de la base de datos**: Junto con la instalación de Ignition, es imperativo configurar un sistema de gestión de bases de datos, como MySQL o MariaDB. A continuación, te proporcionamos recursos útiles para este proceso:
    1. Instalación del sistema de gestión de bases de datos:
        - Si optas por MySQL, sigue las instrucciones detalladas en el siguiente video: Instalación de MySQL.
        - Para aquellos que prefieren trabajar con MariaDB, el siguiente video ofrece una guía completa: Instalación de MariaDB.
    2. Conexión de la base de datos con Ignition:
        - Si has elegido MySQL, este video te guiará en la conexión: Conexión a MySQL.
        - Para quienes optaron por MariaDB, aquí está la guía para la conexión: Conexión a MariaDB.

    Estos recursos visuales proporcionan pasos específicos para establecer una conexión fluida entre Ignition y tu sistema de gestión de bases de datos elegido. Asegúrate de seguir cada detalle cuidadosamente, garantizando así una preparación eficaz para las fases posteriores de desarrollo.

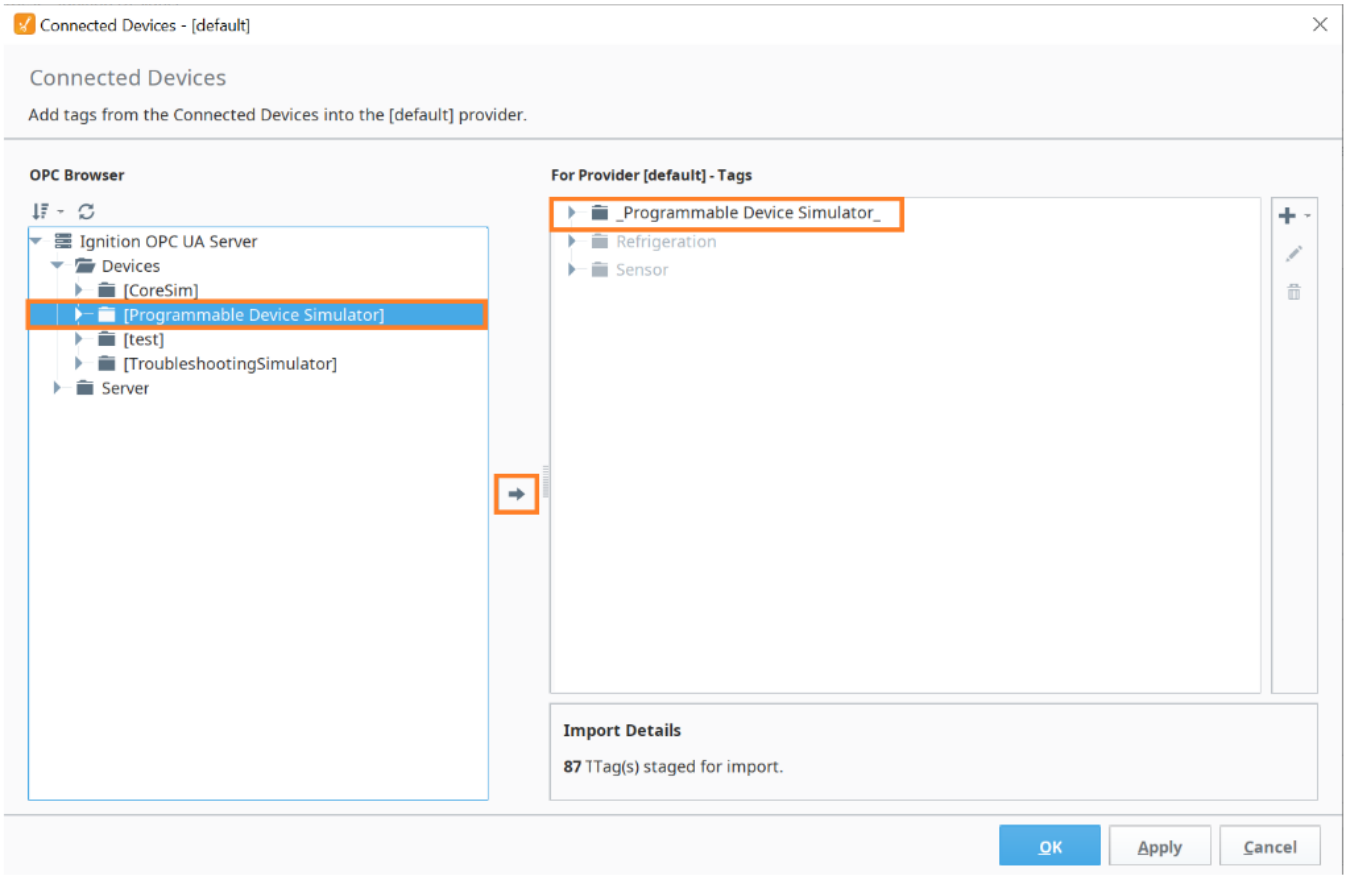
ACTIVIDAD 1: COMPRENDER EL FUNCIONAMIENTO DE LOS TAGS

1. Visualiza el video a continuación para comprender en qué consisten los tags: Tags en Ignition.
2. Ejercicio práctico: Vincular tags con componentes de la interfaz gráfica.

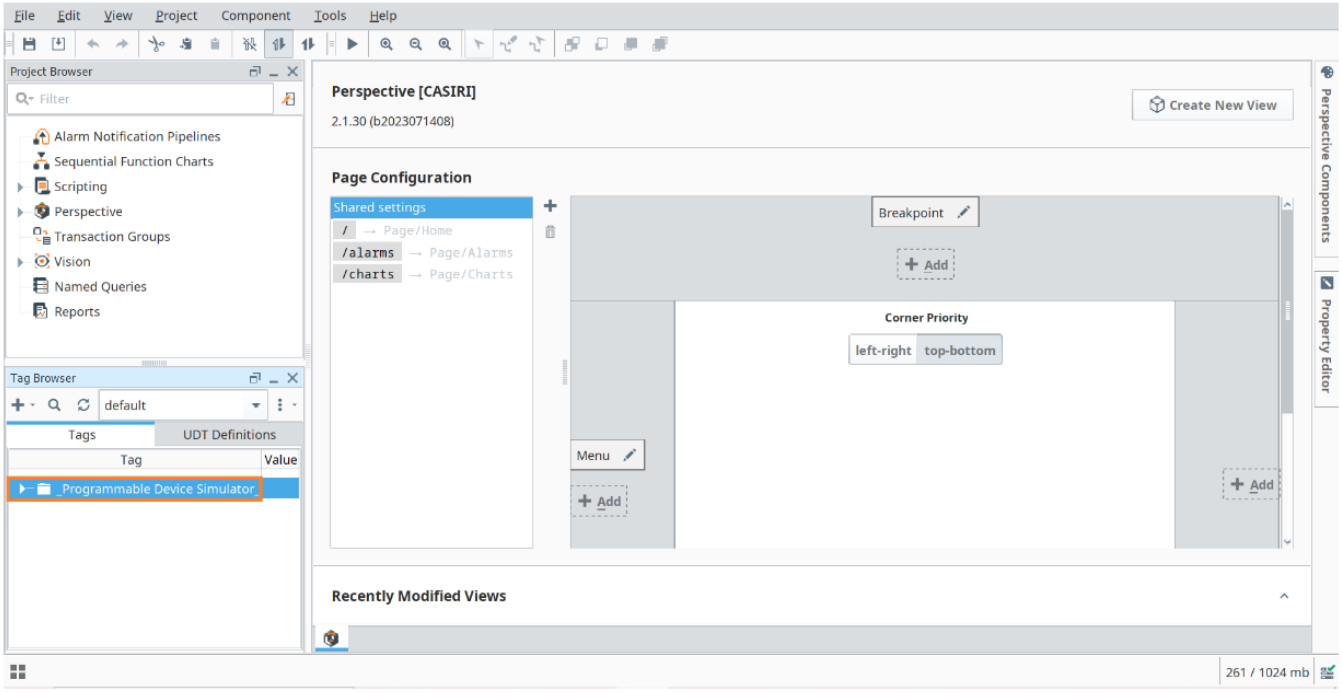
En este ejercicio, exploraremos la vinculación de tags a los componentes de la interfaz gráfica mediante el uso del Simulador de Dispositivo Programable (Programmable Device Simulator) de Ignition. Para comprender cómo operan los tags, simularemos su funcionamiento con esta herramienta específica. Para llevar a cabo esta simulación, te invitamos a seguir detenidamente las instrucciones proporcionadas en el siguiente video: Programmable Device Simulator.
Después de seguir los pasos del video, es importante realizar algunas configuraciones para que los tags aparezcan en el Designer Launcher de Ignition. Para lograr esto, el primer paso es abrir la herramienta Designer Launcher. Luego, dirígete al Tag Browser en la parte izquierda y haz clic en "Browse Devices", tal como se ilustra a continuación.



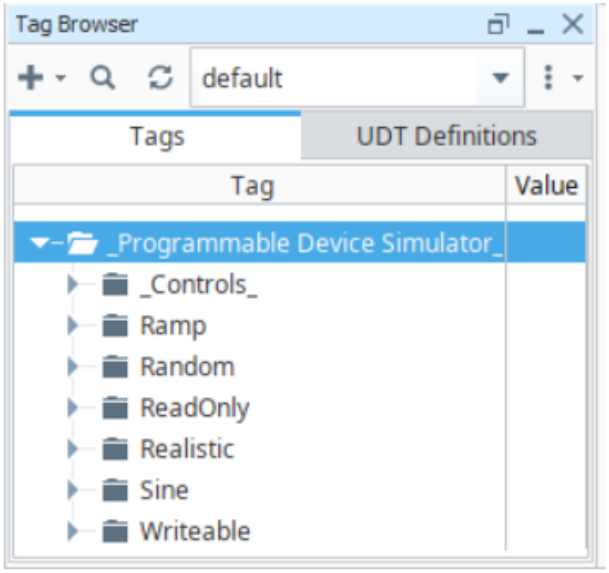
Aparecerá una ventana titulada "Connected Devices". En este punto, es necesario expandir el menú "Ignition OPC UA Server" y, posteriormente, ampliar el menú "Devices". Aquí localizaremos el nuevo dispositivo que pretendemos añadir. Debemos buscar el nombre que asignamos previamente, detallado en el video de configuración del Programmable Device Simulator. Lo seleccionamos en la parte izquierda, damos click en la flecha y aparece en la parte derecha. En este caso, el nombre es [Programmable Device Simulator], como se ilustra en la imagen.



Una vez completados estos pasos, los tags deben aparecer en el Tag Browser, y ahora ya están listos para ser utilizados en la interfaz.



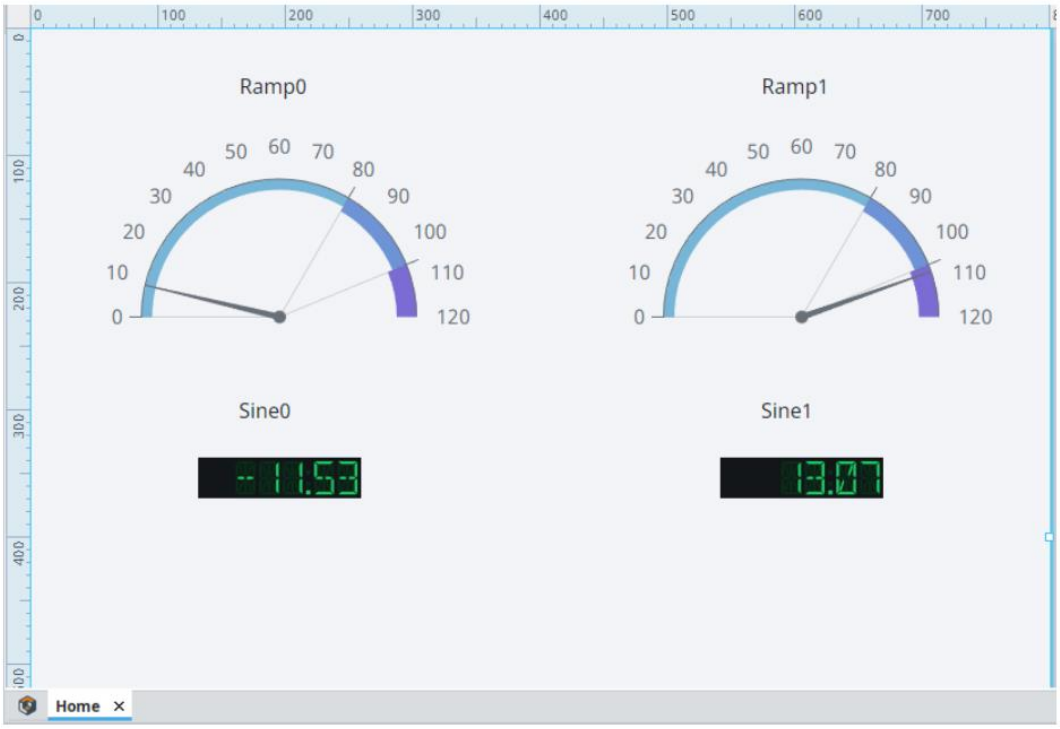
Al hacer click para desplegar en el *Programmable Device Simulator*, se presenta una lista con los tags previamente importados en el simulador. La lista incluye los siguientes tags:



En este momento, los tags ya están disponibles para ser utilizados en la interfaz gráfica de Ignition. Ahora procederemos a emplear componentes de la interfaz para visualizar de manera gráfica los valores de estos tags. Para lograrlo, usaremos Perspective (que es una herramineta de Ignition para crear interfaces gráficas) y crearemos una nueva página. Puedes seguir este procedimiento con la asistencia del siguiente video: Ignition Perspective

Para concluir nuestro ejercicio de vinculación de tags con componentes de la interfaz gráfica, emplearemos algunos componentes disponibles en la paleta "Perspective Components". Este proceso será guiado con la ayuda del siguiente video: Vinculación de tags

Como práctica, siguiendo la explicación anterior, vincula los tags Ramp0, Ramp1, Sine0 y Sine1, en las siguientes ubicaciones: "Programmable Device Simulator/Ramp/Ramp0", "Ramp1", y "Programmable Device Simulator/Sine/Sine0", "Sine1". Utiliza etiquetas "Label" para indicar el nombre del tag (Ramp0, Ramp1...) y también indicadores de tipo "Gauge" y "LED", tal como se ilustra en la figura.



ACTIVIDAD 2: ASOCIAR LOS TAGS A VALORES SIMULADOS EN PYTHON

1. Creación de un script en Python para simular lecturas de sensores.

Con el objetivo de representar gráficamente variables, crearemos un script en Python que generará valores aleatorios y los enviará al servidor configurado en la Guía 1 (Servidor HiveMQ).
NOTA: Puedes utilizar como base el código del publicador empleado previamente en la Guía 1.

Es crucial destacar que los cambios con respecto al publicador de la Guía 1 son los siguientes:

- No estamos definiendo un mensaje específico a enviar; en cambio, publicamos constantemente un valor aleatorio.
- Es fundamental transmitir los datos de las variables en formato JSON para permitir el acceso individual a cada una de ellas.

En el siguiente código, observarás una implementación de los conceptos mencionados anteriormente.

En este caso las variables simuladas son:

- Temperatura
- Humedad
- Presión
- Velocidad del viento

```
import paho.mqtt.client as mqtt
import random
import json
import time

#####################################
# Configuración del cliente MQTT
#####################################
mqtt_broker = "c76316610b3b44d8aa980d6b21594c10.s2.eu.hivemq.cloud"
mqtt_port = 8883
mqtt_topic = "miTopico"
user = "admin"
password = "Password1"
#####################################

#####################################
# Función de conexión
#####################################
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Conexión exitosa al broker MQTT")
    else:
        print(f"Fallo en la conexión. Código de retorno: {rc}")
#####################################

#####################################
```

```
# Callback para manejar la recepción de mensajes (opcional)
####################################
def on_message(client, userdata, msg):
    print(f"Mensaje recibido en el topic {msg.topic}:
{msg.payload.decode()}")
####################################

####################################
# Configuración del cliente MQTT
####################################
client = mqtt.Client()
client.tls_set()  # habilita la conexión por el puerto seguro TLS
client.username_pw_set(user, password)  # establece las credenciales
client.on_connect = on_connect
client.on_message = on_message
####################################

####################################
# Conexión al servidor MQTT
####################################
client.connect(mqtt_broker, mqtt_port, 60)
####################################

####################################
# Inicia el bucle para mantener la conexión activa
####################################
client.loop_start()
####################################

####################################
# Bucle para enviar mensajes
####################################
try:
    while True:
        # Genera datos aleatorios simulando valores de temperatura, presión,
humedad y velocidad del viento
        temperatura = round(random.uniform(20, 30), 2)
        presion = round(random.uniform(1000, 1010), 2)
        humedad = round(random.uniform(40, 60), 2)
        velocidad_viento = round(random.uniform(0, 10), 2)

        # Construye el objeto JSON con los datos simulados
        data = {
            "temperatura": temperatura,
            "presion": presion,
            "humedad": humedad,
            "velocidad_viento": velocidad_viento
        }
```

```
        json_message = json.dumps(data)

        # Publica datos en el topic especificado
        client.publish(mqtt_topic, json_message)

        # Espera 5 segundos antes de enviar el próximo mensaje (puedes
ajustar este valor)
        time.sleep(5)

except KeyboardInterrupt:
    # Detiene el bucle y cierra la conexión al recibir Ctrl + C
    print("\nDesconexión solicitada. Cerrando la conexión...")
    client.disconnect()
```

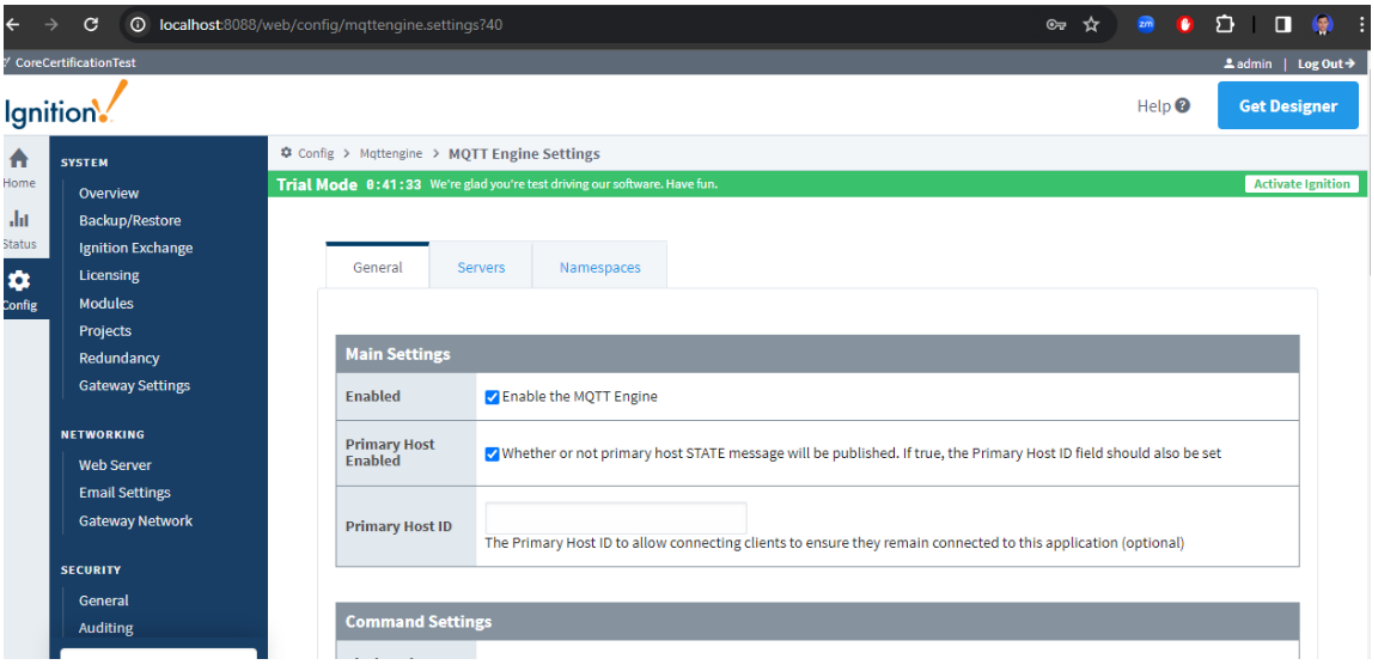
2. Configuración del módulo MQTT Engine.

Después de haber creado el script para generar y enviar datos aleatorios de variables al servidor, el siguiente paso es instalar y configurar el módulo de Ignition que facilita la comunicación a través de MQTT, conocido como el módulo MQTT Engine.

Para llevar a cabo esta instalación, sigue estos pasos:

1. Descarga el MQTT Engine Module desde el siguiente enlace: Ignition Modules.
2. Consulta el siguiente video, que proporciona una guía detallada sobre cómo instalar el módulo que acabas de descargar. Instalar Módulos en Ignition

Tras la instalación del módulo MQTT Engine, es necesario configurarlo para establecer la conexión con el servidor y los tópicos que deseamos visualizar en la interfaz. Para realizar esta configuración, dirigimos nuestra atención a la sección "Config" en el gateway, donde notaremos la creación de una nueva configuración denominada "MQTT Engine". Al hacer clic en "Settings", deberíamos visualizar una ventana similar a la mostrada en la figura adjunta.



El siguiente paso consistirá en dirigirnos a la pestaña "Servers", luego a "Settings", y hacer click en "Create new MQTT Server Setting". De esta manera, crearemos un nuevo servidor MQTT, en el cual especificaremos la URL y posteriormente, los tópicos a los que nos suscribiremos.

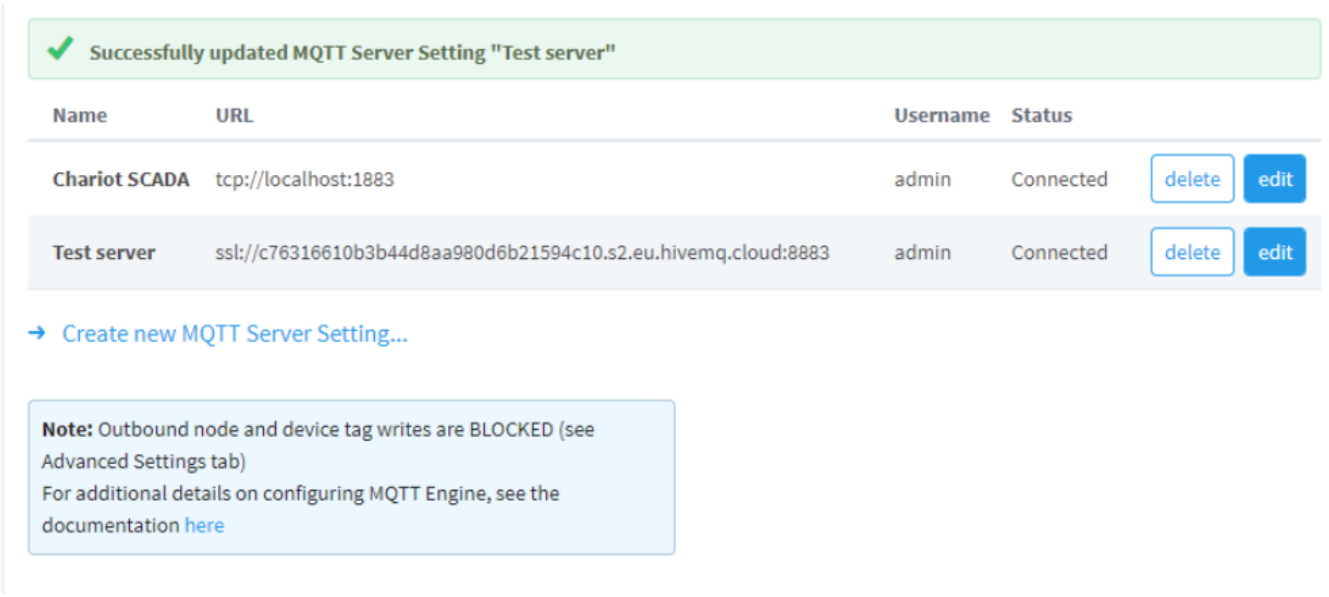
Los parámetros para configurar son los siguientes:

- **Name**: Es el nombre que asignaremos al servidor.
- **URL**: Representa la dirección web del servidor. Aquí debemos ingresar la URL del servidor creado en la Guía 1. Es importante destacar que debe tener el siguiente formato: ssl://c76316610b3b44d8aa980d6b21594c10.s2.eu.hivemq.cloud:8883, donde:
    - ssl://: Indica que se utilizará SSL/TLS para la capa de seguridad en la comunicación. Esto es esencial para cifrar y proteger los datos transmitidos entre el cliente MQTT (en este caso, Ignition) y el servidor MQTT.
    - c76316610b3b44d8aa980d6b21594c10.s2.eu.hivemq. cloud: Es la dirección del servidor MQTT al que te estás conectando. Este es el nombre de dominio o la dirección IP del servidor.
    - 8883: Es el número de puerto utilizado para la comunicación segura mediante SSL/TLS. En este caso, el puerto 8883 es comúnmente utilizado para conexiones MQTT seguras.

NOTA: La única modificación necesaria es la dirección del servidor MQTT; los demás parámetros no deben ser modificados.

- **Username**: Corresponde al nombre de usuario con el que nos conectaremos al servidor. Debe coincidir con el usuario creado en la Guía 1 (Servidor HiveMQ).
- **Password**: Es la contraseña del usuario utilizado para la conexión. También debe ser idéntica a la creada en la Guía 1.

Concluimos haciendo click en "Save Changes", y así se creará el nuevo servidor MQTT en el gateway. Si todas las etapas del proceso se ejecutaron correctamente, el resultado debería ser un nuevo servidor con el estado "Connected", tal como se ilustra en la imagen, donde se creó el servidor llamado "Test server".

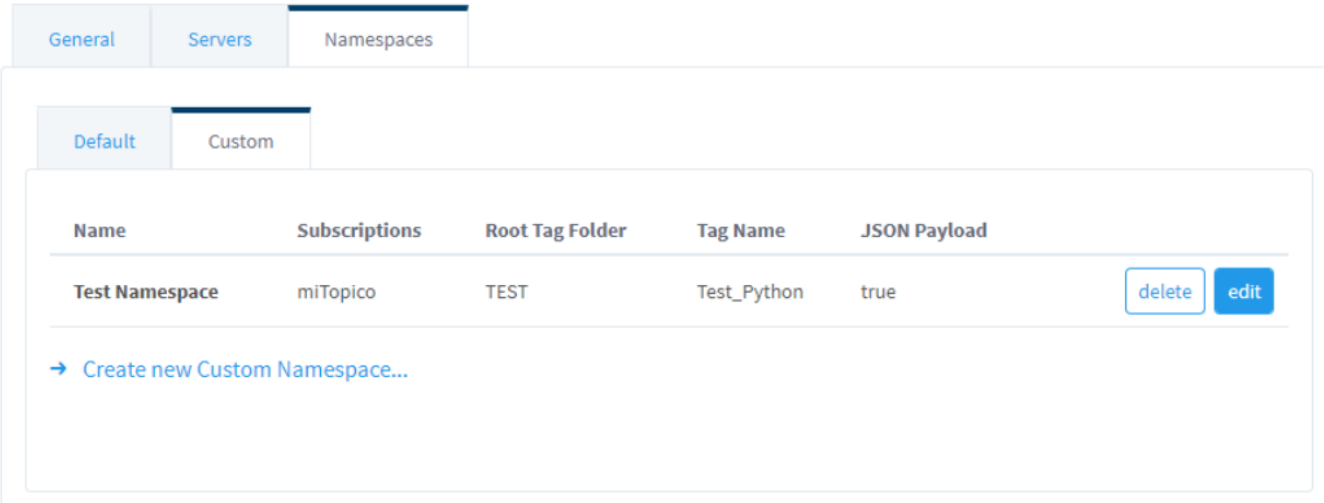


El siguiente paso implica dirigirse a la pestaña "Namespaces", luego a "Custom", y hacer clic en "Create new Custom Namespace". Esto se realiza para especificar los tópicos a los cuales nos suscribiremos y para crear los nombres de los tags en el Designer.

Los parámetros para configurar son los siguientes:

- **Name**: Es el nombre que asignaremos al espacio de nombres.
- **Subscriptions**: Son los tópicos a los cuales nos suscribiremos.
- **Root Tag Folder**: Es el nombre del directorio raíz en el Designer.
- **Tag Name**: Es el nombre del tag, es decir, donde se desplegarán las variables enviadas.
- **JSON Payload**: Envía la carga útil en formato JSON. Para este caso, debe estar seleccionado, ya que, de lo contrario, las variables llegarían como una cadena de caracteres y no sería posible separarlas.

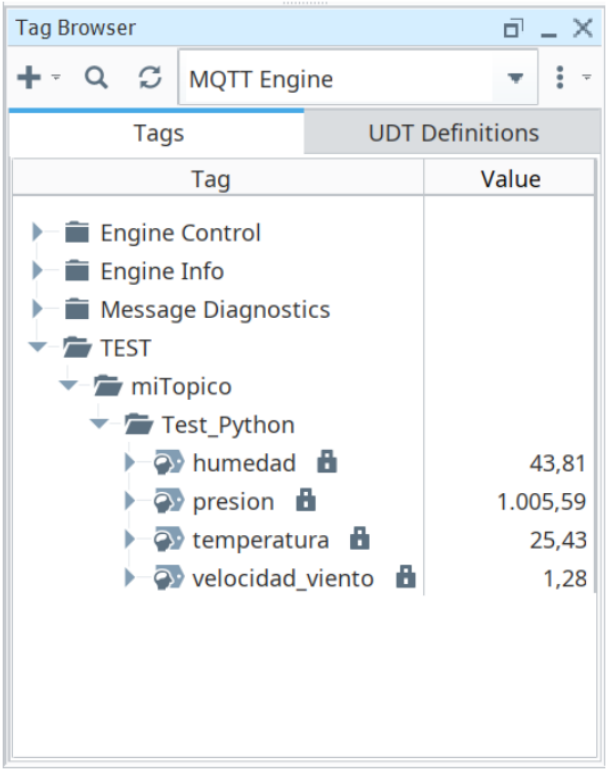
Finalmente, al hacer clic en "Create New Custom Namespace", si todo el proceso fue llevado a cabo correctamente, el resultado será similar al mostrado en la imagen.



3. Verificación de la actualización en tiempo real de los tags en la interfaz.

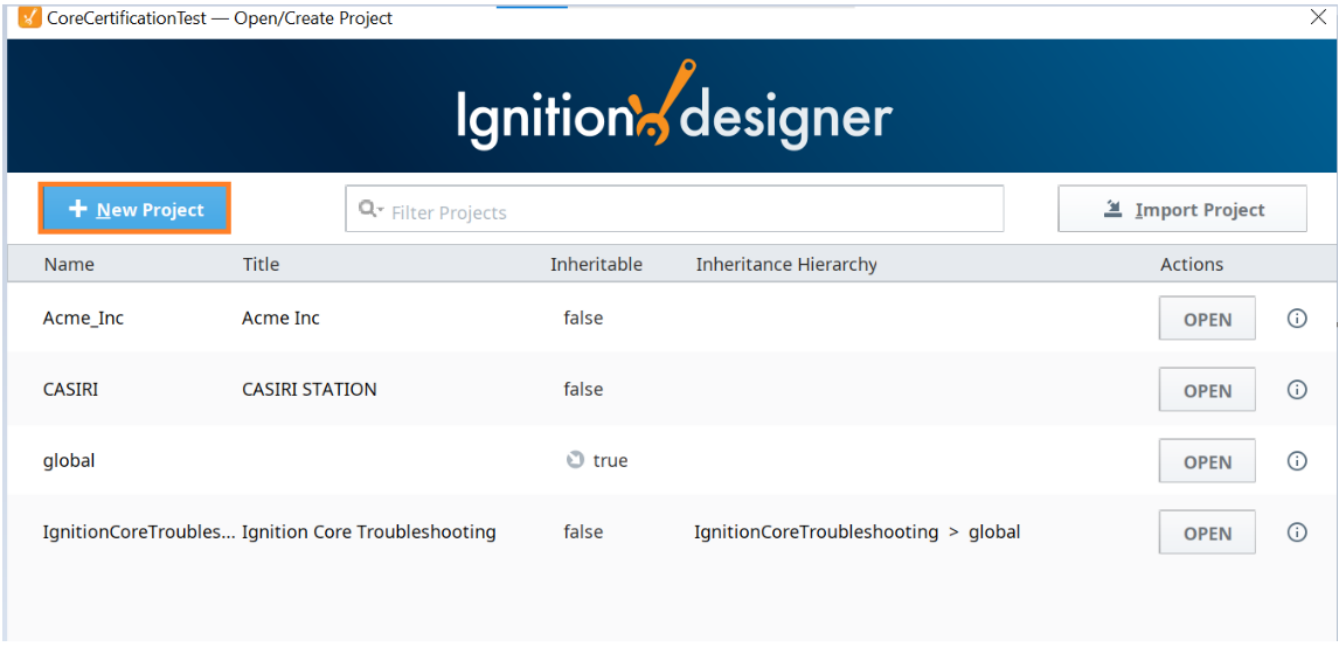
Ahora, procederemos a verificar que los valores simulados que estamos enviando desde Python se reflejen como tags en el Ignition Designer. Para lograr esto, ejecutamos el código de Python, y en el Designer deberá aparecer un nuevo tag; en este caso, "TEST", que fue el nombre asignado en la configuración anterior. A continuación, se presentan el tópico al que nos suscribimos, en este caso, "miTopico", y, por último, el nombre del tag, que en este caso es "Test_Python" (donde se listan todas las variables enviadas). Este proceso se ilustra en la imagen a continuación.

NOTA: Los valores de las variables asociadas al tag "Test_Python" deberían experimentar cambios, ya que se generan de forma aleatoria y se envían continuamente hasta que se detiene la ejecución del script de Python.



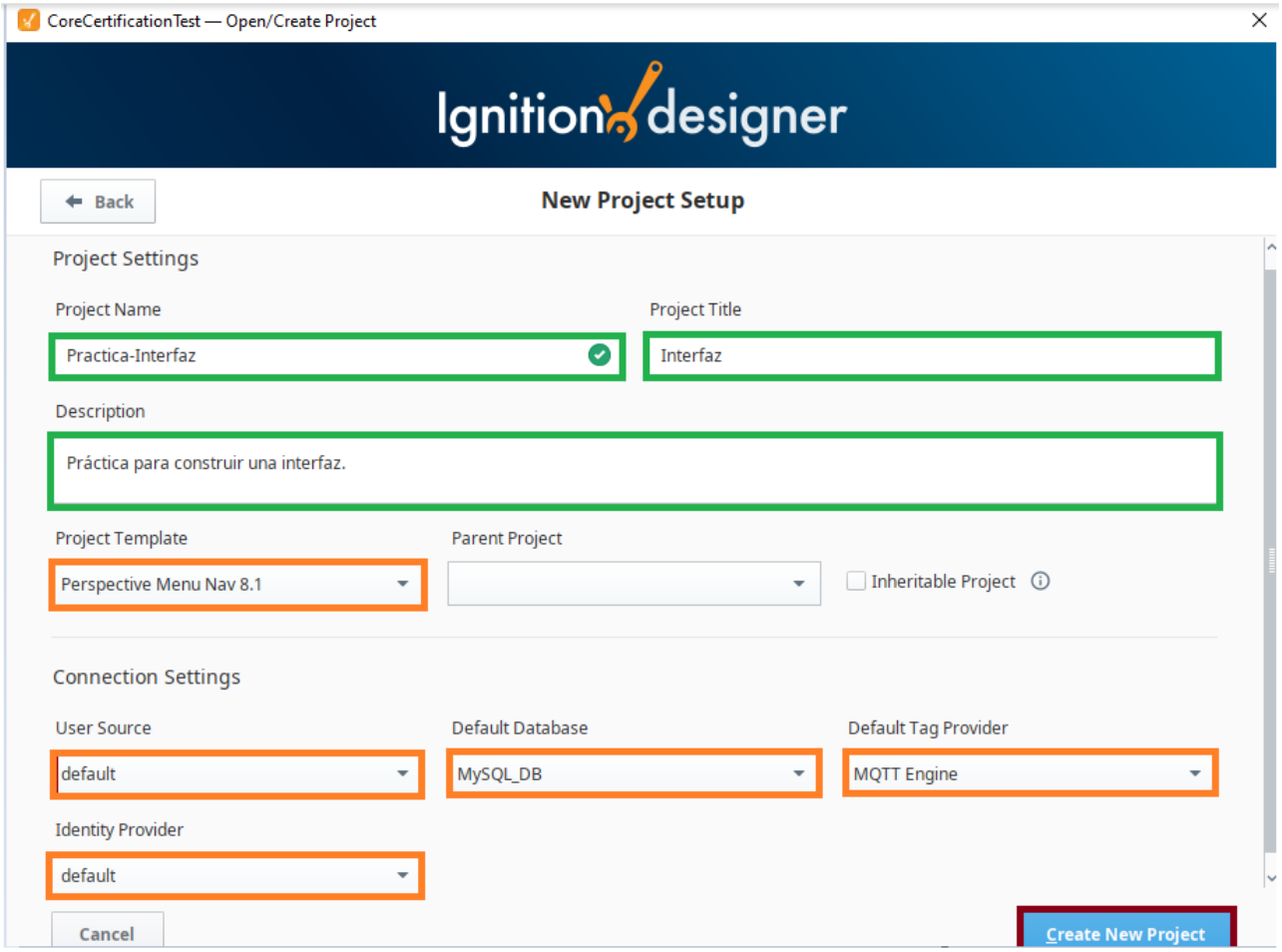
ACTIVIDAD 3: DESARROLLAR UNA INTERFAZ GRÁFICA

Tras adquirir familiaridad con los tags, procederemos a construir una interfaz gráfica utilizando Ignition. Para ello, iniciaremos creando un nuevo proyecto. Al acceder al Designer, haremos clic en "New Project", según se muestra en la imagen.



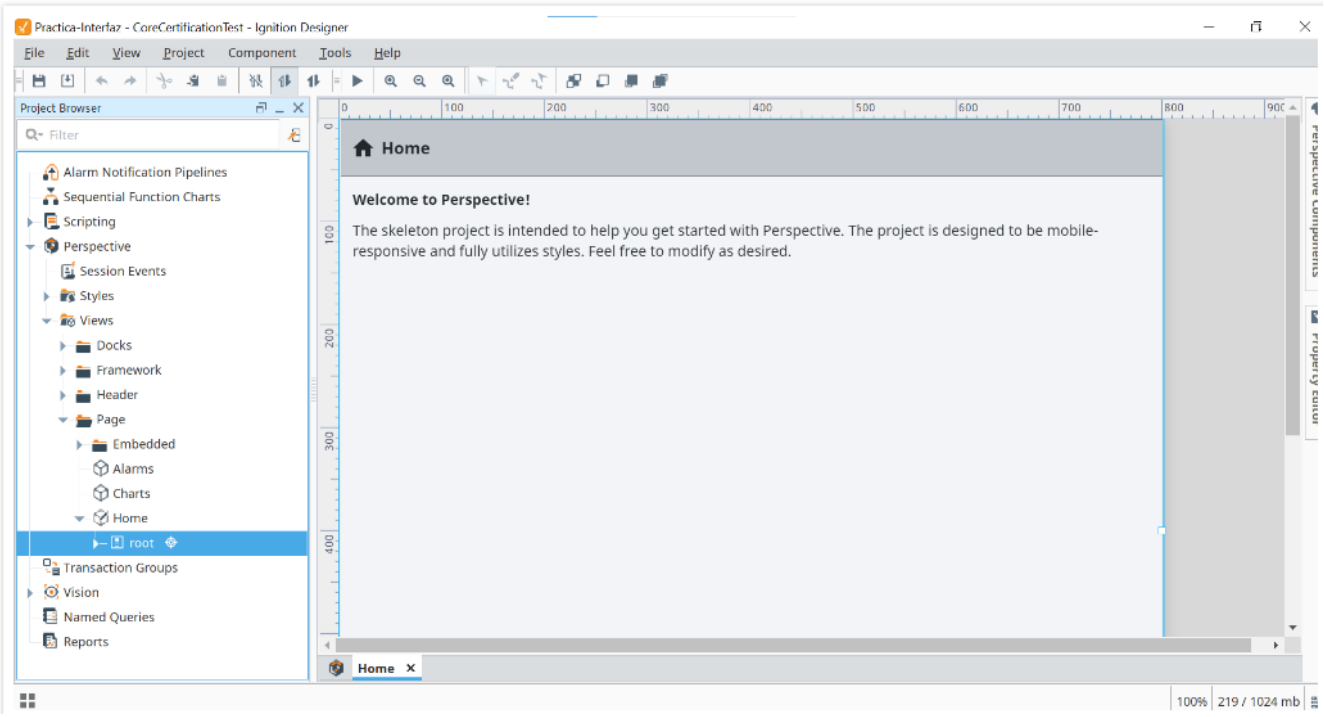
A continuación, aparecerá la ventana que se muestra en la imagen adjunta. Es esencial destacar que las secciones resaltadas en verde son editables, mientras que las marcadas en naranja se sugieren dejar sin modificar; no obstante, no hay inconveniente en realizar cambios en ellas. Finalmente damos click en "Create New Project". Los campos que se encuentran son los siguientes:

- **Project name**: Nombre del proyecto.
- **Project title**: Título de la pestaña cuando se visualiza en el navegador.
- **Description**: Descripción del proyecto.
- **Project template**: Utilizar una plantilla predeterminada, en este caso, diseñada para navegación con menú.
- **User source**: Fuente de usuarios utilizada para gestionar y autenticar usuarios en el sistema. Puedes obtener más información en el siguiente video: User source Ignition
- **Default Database**: La base de datos que se utilizará por defecto. En este caso, seleccionamos la base de datos que creamos previamente.
- **Default Tag Provider**: El proveedor de tags por defecto. En este caso, utilizamos MQTT Engine, que nos permite la comunicación con el servidor MQTT.
- **Identity Provider**: Hace referencia a una fuente externa encargada de verificar la identidad de los usuarios antes de permitirles el acceso al sistema. Puedes obtener más detalles en el siguiente video: Identity Providers Ignition



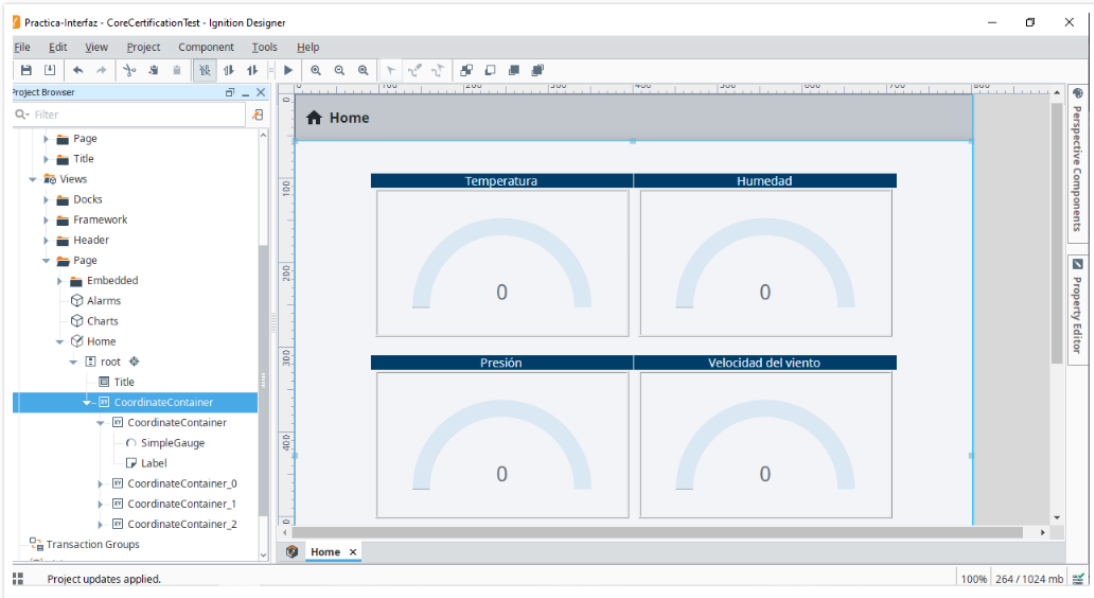
Después de configurar la interfaz de esta manera, podemos notar en el "Project Browser" que en la pestaña "Perspective" se generan algunos elementos como "Styles" y "Views". En este contexto, nos concentraremos en la categoría "Views", que alberga los componentes de la interfaz gráfica. Dentro de "Views", desplegamos la opción "Page" y procedemos a editar la

sección "Home", que representa la vista principal de la interfaz; es decir, la primera vista que se visualiza al ejecutar la interfaz en un navegador web. Esto se ilustra en la imagen adjunta.



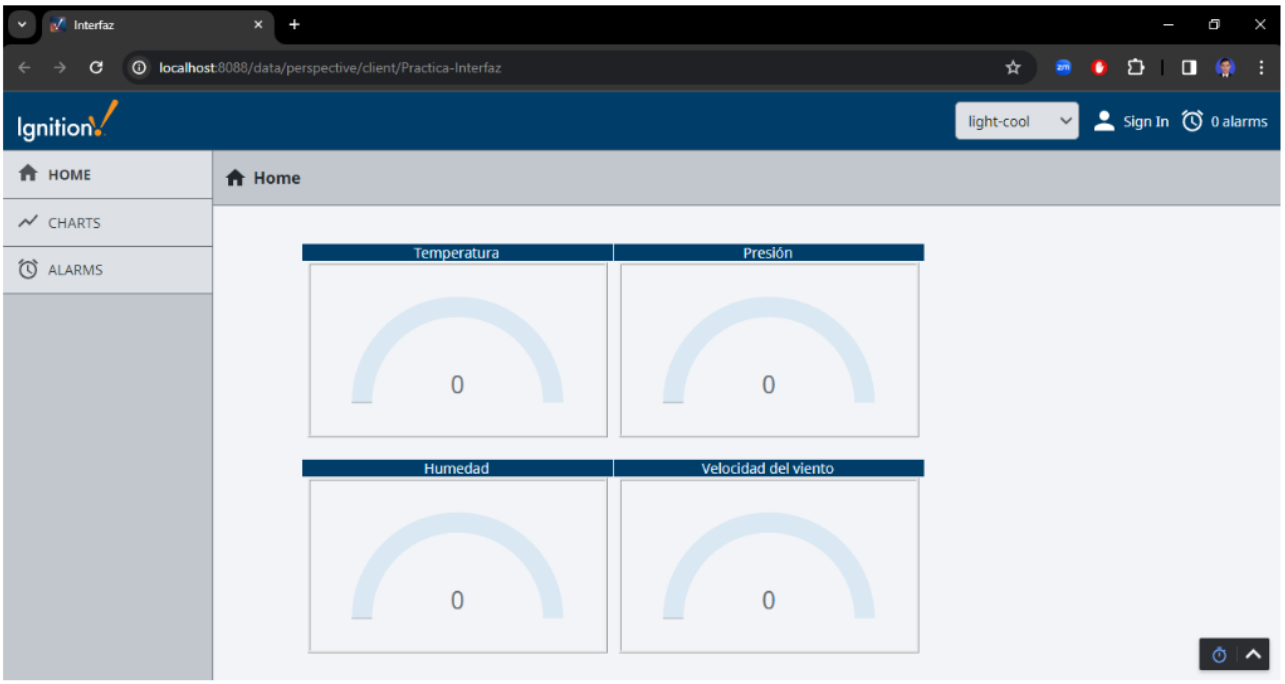
Para editar la sección "Home", procedemos eliminando todos los componentes existentes en su interior, conservando únicamente el título y el componente "Coordinate Container". Luego, insertamos un nuevo componente del tipo "Coordinate Container" y realizamos una selección profunda ("Deep Select") haciendo doble clic sobre el componente recién añadido. Posteriormente, colocamos un componente del tipo "Simple Gauge", el cual utilizaremos para la visualización, y un componente del tipo "Label" para indicar la variable. Repetimos este proceso tres veces más, ya que, en este caso, simularemos la captura de valores de temperatura, humedad, presión y velocidad del viento. El resultado final debería asemejarse a lo mostrado en la imagen de referencia.

NOTA: Para obtener ayuda adicional con el "Coordinate Container", te recomendamos consultar el siguiente video: [Coordinate Container]

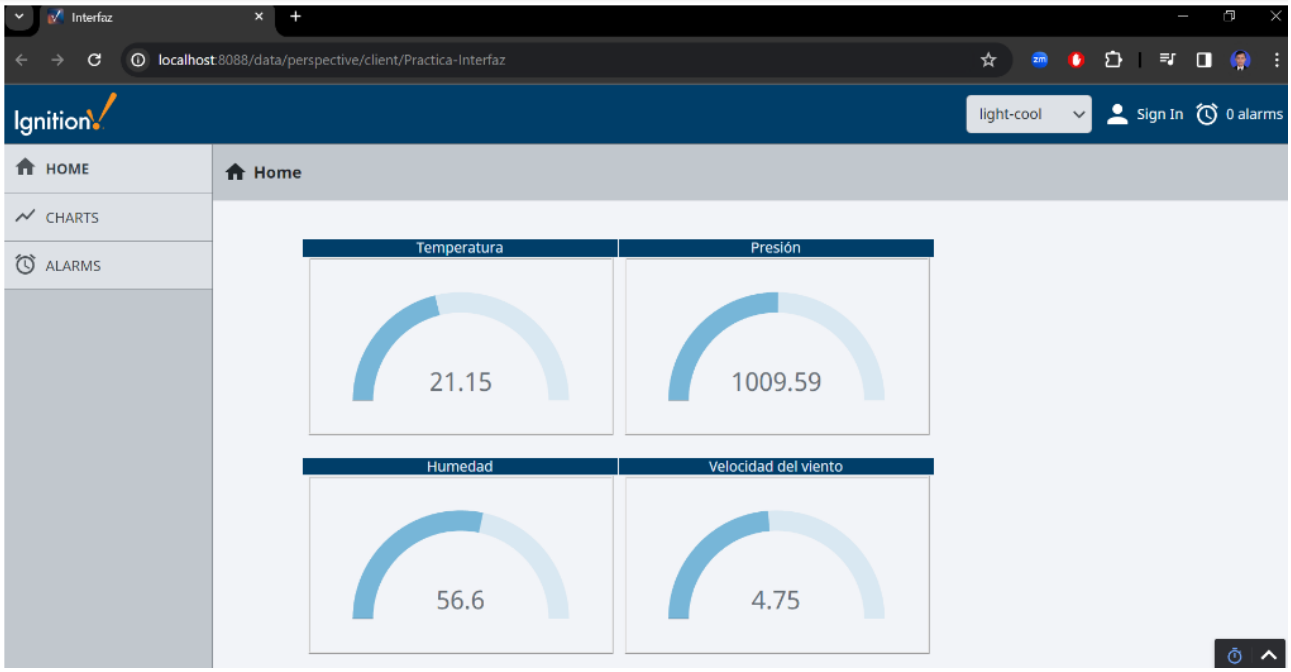


Para visualizar en el navegador web, vamos a la barra de herramientas y seleccionamos la opción "Tools", luego "Launch Perspective" y, por último, "Launch Session". También, puedes realizar este paso presionando la tecla F10. Posterior a esta acción, nuestra interfaz se despliega en el navegador web. La imagen adjunta proporciona una representación visual de ésta.

NOTA: Es importante resaltar que antes de ejecutar la sesión en el navegador se deben guardar los cambios para que se puedan visualizar.



Al completar la interfaz, procedemos a vincular los tags con los componentes de la interfaz gráfica. Luego, ejecutamos la sesión en el navegador para obtener un resultado similar a lo que se muestra en la figura.



En la Figura 1 se observa la arquitectura del protocolo MQTT con un cliente Ignition.

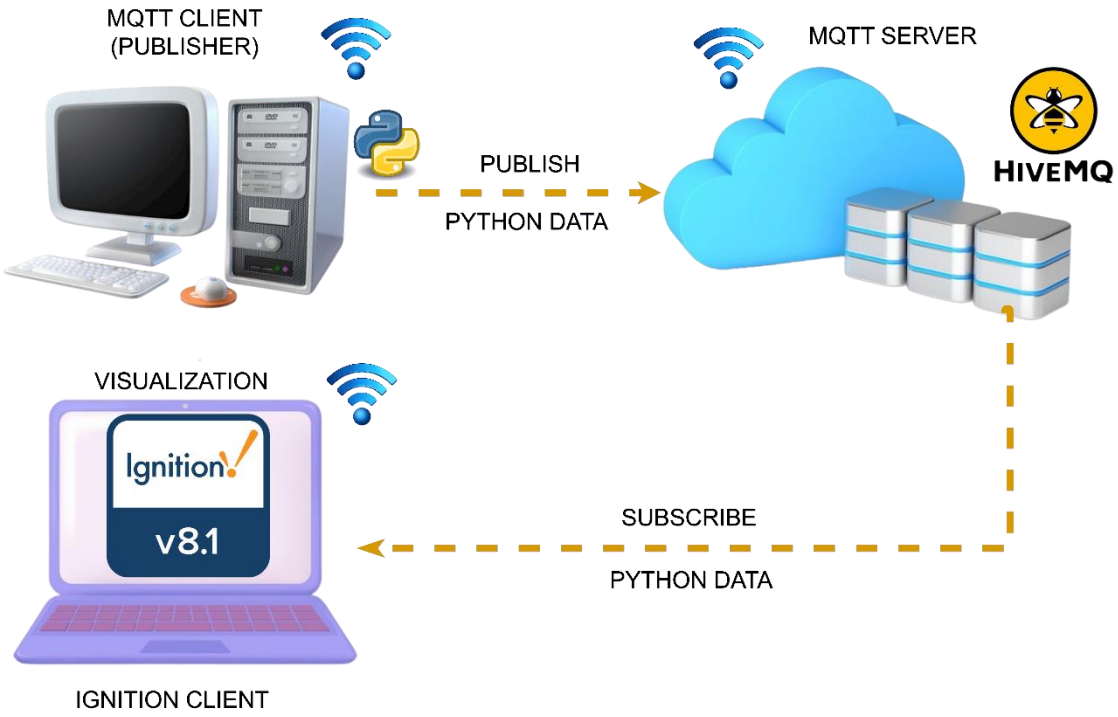


Figura 1. Arquitectura del protocolo MQTT

ACTIVIDAD 4: ALMACENAR DATOS EN BASES DE DATOS

La Figura 2 muestra el esquema del almacenamiento en base de datos del cliente Ignition.

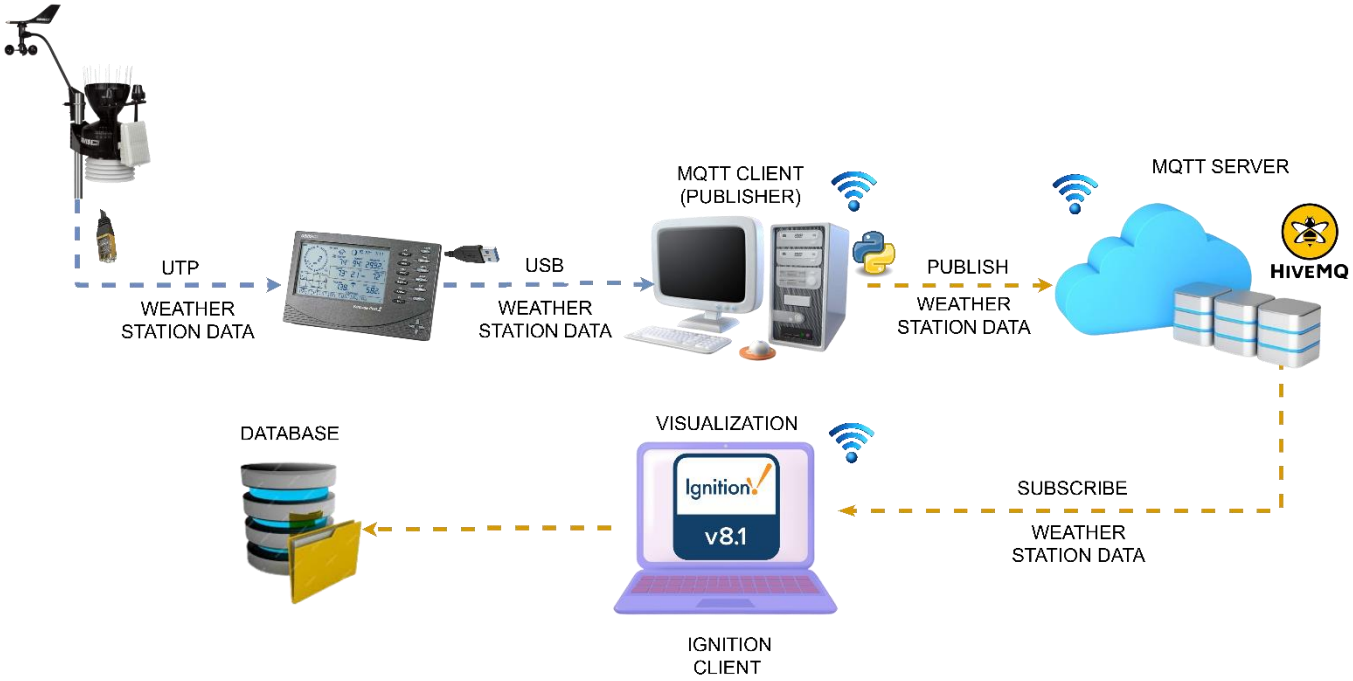


Figura 2. Almacenamiento en base de datos del cliente Ignition

La siguiente fase implica almacenar los datos enviados por Python en una base de datos para luego visualizarlos mediante un componente gráfico, que nos ofrecerá una representación del cambio a lo largo del tiempo. Para llevar a cabo este proceso, es crucial configurar los tags como "Tag History" (una característica clave para el almacenamiento y visualización de datos históricos asociados a un tag). Puedes seguir el siguiente video: Tag History, el cual proporciona una guía detallada sobre cómo realizar esta configuración.

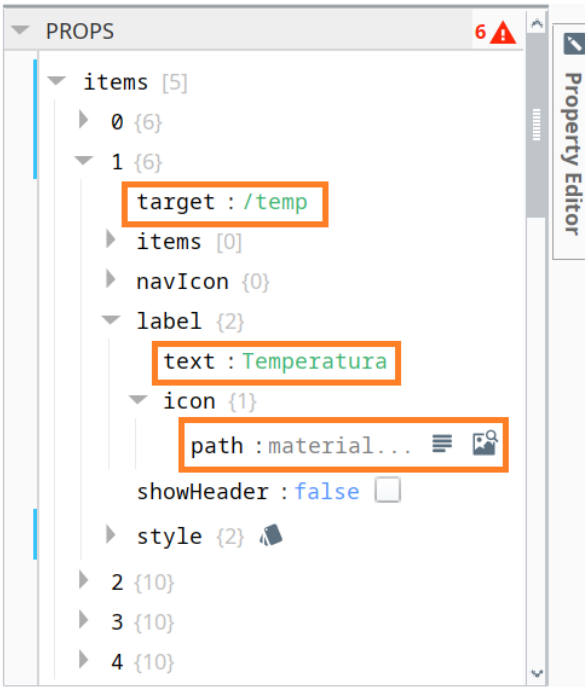
Una vez que los tags están configurados, el siguiente paso es crear nuevas páginas y vistas para acceder a los gráficos de cada una de las variables. Este proceso se detalla en el siguiente video: Crear páginas en Ignition.

Después de haber creado las vistas y páginas, procedemos a la configuración de los gráficos, los cuales se realizan mediante elementos del tipo "Power Chart". Para facilitar esta configuración, se recomienda consultar los siguientes videos:

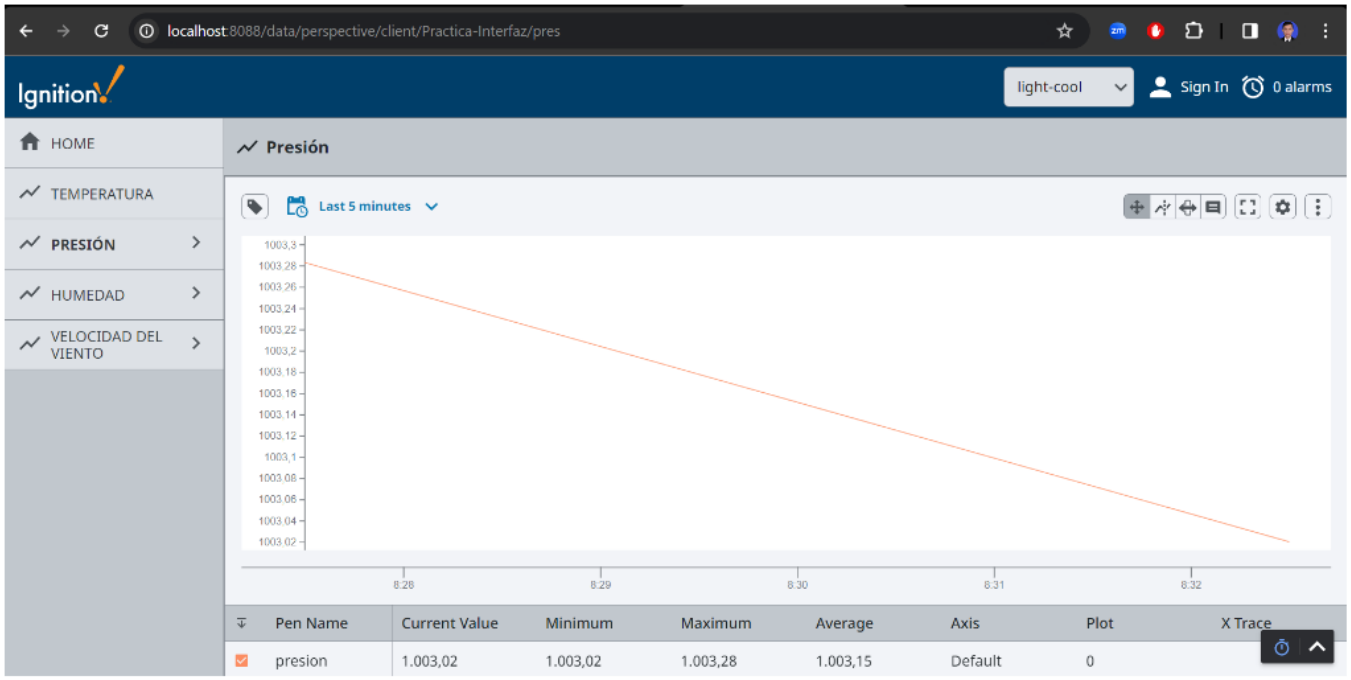
- Power Chart - Descripción general
- Configuración Power Chart

Una vez finalizada la creación de las páginas con las gráficas, el siguiente paso es añadirlas al menú lateral ubicado en la parte izquierda. Para lograr esto, debemos modificar el componente que se encuentra en Perspective/Views/Docks/Menu/root/Features. Accedemos al "Property Editor" para ajustar las opciones del menú lateral. En la sección "PROPS", añadimos tres nuevos elementos correspondientes a las tres variables que deseamos incorporar. En este caso, eliminaremos la opción de alarmas del menú. Las propiedades que requerimos editar son:

- **target**: Página a la que deseamos dirigirnos.
- **text**: Texto que se mostrará en el menú.
- **path**: Ícono que representará la opción en el menú.



El resultado final debería parecerse a lo que se muestra en la imagen; en este caso, estamos viendo la gráfica de la presión.



ACTIVIDAD DE CIERRE (OPCIONAL)

Implementa un proyecto más avanzado que involucre la integración con dispositivos IoT reales.

CONCEPTOS PARA REVISAR

Antes de avanzar a la siguiente guía, revisa los siguientes conceptos:

- Especificación Sparkplug B para MQTT.

REFERENCIAS BIBLIOGRÁFICAS

- Página web de Inductive University. Inductive University
- Manual de usuario Ignition 8.1. Manual Ignition 8.1
- Documentación sobre el módulo MQTT Engine. Módulo MQTT Engine

ENLACE DE GOOGLE COLAB

https://colab.research.google.com/drive/1f2Iu_5QbJcj_s9EG1_IbZKJTCPbQ0IzY?usp=drive_link

Apéndice C. Guía #3

Enlace para acceder a la guía de aprendizaje: https://colab.research.google.com/drive/1MBsSiCV6xTTf7-OXwA4-k2g-S-f6YjLk?usp=drive_link

PRÁCTICA # 06

VISUALIZACIÓN REMOTA DE DATOS USANDO EL PROTOCOLO MQTT SPARKPLUG B

Diseñada por: Felipe Rubio (E3T)
Revisada por: Juan Manuel Rey (E3T)

OBJETIVOS DE APRENDIZAJE

Al finalizar la guía, se espera que el estudiante sea capaz de:

- Adquirir conocimientos sobre el protocolo MQTT Sparkplug B y su aplicación en IoT.
- Aprender a recopilar datos de una estación meteorológica y una cámara, para luego transmitirlos utilizando MQTT Sparkplug B.
- Desarrollar una interfaz completa en Ignition para la visualización de datos de la estación CASIRI.

CONOCIMIENTOS PREVIOS REQUERIDOS

Se espera que el estudiante tenga conocimientos básicos sobre Internet de las Cosas (IoT), protocolo MQTT, programación en Python y manejo de software de desarrollo como Ignition.

INTRODUCCIÓN

En esta guía de laboratorio, exploraremos cómo visualizar de forma remota datos adquiridos de la estación CASIRI utilizando el protocolo MQTT Sparkplug B. El objetivo es aprender a adquirir y visualizar variables ambientales y fotos del cielo a través de una interfaz desarrollada en Ignition.

MATERIALES

Estación CASIRI:

- Estación meteorológica Davis Vantage Pro 2
- Consola para visualización de datos de la estación meteorológica
- Cámara Oculus All Sky
- Controlador de la cámara (Jetson Nano).

Computadora con software de desarrollo:

- Ignition
- Servidor MQTT (HiveMQ Cloud)
- Python IDE (Visual Studio Code)
- SO Linux Ubuntu 20.04.

Conexión a Internet.

ACTIVIDADES PRELIMINARES

- **Preparación del entorno de desarrollo** El primer paso es establecer un entorno de desarrollo para Python utilizando Visual Studio Code, asegurándote de instalar la versión 3.8 o 3.9 de Python.
- **Conectar y verificar la comunicación de la estación CASIRI** Después de instalar Visual Studio Code con la versión específica de Python mencionada, el siguiente paso implica conectar la estación meteorológica DAVIS Vantage Pro 2 y la cámara al controlador Jetson Nano para capturar imágenes en el PC que actuará como el publicador de datos.Es importante asegurarse de que la comunicación entre todos estos dispositivos esté correctamente establecida y funcione sin problemas.

ACTIVIDAD 1: ADQUISICIÓN DE VARIABLES DE LA ESTACIÓN CASIRI

1. Conecta la consola de la estación Davis Vantage Pro 2 al PC que actuará como cliente publicador. Además, asegúrate de conectar la cámara Oculus All Sky al controlador Jetson Nano y luego conectar este controlador al PC que será usado como publicador.
2. Configura el módulo MQTT Engine en el gateway Ignition para establecer la comunicación con el servidor.

ACTIVIDAD 2: IMPLEMENTACIÓN DE SPARKPLUG B

1. Asegurarse de tener instalado Python (versión 3.8 o 3.9).
2. Acceder a la carpeta CASIRI a través del siguiente enlace Implementación de Sparkplug B CASIRI y descargarla. Esta carpeta contiene todos los archivos necesarios para el correcto funcionamiento de la estación meteorológica y la cámara, así como la implementación del protocolo MQTT Sparkplug B.
3. Asegurarse de tener instalado pip (Python).

4. Asegurarse de tener instalada la versión 1.6.1 de la biblioteca paho-mqtt.
5. Instalar la versión 3.20 de la biblioteca protobuf.
   La biblioteca protobuf en Sparkplug B se utiliza para definir la estructura de los mensajes que se intercambian entre los dispositivos y los sistemas de control.
6. Instalar la biblioteca PyVantagePro para adquirir los datos de la estación meteorológica Davis Vantage Pro 2.
7. Cambiar ubicación de la carpeta core del archivo sparkplug_cloud.py.
   - Para cambiar la ubicación de la carpeta core del archivo sparkplug_cloud.py, sigue estos pasos:
   - Ubica el archivo sparkplug_cloud.py en la siguiente ruta: CASIRI/ESTACION/Sparkplug/sparkplug_core/python/core.
   - Abre el archivo sparkplug_cloud.py y busca la línea 16, donde encontrarás la instrucción: `sys.path.insert(0, "/home/casiri/Desktop/CASIRI/ESTACION/Sparkplug/sparkplug_core/python/core")`
   - Modifica la ruta en la línea 16 por la ubicación de la carpeta core en tu PC, donde ejecutarás el programa. Esto es importante para evitar errores al encontrar los archivos de la biblioteca Sparkplug B.

     Al seguir estos pasos, asegúrate de tener la ubicación correcta de la carpeta core en tu PC y de actualizar la ruta en el archivo sparkplug_cloud.py para que el programa funcione sin problemas.

ACTIVIDAD 3: EJECUCIÓN DEL SISTEMA COMPLETO

Seguir las instrucciones del archivo Readme.txt que se encuentra en la carpeta CASIRI descargada previamente. La Figura 1 muestra el esquema de la toma de datos de la estación meteorológica y la publicación usando el protocolo MQTT Sparkplug B.

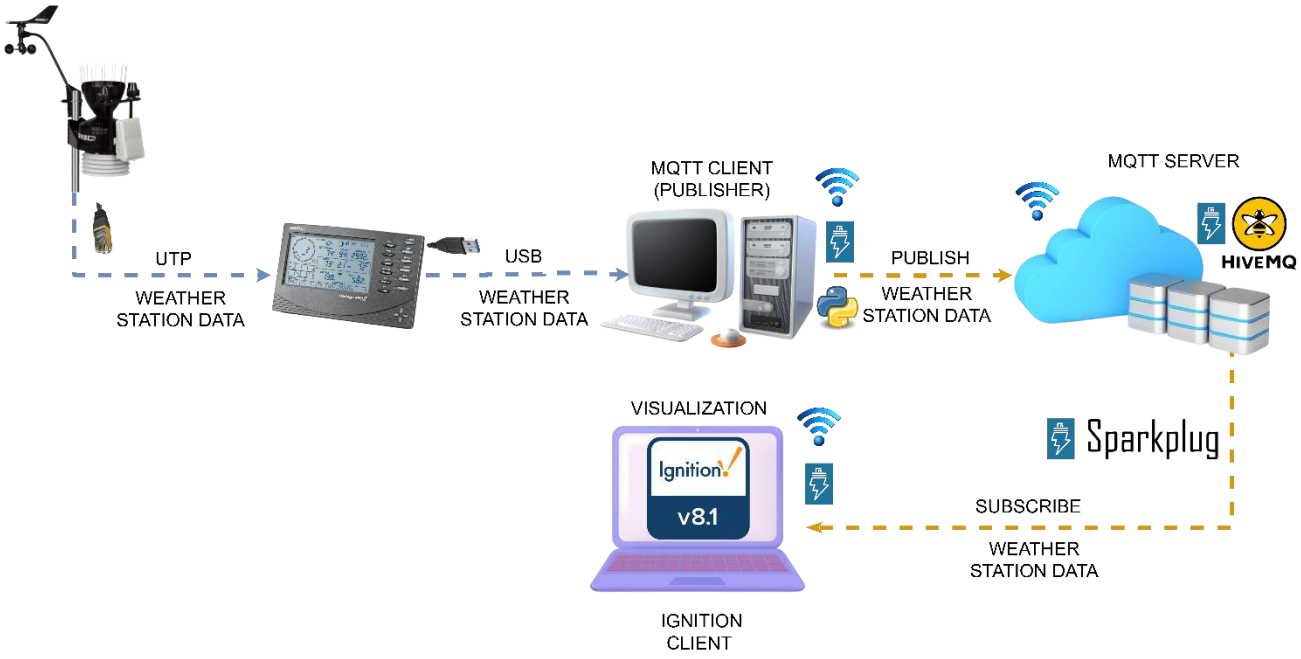


Figura 1. Toma de datos de la estación meteorológica y la publicación usando el protocolo MQTT Sparkplug B

REFERENCIAS BIBLIOGRÁFICAS

- Página web Davis Instruments - Vantage Pro 2. Davis Vantage Pro 2
- Manual de la consola Davis Vantage Pro 2. Manual
- Documentación de la biblioteca Protobuf. Protobuf
- Documentación de la biblioteca PyVantagePro. PyVantagePro

ENLACE DE GOOGLE COLAB

https://colab.research.google.com/drive/1MBsSiCV6xTTf7-OXwA4-k2g-S-f6YjLk?usp=drive_link