Solución IoT basada en Node-RED para la visualización, almacenamiento y procesamiento en la nube para la estación meteorológica CASIRI

Brayan Hernando González Mendoza

Carlos Eduardo Silva Sepúlveda

Trabajo de Grado para Optar el Título de Ingeniero Electrónico

Director

Homero Ortega Boada

PhD. of Engineering Sciences

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingeniería Eléctrica, Electrónica y de Telecomunicaciones

Ingeniería Electrónica

Bucaramanga

2024

Dedicatorias

Este proyecto de grado se lo dedico a todas las personas que me apoyaron en mi formación profesional, especialmente mi familia conformada por mis padres, mi novia y mi hermana, la cual ultima le agradezco que haya creído en mi pagando de nuevo el examen ICFES.

Carlos Eduardo Silva Sepúlveda

Este proyecto de grado se lo dedico a mi madre y padre por brindarme su apoyo, compresión y sobre todo su amor que me brindan todos los días y me permite ser la persona que soy hoy en día. A mi hermano por sus esfuerzos, enseñanzas y el apoyo que siempre me ha brindado.

Brayan Hernando González Mendoza

Agradecimientos

Agradezco a todas las personas que me apoyaron en mi formación como profesional y de alguna manera contribuyeron en ella, como lo fueron profesores, compañeros y familia.

Carlos Eduardo Silva Sepúlveda

Quiero agradecer primeramente a Dios por permitirme culminar satisfactoriamente mi carrera profesional. A nuestro director Homero Ortega Boada, quien siempre estuvo disponible y nos orientó a lo largo del proyecto y cada uno de los profesores de la Escuela que hicieron parte de mi formación y colocaron una semilla para la realización de este proyecto. a mis padres y hermano por el apoyo incondicional.

Brayan Hernando González Mendoza

Tabla de Contenido

Lista Apéndices

Lista de Figuras

Resumen

Título: Solución IoT basada en Node-RED para la visualización, almacenamiento y procesamiento en la nube para la estación meteorológica CASIRI*
Autores: Brayan Hernando González Mendoza, Carlos Eduardo Silva Sepúlveda**
Palabras clave: Node-RED, Nube, IoT, API.

Descripción: La alianza de los grupos de investigación CEMOS, CPS, y RadioGIS ha desarrollado una solución IIoT llamada CASIRI, diseñada para la gestión de mediciones ambientales y el monitoreo de señales de radio. Hasta el momento se han tenido varios enfoques, desde el uso del protocolo MQTT, bases de datos MySQL, visualización basada en HTML, hasta el uso de técnicas más propias del IIoT apoyadas en el software de Ignition. El presente proyecto adopta un enfoque distinto en la búsqueda de la eficiencia para las necesidades concretas del grupo que trabaja en radioastronomía, para ello se apoya al máximo en lograr aprovechar la mejor combinación entre las oportunidades que brinda el uso de un servidor en una plataforma como AWS y lo mejor de las tecnologías serverless, como es el caso de Google Workspace. Como resultado se implementa una solución que usa recursos de Google como base de datos Workspace y cómputo en la nube para el procesamiento de esos datos, pero usa Node-RED a nivel de un servidor de muy bajo costo sobre AWS. La implementación contiene aspectos originales y futuristas, al menos para usuarios de radioastronomía y similares ya que permite satisfacer la necesidad de medir siempre en presencia o en ausencia de conectividad, es decir en lugares remotos donde no hay internet. Logra también la mejor relación de costos ya que los procesos de medición no se detienen aun cuando el servidor en AWS está apagado, los datos siguen siendo registrados en los recursos basados en Google WorkSpace.. Dicho en otras palabras, la solución basada en Node-RED que es software de uso libre sobre AWS se limita a la visualización en línea de datos, sin que el usuario requiera instalar software especializado en su computador o terminal móvil. Esa es al menos la visión identificada gracias a los resultados de este proyecto.

---

*Proyecto de grado

** Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería Eléctrica, Electrónica y de Telecomunicaciones. Director: Homero Ortega Boada - Ph.D of Engineering Sciences.

Abstract

Title: Node-RED based IoT solution for visualization, storage and cloud processing for the CASIRI weather station*

Authors: Brayan Hernando González Mendoza, Carlos Eduardo Silva Sepúlveda**

Key Words: Node-RED, Nube, IoT, API.

Description: The alliance of research groups CEMOS, CPS, and RadioGIS has developed a IIoT solution called CASIRI, designed for environmental measurement management and radio signal monitoring. So far there have been several approaches, from the use of the MQTT protocol, MySQL databases, HTML-based visualization, to the use of more IIoT techniques supported in the Ignition software. This project takes a different approach in the search for efficiency for the specific needs of the group working in radio astronomy, To do this, it relies to the maximum in order to take advantage of the best combination between the opportunities offered by the use of a server in a platform such as AWS and the best of serverless technologies, such as Google Workspace. As a result, a solution is implemented that uses Google resources such as workspace database and cloud computing for the processing of that data, but uses Node-RED at the level of a very low-cost server over AWS. The implementation contains original and futuristic aspects, at least for radio astronomy users and the like since it allows to satisfy the need to always measure in presence or in absence of connectivity, that is in remote places where there is no internet. It also achieves the best cost ratio since the measurement processes do not stop even when the server in AWS is off, the data is still recorded in the resources based on Google WorkSpace. In other words, the Node-RED-based solution that is free-to-use software on AWS is limited to online data viewing, without the user requiring to install specialized software on their computer or mobile terminal. That is at least the vision identified by the results of this project.

---

*Proyecto de grado

** Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería Eléctrica, Electrónica y de Telecomunicaciones. Director: Homero Ortega Boada - Ph.D of Engineering Sciences.

Introducción

La inexplorada extensión de la Antártida ha dejado su huella en la Universidad Industrial de Santander (UIS), participando activamente en la Exploración Antártica de Colombia durante dos años consecutivos, ahora en su décima edición. En este viaje, emerge un proyecto que no solo conecta la tierra y el cielo, sino que también sitúa a la UIS en la vanguardia de la investigación tecnológica. A pesar de la fascinación por estudiar el universo distante, se reconoce la importancia intrínseca de comprender nuestro entorno terrestre. Casi todo avance cósmico tiene sus raíces en proyectos de radioastronomía, como la capacidad de captar los susurros del universo desde la tierra. Justamente, realizar esas tareas también demanda un profundo conocimiento de las condiciones ambientales que pueden influir en la recepción apropiada de señales de radio que provienen del espacio profundo.

Hoy numerosas tecnologías facilitan el desarrollo de soluciones IoT a nivel de la nube, una de ella es Node RED, desarrollada por IBM y utilizada en una amplia variedad de aplicaciones en el ámbito del Internet de las cosas (IoT), desde la automatización industrial, la integración de sistemas y desarrollo rápido de prototipos. El mayor potencial de Node RED está en las soluciones a nivel de nube. Pero también tiene una particularidad que ha impulsado fuertemente su desarrollo. Se trata del uso de un entorno de programación visual basado en nodos que permite la creación de aplicaciones y flujos de trabajo de manera intuitiva. En términos reales, Node RED equivale al uso de Node.js pero potenciada además con programación gráfica. En este sentido, vale la pena destacar que Node.js es un entorno de tiempo de ejecución de código abierto para facilitar la ejecución de código JavaScript en el lado del servidor, algo que que tradicionalmente se había limitado al entorno del navegador. En otras palabras, Node.js actúa como un impulsor del cómputo en la nube, destacando su capacidad para aprovechar eficientemente las soluciones preexistentes

en la nube, ya sea en forma de servicios web, aplicaciones web o APIs. De esta manera, no sería exagerado afirmar que Node-RED representa una revolución en el desarrollo de soluciones en la nube, donde la única limitación radica en la creatividad y visión de implementación.

Un área que ha experimentado un notable crecimiento gracias al IoT es el monitoreo y la predicción meteorológica. La recopilación precisa de datos meteorológicos es esencial para comprender y anticipar fenómenos atmosféricos, permitiendo una planificación más efectiva y decisiones informadas en diversos sectores como la agricultura, construcción, aviación y gestión de desastres naturales. En la UIS, se ha forjado una alianza entre grupos de investigación como RadioGIS, CEMOS, CPS, Senivan y Dautom para impulsar desarrollos de IIoT. Un ejemplo de estos es CASIRI (Caracterización de SItios en Radio Interferencias), un sistema diseñado para caracterizar sitios con condiciones ambientales apropiadas para la instalación de radio observatorios en bajas frecuencias, apoyando así la radioastronomía. CASIRI incluye una estación meteorológica con sensores ambientales, pero va más allá porque incluye equipos de radio para evaluar que la presencia de interferencias terrenales o espaciales no superen límites tolerables para recibir señales del espacio lejano. La estación CASIRI ha sido diseñada específicamente para instalarse en lugares con condiciones climáticas adversas, donde la recopilación precisa de datos meteorológicos es esencial, como en regiones montañosas o zonas árticas, presentando desafíos únicos debido a su acceso limitado y condiciones extremas.

El problema global abordado por este proyecto de grado se centra en mejorar el rendimiento de la estación meteorológica CASIRI. Las causas de este problema son diversas, desde la necesidad de conectar sensores de ambiente y equipos de radio que detecten señales de banda ancha en tiempo real, hasta el almacenamiento, procesamiento, control remoto de la información

y su visualización en linea. Este proyecto contribuye específicamente a la necesidad de soluciones en la nube, utilizando herramientas de desarrollo apropiadas para CASIRI.

Revisadas las historias de usuario, así como las ventajas que tiene Node-RED se ha identificado la necesidad de una nueva versión de CASIRI que explota lo mejor de tecnologías de la nube, como soluciones serverless y un servidor de Node-RED. Este no requiere licencia especial y brinda amplias capacidades en la nube. En la etapa de planeación del proyecto se identificaron los siguientes objetivos:

Mejorar el sistema CASIRI mediante la implementación de Node RED para el diseño de soluciones que incluyan visualización y procesamiento en la nube.

Estudiar el sistema CASIRI y en qué medida responde a las necesidades de los usuarios, como punto de partida.

Estudiar las características y funcionalidades específicas de Node RED para responder a las necesidades identificadas. Esto implica analizar su capacidad para gestionar flujos de datos, interactuar con dispositivos y servicios externos, y ofrecer soluciones escalables y eficientes.

Diseñar e implementar una solución de visualización en nube basada en Node RED.

Ampliar las capacidades de la solución para considerar el guardado de la información en nube.

Para abordar estas necesidades se ha desarrollado una solución IoT que consta inicialmente con la implementación de un driver basado en Python para recopilar y organizar datos de manera local. El script de Python procesa las muestras de la estación, las organiza en un formato CSV y

las carga a la nube mediante la API de Google Sheets, para su posterior procesamiento con las herramientas de Google Workspace y Node-RED. La información en la nube se procesa utilizando los atributos que brinda Node-RED, en donde a dicha información se le hacen distintos tratamientos para el desarrollo de distintos servicios. Toda esta información se visualiza en una HMI (human-machine interface) diseñada para satisfacer las necesidades de los interesados que van desde la consulta en línea del historial de datos hasta la integración de Google Calendar para programar eventos en la estación. Para lograr la máxima accesibilidad posible de la interfaz gráfica, se anticipa el uso de una IP pública proporcionada por una máquina virtual de AWS que permite a los usuarios interactuar de forma remota con la HMI a través del dashboard implementado sobre Node-RED.

Este desarrollo se alinea con el objetivo general del proyecto de mejorar el sistema CASIRI mediante la implementación de Node-RED para diseñar soluciones que integren visualización y procesamiento en la nube, teniendo en cuenta también los objetivos específicos que incluyen el estudio del sistema CASIRI, la exploración de las características de Node-RED, y la implementación de soluciones de visualización en la nube, ampliando las capacidades para contemplar el almacenamiento de información en dicho entorno.

Gracias a los resultados de este proyecto se permitirá explorar el potencial de Node RED en soluciones IoT e IIoT no sólo de monitoreo remoto de estaciones meteorológicas. Se espera que esto fortalezca la investigación y el desarrollo tecnológico en el campo de IoT e IIoT, consolidando a la UIS como líder en este ámbito. Socialmente, la disponibilidad de datos meteorológicos en tiempo real beneficiará a sectores como la agricultura, construcción, aviación y gestión de desastres naturales, facilitando la toma de decisiones informadas y promoviendo el desarrollo sostenible.

1. Marco Referencial

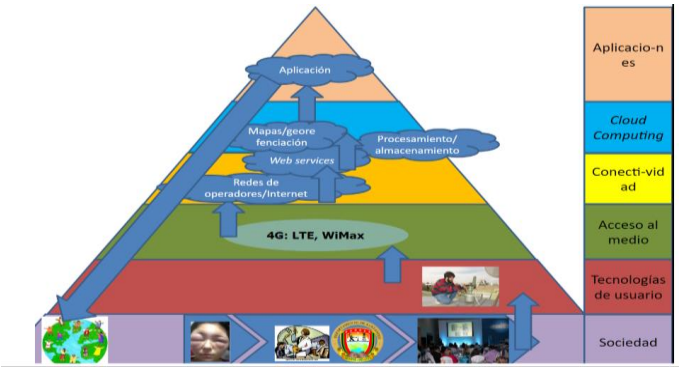
1.1 Ciudades inteligentes

Una ciudad inteligente y sostenible utiliza las tecnologías de la información y la comunicación (ICT, por sus siglas en inglés) para mejorar la calidad de vida, la eficiencia y la competitividad, garantizando al mismo tiempo que responde a las necesidades de las generaciones presentes y futuras. Según la Comisión Económica para Europa de las Naciones Unidas (CEPE), la definición de smart city incluye elementos como la conectividad a Internet doméstica generalizada y el Wi-Fi en zonas públicas, así como infraestructuras inteligentes, Smart meter eléctricos, datos abiertos y una administración electrónica. (*Smart City: significado, características y ejemplos | Enel X*, s. f.).

Un modelo para comprender bien este concepto de ciudades inteligentes es el ilustrado en la figura 1, en donde se plantea un esquema convergente por capas.

Figura 1.

Modelo convergente de smart cities.



Nota. Modelo convergente de smart cities en el que se basó el modelo del proyecto.

En este modelo, en la primera capa se ubica la sociedad con todas las necesidades que tiene bien sea en medicina, educación, medioambiente, agro, etc. En la siguiente capa están todas las tecnologías que hacen contacto directo con la sociedad como: instrumentos de hospitales, sensores del medio ambiente, etc. En la tercera capa están los medios brindan acceso para que esas tecnologías, por muy sencillas que sean tengan capacidades de red bien sea alámbrica o inalámbrica. En la cuarta capa está la internet desde el punto de vista de la red de redes que permite interconectar todo. En la quinta capa están recursos de mayores capacidades como los que ofrecen grandes operadores como Google, AWS, entre otros, pero también los servidores que nosotros mismos creemos para ofrecer servicios. Allí se resuelve el cómputo y el almacenamiento en la nube. En la última capa están las aplicaciones que atienden directamente las necesidades de la sociedad.

1.2 Node

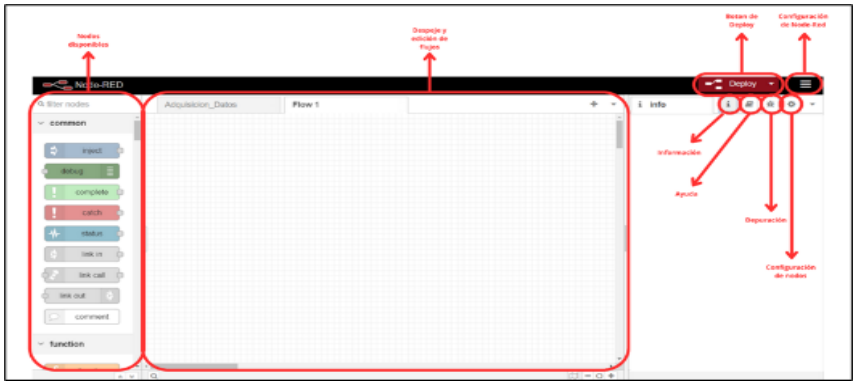
Node.js se presenta como una herramienta esencial para las ciudades inteligentes gracias a su eficiencia y escalabilidad en el desarrollo de aplicaciones de servidor y red. Su capacidad para manejar grandes volúmenes de datos en tiempo real, junto con un rendimiento rápido basado en el motor V8 de Google Chrome, lo hace idóneo para casos de uso como el control de tráfico, monitoreo de sensores y gestión de servicios públicos. Su versatilidad también destaca en la creación de APIs REST, implementación de chats en tiempo real y desarrollo de interfaces interactivas, lo que lo convierte en una herramienta adaptable y poderosa para abordar los desafíos tecnológicos de las ciudades inteligentes. (Infante & Infante, 2023)

1.3 Node-RED

Node-RED representa una revolución en comparación con otras tecnologías serverless, ya que va más allá de ser simplemente una herramienta de programación visual. Esta herramienta se destaca por su impacto innovador y su amplio alcance, marcando una diferencia significativa en el panorama tecnológico. Funcionando como un editor de flujo basado en el navegador, Node-RED ofrece una representación visual de relaciones y funciones, permitiendo a los usuarios programar de manera intuitiva sin la necesidad de escribir código. En un contexto donde otras tecnologías serverless no logran igualar sus capacidades, Node-RED se posiciona como una solución única. Cada nodo en esta plataforma, ya sea de inyección o de función, desempeña un papel crucial en la creación de mensajes y la ejecución de tareas específicas, consolidando su estatus como una herramienta revolucionaria que supera los límites convencionales de las plataformas de programación visual en la nube. Con una gran cantidad de estos nodos para elegir, Node-Red hace que el conectar los dispositivos de hardware, APIs y servicios en línea sea más fácil que nunca. (*Jesús Darío - Toptal*).

Figura 2.

Software Node-RED.



Nota. Gateway de Node-RED donde se implemento el flujo de la solución IoT.

1.4 API

Las API son mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos. Por ejemplo, el sistema de software del instituto de meteorología contiene datos meteorológicos diarios. La aplicación meteorológica de su teléfono "habla" con este sistema a través de las API y le muestra las actualizaciones meteorológicas diarias en su teléfono. API significa "interfaz de programación de aplicaciones". En el contexto de las API, la palabra aplicación se refiere a cualquier software con una función distinta. La interfaz puede considerarse como un contrato de servicio entre dos aplicaciones. Este contrato define cómo se comunican entre sí mediante solicitudes y respuestas. La documentación de su API contiene información sobre cómo los desarrolladores deben estructurar esas solicitudes y respuestas. (*AWS Amazon*)

1.5 AWS

Amazon Web Services (AWS) es la nube más adoptada y completa en el mundo, que ofrece más de 200 servicios integrales de centros de datos a nivel global. AWS está diseñado para ser el entorno de informática en la nube más flexible y seguro disponible en la actualidad. Nuestra infraestructura principal se creó para cumplir con los requisitos de seguridad del ejército, los bancos internacionales y otras organizaciones que deben cumplir requisitos de confidencialidad estrictos. (*What-is-AWs,* s. f.)

1.6 Máquina Virtual

Una máquina virtual de sistema es aquella que emula a un ordenador completo. En palabras llanas, es un software que puede hacerse pasar por otro dispositivo -como un PC- de tal modo que

puedes ejecutar otro sistema operativo en su interior. Tiene su propio disco duro, memoria, tarjeta gráfica y demás componentes de hardware, aunque todos ellos son virtuales. (Ramírez, 2020)

1.7 Dirección IP

Una dirección IP es una dirección única que identifica a un dispositivo en Internet o en una red local. IP significa “protocolo de Internet”, que es el conjunto de reglas que rigen el formato de los datos enviados a través de Internet o la red local. En esencia, las direcciones IP son el identificador que permite el envío de información entre dispositivos en una red. Contienen información de la ubicación y brindan a los dispositivos acceso de comunicación. Internet necesita una forma de diferenciar entre distintas computadoras, routers y sitios web. Las direcciones IP proporcionan una forma de hacerlo y forman una parte esencial de cómo funciona Internet. (*Qué es una dirección IP: definición y explicación*, 2023)

1.8 Estación meteorológica CASIRI

La estación de radioastronomía está ubicada en el laboratorio de investigación RadioGis de la Universidad Industrial de Santander. Consta de 3 subsistemas, estaciones Equipos meteorológicos que captan datos atmosféricos, cámaras que captan imágenes a cielo abierto y USRP (Universal Software Radio Peripheral) se utiliza para recibir señales de radio en un ancho de banda específico.

1.9 Web Service

Un servicio web (o web service) es una vía de intercomunicación e interoperabilidad entre máquinas conectadas en Red. En el mundo de Internet se han popularizado enormemente, ya se

trate de web services públicos o privados. Generalmente, la interacción se basa en el envío de solicitudes y respuestas entre un cliente y un servidor, que incluyen datos. (De Zúñiga, 2023)

1.10 IoT

El término IoT, o Internet de las cosas, se refiere a la red colectiva de dispositivos conectados y a la tecnología que facilita la comunicación entre los dispositivos y la nube, así como entre los propios dispositivos. El Internet de las cosas integra las "cosas" de uso diario con Internet. Los ingenieros en informática llevan agregando sensores y procesadores a los objetos cotidianos desde los años 90. Un sistema común de IoT funciona mediante la recopilación y el intercambio de datos en tiempo real. Un sistema de IoT tiene tres componentes: Dispositivos inteligentes, aplicación de IoT y una interfaz de usuario gráfica. (*¿Qué es IoT? - Explicación del Internet de las cosas - AWS*, s. f.)

1.11 HMI

HMI, o interfaces humano-máquina, es un componente que posibilita la comunicación entre un usuario y una máquina, software o sistema. Se trata de un panel interactivo diseñado para entornos industriales o de control, donde facilita la visualización de datos en tiempo real y permite al usuario ejercer control sobre las máquinas mediante una interfaz gráfica de usuario. En esencia, las HMI sirven como intermediarios que traducen las acciones humanas en comandos comprensibles para la maquinaria, al mismo tiempo que proporcionan retroalimentación visual instantánea sobre el estado y el rendimiento del sistema. (*¿Qué significa HMI? Interfaz humano-máquina | COPA-DATA*, 2023)

1.12 Python

Python es un lenguaje de programación interpretado, orientado a objetos, de alto nivel y con semántica dinámica. Sus estructuras de datos integradas de alto nivel, combinadas con escritura dinámica y enlace dinámico, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de secuencias de comandos o pegamento para conectar componentes existentes. La sintaxis simple y fácil de aprender de Python enfatiza la legibilidad y, por lo tanto, reduce el costo de mantenimiento del programa. Python admite módulos y paquetes, lo que fomenta el modularidad del programa y la reutilización del código. Python no solo es reconocido por su elegancia y facilidad de aprendizaje, sino que también se destaca como un competidor significativo en el desarrollo web, con frameworks como Flask y Django, Python ofrece soluciones robustas y flexibles para la creación de aplicaciones web modernas. En el front-end, herramientas como Django REST framework facilitan la construcción de interfaces de usuario interactivas y atractivas. Al mismo tiempo, en el back-end, Python se ha consolidado como una opción líder, respaldado por su eficiencia y amplia comunidad de desarrolladores. Esta versatilidad en el ámbito del desarrollo web consolida a Python como un lenguaje integral en el panorama tecnológico actual.

1.12.1 Librería pandas

Pandas es una biblioteca de análisis de datos en Python que proporciona estructuras de datos flexibles y herramientas para el análisis y manipulación de datos. La estructura principal de datos en pandas es el DataFrame, que es una tabla bidimensional con etiquetas en filas y columnas. Pandas facilita la carga, limpieza, transformación y análisis de datos de manera eficiente.

1.12.2 Librería openpyxl

Openpyxl es una biblioteca de Python que permite trabajar con archivos Excel en el formato de archivo xlsx. Proporciona funcionalidades para crear, leer, modificar y guardar archivos Excel. Con openpyxl, puedes acceder a las celdas, aplicar estilos, trabajar con fórmulas y realizar diversas operaciones en hojas de cálculo Excel.

1.12.3 Librería os

El módulo os es parte de la biblioteca estándar de Python y proporciona funciones para interactuar con el sistema operativo. Permite realizar operaciones relacionadas con la gestión de archivos y directorios, como verificar la existencia de archivos, crear o eliminar directorios, obtener información sobre rutas de archivos, entre otras. En el código proporcionado, se utiliza para verificar la existencia de un archivo antes de realizar operaciones en él.

1.13 JavaScript

JavaScript® (a menudo abreviado como JS) es un lenguaje ligero, interpretado y orientado a objetos con funciones de primera clase, y mejor conocido como el lenguaje de programación para las páginas Web, pero también se utiliza en muchos entornos que no son de navegador. Es un lenguaje de scripts que es dinámico, multi paradigma, basado en prototipos y admite estilos de programación orientados a objetos, imperativos y funcionales.

JavaScript se ejecuta en el lado del cliente de la web, y se puede utilizar para estilizar/programar cómo se comportan las páginas web cuando ocurre un evento. JavaScript es un potente lenguaje de scripts y fácil de aprender, ampliamente utilizado para controlar el comportamiento de las páginas web. (*Acerca de JavaScript | MDN*, s. f.)

1.14 Front-END

El frontend o «desarrollo del lado del cliente» se refiere a la práctica de producir HTML, CSS y JavaScript. Estos tres elementos se encargan de dar forma a la parte frontal de un sitio web o aplicación. Esto incluye los fondos, colores, texto, animaciones o efectos. El frontend sirve para realizar la interfaz de un sitio web, desde su estructura hasta los estilos, como pueden ser la definición de los colores, texturas, tipografías, secciones, entre otros. Su uso es determinante para que el usuario tenga una buena experiencia dentro del sitio o aplicación. (Coppola, 2023)

1.15 Back-END

El backend es el encargado de procesar toda la información que alimenta a un frontend. Se compone de marcos, bases de datos o códigos. Para que un sitio web o aplicación opere efectivamente, se requiere mucha información y datos que se almacenan en «la parte trasera» de un sistema informático. En oposición al frontend, el usuario no puede ver o acceder a esta información. El backend son todos los códigos ocultos que sirven para que una página web o aplicación funcione correctamente. Además, de su estructura y organización depende la experiencia de usuario. De igual forma, el backend se encarga de optimizar otros elementos y recursos como la seguridad y privacidad en un sitio web o aplicación.

1.16 Serverless

Serverless, o sin servidor, describe una solución tecnológica que posibilita el desarrollo y la ejecución eficiente de aplicaciones con una rápida puesta en marcha y un menor costo total de propiedad. Aunque en realidad existen servidores en segundo plano, la característica distintiva radica en que el proveedor de servicios en la nube asume la responsabilidad total de la

administración de la infraestructura subyacente. Esto implica que los desarrolladores ya no necesitan preocuparse por tareas como el aprovisionamiento, la gestión de servidores, sistemas operativos o software. En lugar de ello, pueden enfocarse exclusivamente en la escritura y optimización del código de la aplicación. (Flores, 2023)

1.17 Google Apps Script

Google Apps Script es una plataforma que posibilita el desarrollo de scripts con diversas funcionalidades, ejecutándose en un entorno definido por Google. Estas mini-aplicaciones se construyen utilizando el lenguaje de programación Javascript, con algunas características especiales adaptadas a este entorno. La versatilidad de Google Apps Script permite la conexión con fuentes de datos tanto dentro del propio ecosistema de Google como externas. En esencia, los scripts y aplicaciones desarrollados en Google Apps Script están principalmente diseñados para automatizar tareas internas y procesar información de manera eficiente. Aunque la plataforma no excluye otros posibles usos, sus limitaciones la convierten en una herramienta especialmente adecuada para crear utilidades que optimizan procesos y mejoran la productividad en el contexto de las aplicaciones y servicios ofrecidos por Google. (Beservices, 2023)

1.18 Google Sheet API.

La Google Sheets API es una interfaz de programación de aplicaciones proporcionada por Google, que permite a los desarrolladores interactuar y manipular hojas de cálculo en la nube de Google Sheets. Al aprovechar esta API, los usuarios pueden automatizar procesos, colaborar en tiempo real y lograr eficiencia en el trabajo en equipo. La API posibilita un enfoque innovador hacia la automatización de tareas mediante la utilización de macros, aprovechando las capacidades en la nube para facilitar la gestión y optimizar los sistemas empresariales. En esencia, la Google

Sheets API ofrece una potente herramienta para integrar y mejorar la funcionalidad de las hojas de cálculo en la nube, permitiendo un flujo de trabajo más eficiente y colaborativo para todas las necesidades empresariales.

2. Desarrollo de la solución

2.1 Historias de usuario

En el marco de nuestro proyecto basado en métodos ágiles, donde la flexibilidad y adaptabilidad son clave, recurrimos a una herramienta fundamental: las historias de usuario. Estas historias no son simples relatos, sino estrategias específicas que nos permiten comprender y abordar las necesidades de los usuarios de manera precisa y centrada en sus requerimientos.

Las historias de usuario son piezas narrativas que describen una funcionalidad deseada desde la perspectiva del usuario. En nuestro contexto, nos sirven como guías para desarrollar soluciones que se alinean directamente con las expectativas y necesidades de quienes interactúan con nuestro sistema.

En este proyecto, hemos identificado las siguientes historias de usuario, cada una representando un aspecto clave de las necesidades identificadas:

Como ingeniero electrónico y miembro del grupo de investigación RadioGis, busca que la estación meteorológica Davis envíe mediciones precisas de temperatura, humedad, presión atmosférica y otros parámetros relevantes a Google Sheet. Esta acción se realiza utilizando funciones serverless de Google Workspace, con el objetivo de asegurar un almacenamiento eficiente y accesible.

En su rol como usuario, busca acceder a las mediciones de la estación meteorológica, tanto localmente como de forma remota. Utiliza la plataforma Node-RED para visualizar en tiempo real y Google Sheet para acceder a datos históricos, asegurando flexibilidad y eficiencia en el proceso.

En su función como ingeniero de IoT y director del grupo de investigación RadioGis, tiene la aspiración de establecer una interconexión fluida entre la estación meteorológica Davis, Google Sheet y Google Calendar. Para lograr esto, se valdrá de servicios serverless de Google Workspace, con el propósito de facilitar la integración y optimizar los costos de mantenimiento.

Como ingeniero electrónico y estudiante de maestría, mi objetivo es emplear Node-RED para evaluar el sistema que proporciona servicios de visualización en tiempo real, actualmente desarrollado en Ignition. La intención es identificar y comparar las ventajas que Node-RED ofrece en relación con la solución existente, especialmente en el contexto de mediciones ambientales.

Como usuario avanzado, desea tener la capacidad de consultar el historial de mediciones almacenado en Google Sheet. Utiliza Node-RED para visualizar datos específicos en intervalos de tiempo definidos, proporcionando información detallada sobre las condiciones ambientales pasadas.

Finalmente, en su papel como director del grupo de investigación RadioGis, aspira a utilizar Google Calendar para programar eventos relacionados con la estación meteorológica, como calibraciones o mantenimientos planificados. Busca integrar esta funcionalidad de manera eficiente con las demás plataformas involucradas en el proyecto.

Teniendo en cuenta lo anterior, las necesidades de los usuarios e interesados identificadas fueron las siguientes:

- Toma de datos y almacenamiento en la nube.
- Acceso a los datos de forma local y remota.
- Implementación de un sistema IoT (Interconexión entre varias tecnologías).
- Desarrollo de servicios de visualización de mediciones registradas en tiempo real.
- Servicios de consulta de historial de mediciones registradas.
- Servicio de programación de eventos apoyada en Google Calendar.

2.2 Propuesta para una solución con visión futurista.

Para abordar las diversas necesidades de los usuarios en el contexto de la solución IoT para radioastronomía, se ha propuesto un diseño general detallado en la figura 3. Este enfoque se centra en resolver los problemas considerando los recursos más accesibles y al alcance del proyecto. En este contexto, se ha decidido desarrollar en primera instancia un driver basado en Python para la solución tanto remota como local.

El objetivo principal del driver es recopilar y organizar los datos. El script de Python, encargado de obtener las muestras de la estación, las organiza en un formato CSV separado por puntos y comas, lo cual podría dificultar su análisis posterior. Para superar este desafío, se plantea que el driver de la solución remota cargue este archivo organizado a la nube utilizando la API de Google Sheets. Al mismo tiempo, se almacenarán los datos para una solución local, donde se accederá a través de una Interfaz Humano-Máquina (HMI) que registrará las muestras censadas por la estación.

Una vez en la nube, se busca acceder a esta información mediante web services, web hooks y otras posibilidades que ofrece la nube. Se ha seleccionado Node-RED como la herramienta para procesar los datos y desarrollar la interfaz, permitiendo a los usuarios interactuar con los datos y

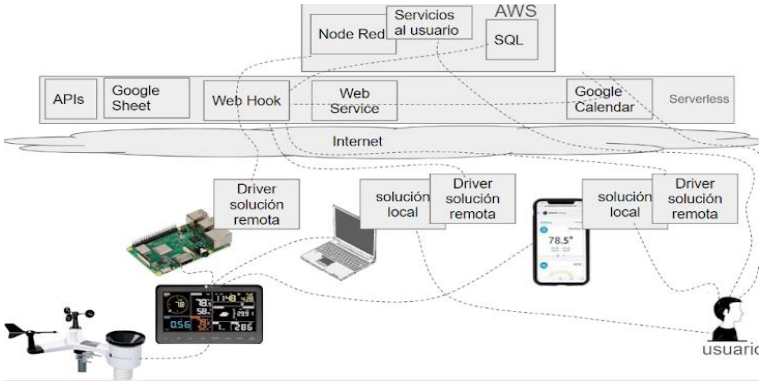
visualizarlos de manera remota. Para lograr esto, se plantea la necesidad de contar con una IP pública, y para ello, se considera utilizar una máquina virtual proporcionada por Amazon Web Services (AWS). En esta máquina virtual, se podrán utilizar servicios como Node-RED, SQL, Mosquito, entre otros servicios en la nube.

En Node-RED, se planifica acceder a la información recopilada por la API de Google Sheets, procesarla y visualizarla en una HMI creada en Node-RED. Esta HMI estará diseñada con los atributos necesarios para satisfacer las necesidades de los interesados. Además, se contempla la integración de Google Calendar para programar eventos en la estación. Para esta conexión, se anticipa la utilización de un web service como intermediario que recopile los datos del calendario, y se accederá a este web service en Node-RED.

Al acceder a la IP pública de la máquina virtual en el puerto de Node-RED y su dashboard, los usuarios podrán interactuar con la HMI de forma remota y hacer uso de los servicios que necesitan. Este enfoque integrado busca proporcionar una solución completa y accesible para satisfacer las diversas necesidades de los usuarios en el ámbito de la radioastronomía.

Figura 3.

Propuesta de solución.



Nota. Propuesta inicial de la solución IoT propuesta.

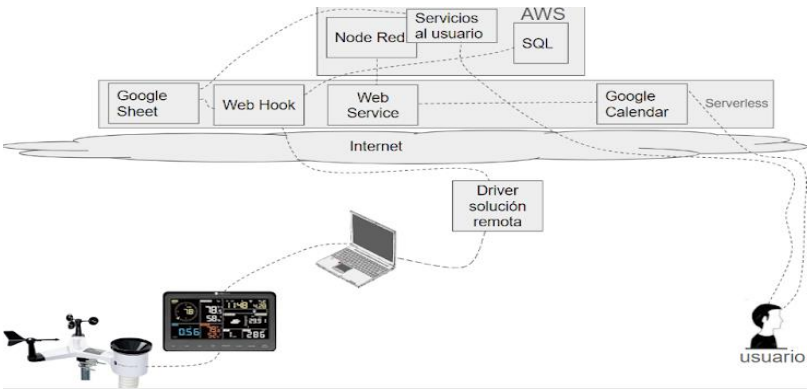
2.3 Solución acotada a los requerimientos del proyecto

Reconociendo la trascendencia de la solución propuesta, cabe destacar que, por una decisión estratégica, nos enfocamos inicialmente en la implementación remota ilustrada en la figura 4. Este enfoque se fundamenta en una estrategia de desarrollo secuencial, ya que, al abordar primero la solución remota, buscamos una base sólida para fases posteriores del proyecto.

La solución remota se ha diseñado para integrarse sin inconvenientes con la solución local. Esta arquitectura flexible, que busca implementar en gran medida varios aspectos de forma serverless, abre la puerta a diversas posibilidades, como la incorporación de funciones adicionales como GPS, alertas y otras innovaciones que la creatividad pueda proporcionar. La sincronización entre ambas soluciones ofrece un espacio dinámico para futuras expansiones y mejoras, facilitando una adaptación ágil a medida que evolucionan las necesidades y oportunidades del proyecto. Por ello, sentar unas bases sólidas en el desarrollo de este resulta un aspecto importante a tener en cuenta.

Figura 4.

Solución acotada a los requerimientos del proyecto



Nota. Solución acotada a los requerimientos del proyecto sin el driver local.

2.3.1 Solución conexión entre consola y máquina, toma y registro de datos.

Para la conexión entre consola y máquina se implementó el manual "3.3 protocolo toma de datos" que tenía el grupo de investigación Radiogis que nos compartió para realizar la correcta conexión entre la consola Davis y el PC. Siguiendo los siguientes pasos:

- Para iniciar la sincronización de la estación se debe presionar el botón "done" de la consola dos veces y mantenerlo presionado hasta que muestre las variables de medida.
- Activar el entorno virtual Estacion_ambiental, para esto abra una terminal y ejecute el comando "source estacion_ambiental/bin/activate".

Se ejecutan los códigos diseñados para la toma de datos compartidos por el grupo de investigación RadioGIS los cuales tienen un funcionamiento de conectar el pc o máquina que se esté utilizando para la toma de datos, el cual nos permite tomar n muestras cada x tiempo (segundos) y nos los almacena en un archivo .csv. Los comandos para realizar estos procesos son los siguientes:

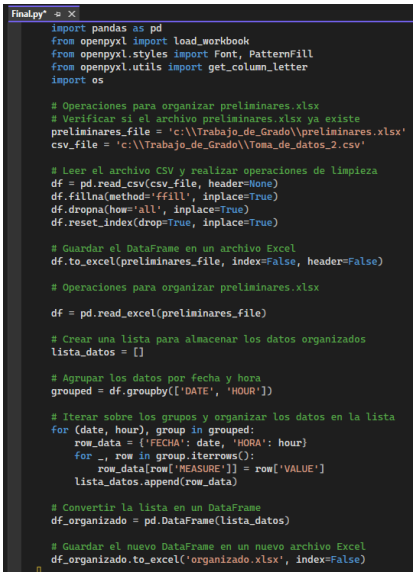
- Luego el comando "cd /home/comdiguis/Desktop/CASIRI/ESTACION" luego de esto el comando "python toma_estacion.py". Luego de esto se le pedirá que coloque cuantas tomas desea que se realicen y el tiempo entre ellas (valor expresado en segundos).
- En la ruta "/home/comdiguis/Desktop/CASIRI/ESTACION" se encuentra el archivo datos.csv en el cual queda almacenado el histórico de los datos tomados.

Cabe aclarar que las direcciones dependen de dónde estén almacenados los archivos que se van a ejecutar. Luego de implementar esos códigos, se desarrolló un código py para organizar los

datos en un Excel por columnas cada variable, para poder subir a la nube el archivo Excel organizado. El código implementado se muestra en la figura 5.

Figura 5.

Código Python para organizar los datos de la estación.



```python
import pandas as pd
from openpyxl import load_workbook
from openpyxl.styles import Font, PatternFill
from openpyxl.utils import get_column_letter
import os

# Operaciones para organizar preliminares.xlsx
# Verificar si el archivo preliminares.xlsx ya existe
preliminares_file = 'c:\\Trabajo_de_Grado\\preliminares.xlsx'
csv_file = 'c:\\Trabajo_de_Grado\\Toma_de_datos_2.csv'

# Leer el archivo CSV y realizar operaciones de limpieza
df = pd.read_csv(csv_file, header=None)
df.fillna(method='ffill', inplace=True)
df.dropna(how='all', inplace=True)
df.reset_index(drop=True, inplace=True)

# Guardar el DataFrame en un archivo Excel
df.to_excel(preliminares_file, index=False, header=False)

# Operaciones para organizar preliminares.xlsx

df = pd.read_excel(preliminares_file)

# Crear una lista para almacenar los datos organizados
lista_datos = []

# Agrupar los datos por fecha y hora
grouped = df.groupby(['DATE', 'HOUR'])

# Iterar sobre los grupos y organizar los datos en la lista
for (date, hour), group in grouped:
    row_data = {'FECHA': date, 'HORA': hour}
    for _, row in group.iterrows():
        row_data[row['MEASURE']] = row['VALUE']
    lista_datos.append(row_data)

# Convertir la lista en un DataFrame
df_organizado = pd.DataFrame(lista_datos)

# Guardar el nuevo DataFrame en un nuevo archivo Excel
df_organizado.to_excel('organizado.xlsx', index=False)
```

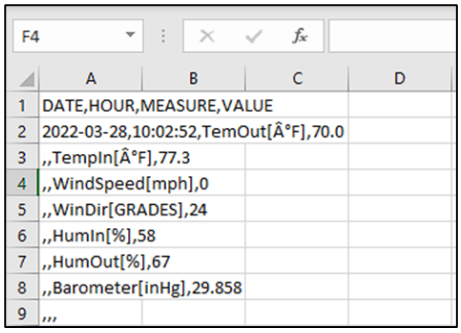
Nota. Código basado en Python para organizar y actualizar los datos de la estación.

El código que se muestra en la figura 5 tiene los siguientes propósitos de funcionamiento:

- Lee un archivo CSV (Toma_de_datos.csv) utilizando pandas y realiza algunas operaciones de limpieza, como llenar los valores faltantes hacia adelante (fillna(method='ffill')), eliminar filas completamente nulas (dropna(how='all')), y restablecer los índices del DataFrame (reset_index).

Figura 6.

Visualización del archivo Toma_de_datos.csv.



	A	B	C	D
1	DATE,HOUR,MEASURE,VALUE			
2	2022-03-28,10:02:52,TemOut[Â°F],70.0			
3	,,TempIn[Â°F],77.3			
4	,,WindSpeed[mph],0			
5	,,WinDir[GRADES],24			
6	,,HumIn[%],58			
7	,,HumOut[%],67			
8	,,Barometer[inHg],29.858			
9	,,,			

Nota. Visualización del archivo csv generado por la estación meteorológica Davis.

- Guarda el DataFrame limpio en un nuevo archivo Excel llamado **preliminares.xlsx**.

Figura 7.

Tabulación de los datos del archivo preliminares.xlsx

	A	B	C	D
1	DATE	HOUR	MEASURE	VALUE
2	2022-03-2	10:02:52	TemOut[°F]	70.0
3	2022-03-2	10:02:52	TempIn[°F]	77.3
4	2022-03-2	10:02:52	WindSpeed[mph]	0
5	2022-03-2	10:02:52	WinDir[GRADES]	24
6	2022-03-2	10:02:52	HumIn[%]	58
7	2022-03-2	10:02:52	HumOut[%]	67
8	2022-03-2	10:02:52	Barometer[inHg]	29.858
9	2022-03-2	10:02:52	Barometer[inHg]	29.858
10	2022-03-2	11:45:31	TemOut[°F]	70.7
11	2022-03-2	11:45:31	TempIn[°F]	74.1
12	2022-03-2	11:45:31	WindSpeed[mph]	0
13	2022-03-2	11:45:31	WinDir[GRADES]	25
14	2022-03-2	11:45:31	HumIn[%]	60
15	2022-03-2	11:45:31	HumOut[%]	62
16	2022-03-2	11:45:31	Barometer[inHg]	29.814
17	2022-03-2	11:45:31	Barometer[inHg]	29.814

Nota. Tabulación de los datos del archivo preliminares.xlsx.

- Lee el archivo Excel recién creado (preliminares.xlsx) utilizando pandas.
- Agrupa los datos por las columnas 'DATE' y 'HOUR'.

- Itera sobre los grupos y organiza los datos en una lista llamada lista_datos.
- Convierte la lista de datos en un nuevo DataFrame (df_organizado).
- Guarda este nuevo DataFrame en un nuevo archivo Excel llamado organizado.xlsx.

Figura 8.

Visualización de la tabla con los datos organizados del archivo organizado.xlsx.

	A	B	C	D	E	F	G	H	I
1	FECHA	HORA	TemOut[°F]	TempIn[°F]	WindSpeed[mph]	WinDir[GRADES]	HumIn[%]	HumOut[%]	Barometer[inHg]
2	2017-05-14	19:37:26	75,4	78,9	0	197	68	73	29,789
3	2017-05-14	19:37:28	75,4	78,9	0	197	68	73	29,789
4	2017-05-14	19:37:30	75,4	78,9	0	197	68	73	29,787
5	2017-05-14	21:00:01	75,4	78,9	0	196	68	73	29,787

Nota. Visualización de la tabla con los datos organizados del archivo organizado.xlsx.

2.3.2 Solución almacenamiento en la nube.

Para llevar a cabo la actualización de un archivo en Google Sheets que se utilizó como almacenamiento en la nube, se desarrolló una API mediante la Consola de Desarrolladores de Google, se llevaron a cabo los siguientes pasos en la implementación de la API:

- Se accede a la Consola de Desarrolladores de Google.
- Se inicia sesión con una cuenta de Google o se crea una nueva si es necesario.
- Se selecciona un proyecto existente o se crea uno nuevo.
- Se realiza la configuración de Credenciales y cuenta de servicio.
- Se navega al menú "API y servicios" > "Credenciales".
- Se crean nuevas credenciales seleccionando "Cuenta de servicio".
- Se configura la cuenta de servicio proporcionando un nombre descriptivo y asignando el rol de "Editor" al proyecto.

- Opcionalmente, se agrega una dirección de correo electrónico de "Propietario de los datos" según sea necesario.
- Se hace clic en "Continuar" y, si es necesario, se configuran permisos adicionales en la página "Permisos".
- En la sección "Cuenta de servicio", se selecciona "Crear una clave". Se elige el tipo de clave, recomendando el formato JSON.
- Se descarga el archivo JSON que contiene las credenciales de la cuenta de servicio. El archivo se guarda en un lugar seguro, ya que será necesario para la autenticación en el script de Python.

Estos pasos forman parte del proceso de implementación de la API para la actualización de un archivo en Google Sheets a través de la Consola de Desarrolladores de Google. El archivo JSON descargado se utilizará en el código Python para autenticar y ejecutar las operaciones necesarias en las hojas de cálculo de Google Sheets.

Figura 9.

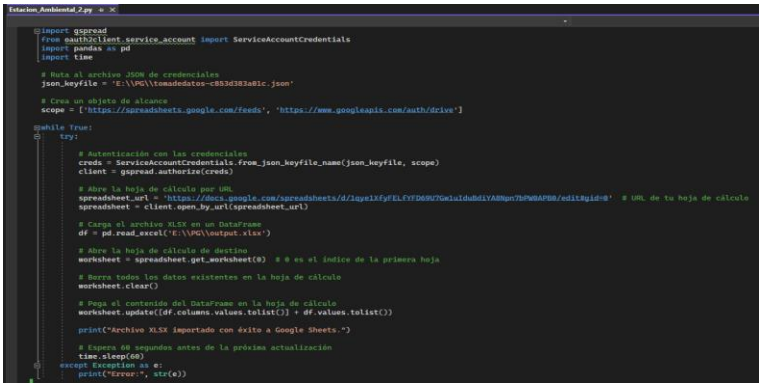
Credenciales del API archivo JSON.

Nota. Credenciales del API archivo JSON.

Para poder subir los datos tomados se implementó el siguiente código Python utilizando la API anteriormente mencionada.

Figura 10.

Código Python para subir datos a la nube.



```python
import gspread
from oauth2client.service_account import ServiceAccountCredentials
import pandas as pd
import time

# Ruta al archivo JSON de credenciales
json_keyfile = 'E:\\PG\\tomadedatos-c853d383a81c.json'

# Crea un objeto de alcance
scope = ['https://spreadsheets.google.com/feeds', 'https://www.googleapis.com/auth/drive']

while True:
    try:

        # Autenticación con las credenciales
        creds = ServiceAccountCredentials.from_json_keyfile_name(json_keyfile, scope)
        client = gspread.authorize(creds)

        # Abre la hoja de cálculo por URL
        spreadsheet_url = 'https://docs.google.com/spreadsheets/d/1qye1XfyFELfYFDM9U7GwluIduBdiYA8Nqn7bPW0APB0/edit#gid=0'  # URL de tu hoja de cálculo
        spreadsheet = client.open_by_url(spreadsheet_url)

        # Carga el archivo XLSX en un DataFrame
        df = pd.read_excel('E:\\PG\\output.xlsx')

        # Abre la hoja de cálculo de destino
        worksheet = spreadsheet.get_worksheet(0)  # 0 es el índice de la primera hoja

        # Borra todos los datos existentes en la hoja de cálculo
        worksheet.clear()

        # Pega el contenido del DataFrame en la hoja de cálculo
        worksheet.update([df.columns.values.tolist()] + df.values.tolist())

        print("Archivo XLSX importado con éxito a Google Sheets.")

        # Espera 60 segundos antes de la próxima actualización
        time.sleep(60)
    except Exception as e:
        print("Error:", str(e))
```

Nota. Código Python para subir datos a la nube.

Al ejecutar el código de la figura 10. Nos permite almacenar en la nube los datos de la estación CASIRI, quedando almacenados por fecha, hora y variable, como se muestra en la figura 11.

Figura 11.

Datos almacenados en la nube

Toma_de_Datos ☆ Guardado en Drive

Archivo Editar Ver Insertar Formato Datos Herramientas Extensiones Ayuda

Menús 100% € % .0 .00 123 Predet... 10 B I

M3 fx

	A	B	C	D	E	F	G	H	I
1	FECHA	HORA	TemOut[°F]	TempIn[°F]	WindSpeed[mph]	WinDir[GRADES]	HumIn[%]	HumOut[%]	Barometer[inHg]
2	2017-05-14	19:37:26	75,4	78,9	0	197	68	73	29,789
3	2017-05-14	19:37:28	75,4	78,9	0	197	68	73	29,789
4	2017-05-14	19:37:30	75,4	78,9	0	197	68	73	29,787
5	2017-05-14	21:00:01	75,4	78,9	0	196	68	73	29,787
6	2017-05-23	21:00:03	75,4	78,9	0	197	68	73	29,789
7	2017-05-24	21:10:54	75,4	78,9	0	197	68	73	29,789
8	2017-05-25	21:10:55	76	78,8	0	263	74	78	29,882
9	2017-05-26	22:22:10	76	78,8	0	263	74	78	29,882
10	2017-05-27	22:22:11	76	78,8	0	263	74	78	29,882
11	2017-06-14	22:25:22	73	74	0	274	67	62	29,771
12	2017-06-15	22:25:23	73	74	0	274	67	62	29,771

Nota. Datos almacenados en la nube.

2.3.3 Desarrollo de la interfaz HMI en Node-Red.

En el marco de nuestro proyecto, pretendemos emplear el potente conjunto de herramientas de Node-RED, específicamente sus dashboards, para desarrollar interfaces gráficas de usuario (GUI) altamente efectivas. Los dashboards de Node-RED ofrecen una solución integral para la visualización y control de datos en tiempo real, permitiéndonos diseñar interfaces intuitivas y personalizadas. Mediante la implementación de widgets versátiles, como gráficos dinámicos, indicadores y controles interactivos, es posible crear HMIs adaptadas a las necesidades específicas de los interesados. Estos dashboards no solo facilitan la comprensión visual de datos complejos, sino que también proporcionan a los usuarios la capacidad de interactuar de manera efectiva con el sistema. La flexibilidad y la facilidad de configuración de los dashboards de Node-RED son elementos clave para optimizar la experiencia del usuario y mejorar la eficiencia operativa esperada de nuestro proyecto. En la figura 12 se muestra un ejemplo de un HMI creado a base de dashboards proporcionados por Node-red para automatizar el proceso de configuración de paneles de control de máquinas industriales.

Figura 12.

Interfaz HMI en Node-RED

Nota. Interfaz HMI en Node-RED

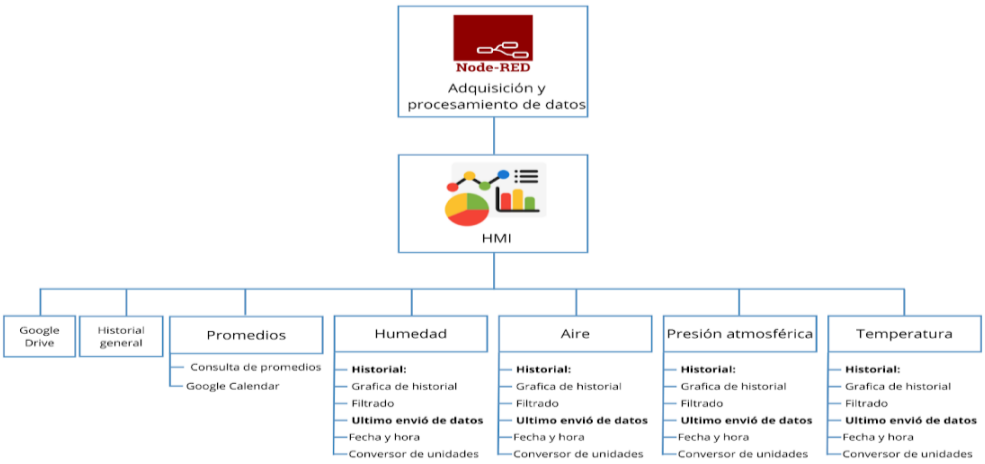
Un planteamiento inicial para el desarrollo de la interfaz HMI fue el mostrado en la figura 13. La estación tiene la capacidad de capturar datos que abarcan la temperatura ambiente interna y externa, la presión atmosférica, la humedad dentro y fuera de la estación, así como la velocidad y dirección del viento. Se ha concebido la asignación de una sección específica en la interfaz para cada una de estas variables. Esto permitirá la visualización en tiempo real a través de indicadores de color, con la opción de ajustar las unidades de medida. Además, se ofrecerá la posibilidad de visualizar la hora y fecha del último envío de datos. Asimismo, se integrará una gráfica que mostrará un historial de datos, brindando al usuario la capacidad de filtrar las fechas de interés. Al desplazarse sobre la gráfica, se proporcionarán indicativos de fecha, hora y valor de la muestra seleccionada.

Esta estructura facilitará el análisis del comportamiento de cada variable, suministrando información precisa para anticipar proyecciones futuras. Dichos datos son cruciales para la caracterización de sitios con potencial para la radioastronomía y para la monitorización en tiempo

real. Se han contemplado apartados adicionales con enfoques diversos. Una sección compartirá un enlace a Google Drive con datos relevantes sobre el desarrollo de la interfaz, destinados a futuros desarrolladores para facilitar la comprensión e integración de nuevas tecnologías. Otro apartado se centrará en la presentación de promedios de datos, con una gráfica que indicará promedios según la necesidad del usuario. También se integrará Google Calendar, permitiendo al usuario editar eventos y programar correos electrónicos detallados sobre estos promedios. La última sección consistirá en un registro histórico de todos los datos capturados por la estación, funcionando como visualización de la base de datos en la nube implementada con la API de Google Sheets.

Figura 13.

Diagrama adquisición y procesamiento de datos



Nota. Diagrama adquisición y procesamiento de datos.

Como se evidencia en la figura 13, se busca proporcionar al usuario una interfaz que integre diversos servicios para satisfacer sus necesidades. A continuación, se detallarán el diseño y la implementación de cada uno de estos servicios.

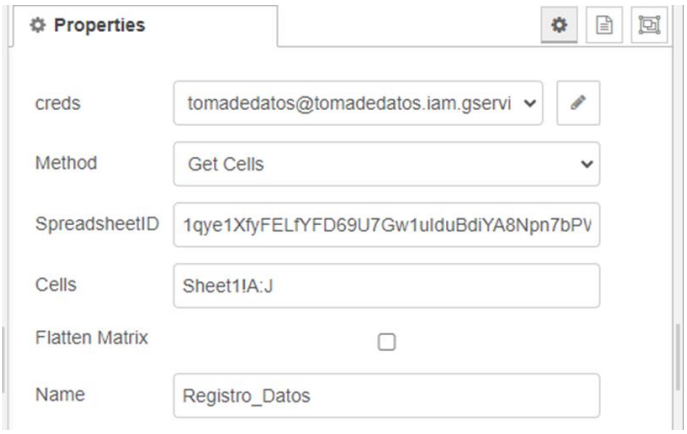
2.3.3.1 Servicio de consulta de Historial general de datos registrados

2.3.3.1.1 Conexión entre la base de datos en la nube y Node-RED.

El desarrollo del servicio que permite la consulta del historial completo de datos registrados por la estación meteorológica CASIR consideró varios aspectos fundamentales, como lo fue la conexión entre la API de Google sheets y el GateWay de Node-Red. Gracias a un nodo proporcionado por Google se pueden asociar estas dos tecnologías mediante el uso del Json generado por la cuenta de servicio asociada a la API de Google Sheets. Este nodo llamado "GSheet" tiene en su configuración un espacio para digitar el id del sheet, la clave Json de la API, la hoja, y por último las filas y columnas de donde se desea extraer los datos como se ve en la figura 14. Este nodo funciona de tal forma que al recibir un mensaje a su entrada, evalúa los datos digitados en su configuración y refleja los datos extraídos del sheet en un msg.payload a su salida.

Figura 14.

Configuración del nodo Gsheet

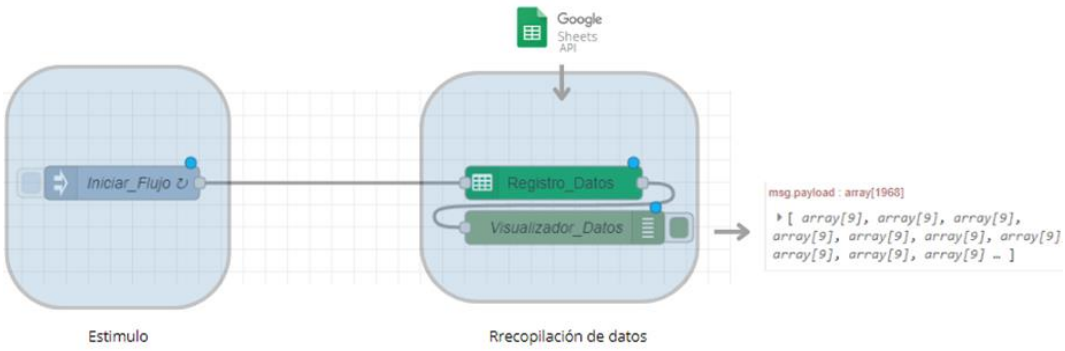


Nota. Configuración del nodo GSheet.

Para un correcto funcionamiento de este nodo se planteó generar periódicamente un mensaje que contenga una carga útil msg.payload gracias a un nodo Inject como se muestra en la figura 15, así cuando este mensaje llegue al nodo GSheet este retornara los valores deseados en forma de arrays, esto con el fin de identificar cambios en el sheet y cargar estos nuevos datos a Node-RED.

Figura 15.

Conexión entre la base de datos en la nube y Node-Red



Nota. Conexión entre la base de datos en la nube y Node-Red

2.3.3.1.2 Depuración y visualización de datos.

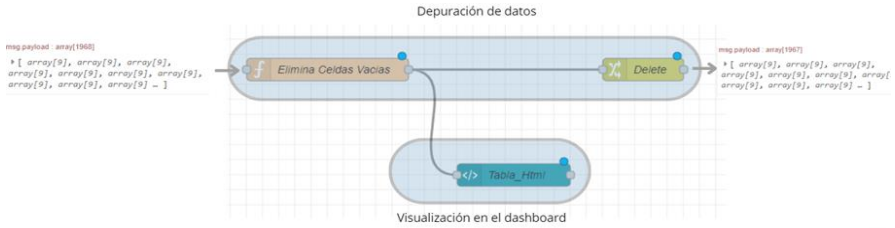
Una vez que los datos se recopilaron en forma de arrays, se buscó una limpieza de datos para su futura presentación en el panel de control, tal como se muestra en la figura 16. Se hace referencia a este proceso como "limpieza" debido a la identificación de una falla que generaba una sobrecarga en la plataforma debido al elevado flujo de datos. Esta problemática se originaba en la presencia recurrente de celdas vacías en el archivo CSV cada vez que se actualizaba el Sheet.

Para abordar este problema, se implementó una depuración de datos a través de un nodo Function. Además, se introdujo un nodo Change con la función "Delete" para eliminar el primer

array de datos en el mensaje. Este array contiene los encabezados de las columnas y no presentaban valores numéricos que requieren procesamiento posterior.

Figura 16.

Depuración y visualización de datos.



Nota. Depuración y visualización de datos.

El código en JavaScript contenido en el nodo function llamado “Elimina Celdas Vacias”, como se ilustra en la figura 17, realiza una iteración sobre todos los valores de los arrays presentes en el msg.payload, eliminando aquellos que se encuentren vacíos.

Figura 17.

Código JavaScript function “Elimina Celdas Vacías”

```
if (msg.payload.celda !== "") {
return msg;
} else {
return null;
}
```

Nota. Código JavaScript function “Elimina Celdas Vacías”.

Para eliminar el primer array de datos contenidos en la carga útil del mensaje, se configuro en las reglas del nodo Change la eliminación de la columna 0 correspondiente a los encabezados de los datos.

Con el fin de lograr una visualización integral de todos los datos en el panel de control, se integró un nodo "ui_template". Este nodo permite la ejecución de código HTML en su interior y lo presenta en un grupo específico asignado en el panel de control. En esta instancia, decidimos establecer un grupo denominado "Historial de Datos" en el panel de control. Este grupo proporciona al usuario la capacidad de revisar todo el histórico de datos recopilados por la estación, junto con sus correspondientes fechas y horas. El código en HTML contenido en el nodo ui_template, como se ilustra en la figura 18, presenta un conjunto de estilos y elementos HTML que se utilizan para la creación y estilización de una tabla dinámica en un entorno web. Este código se ha diseñado para ser integrado en un panel de control y permite la visualización estructurada de datos en una tabla.

La sección de estilos CSS define el aspecto visual de la tabla, eliminando las líneas de borde y aplicando colores de fondo distintivos a las filas y celdas. La estructura de clases como "first-row", "odd-row", y "even-row" proporciona una apariencia diferenciada para la primera fila y filas alternas, mejorando la legibilidad y presentación visual que se ilustra en la figura 18.

Dentro del elemento con el identificador "myContainer", se encuentra la tabla construida dinámicamente utilizando AngularJS (ng-repeat). Este enfoque permite que la tabla se genere automáticamente a partir de los datos presentes en el objeto "msg.payload". Cada fila de la tabla contiene celdas correspondientes a los elementos de un array, y los estilos aplicados anteriormente se utilizan para destacar visualmente diferentes partes de la tabla.

Figura 18.

HTML para la visualización del historial general de datos

```
<style>
    table {
      width: 100%;
      border-collapse: collapse;}
    th, td {  text-align: left;
      padding: 8px;}
    th {background-color: #4CAF50;
color: white;}
   .first-row { color: #FFFFFF;
      background-color: #000000;}
    .odd-row {color: #FFFFFF;
      background-color: #1E1F1E;}
    .even-row {color: #FFFFFF;
background-color: #1A1A1A;}
</style>
<div id="myContainer" style="height: 600px; overflow: auto;">
   <table>
      <tbody>
                <tr ng-repeat="row in msg.payload" ng-class="{ 'first-row': $index === 0,
'odd-row': ($index > 0 && $index % 2 !== 0), 'even-row': ($index > 0 && $index % 2 ===
0) }">
            <td>{{row[0]}}</td>
            <td>{{row[1]}}</td>
            <td>{{row[2]}}</td>
            <td>{{row[3]}}</td>
            <td>{{row[4]}}</td>
            <td>{{row[5]}}</td>
            <td>{{row[6]}}</td>
            <td>{{row[7]}}</td>
            <td>{{row[8]}}</td>
            <td>{{row[9]}}</td>
         </tr>
      </tbody>
   </table>
</div>
```

Nota. HTML para la visualización del historial general de datos

2.3.3.2 Servicio de consulta en línea del historial de datos y filtrado.

Para desarrollar el servicio de consulta en línea del historial de datos,y su respectivo filtrado, se tomó el diseño del servicio de consulta del historial general y se abordó diversos aspectos y desafíos que surgieron durante la concepción e implementación de la solución. El diseño se estructuró en varios grupos clave, destacando los siguientes:

2.3.3.2.1 Tratamiento de datos.

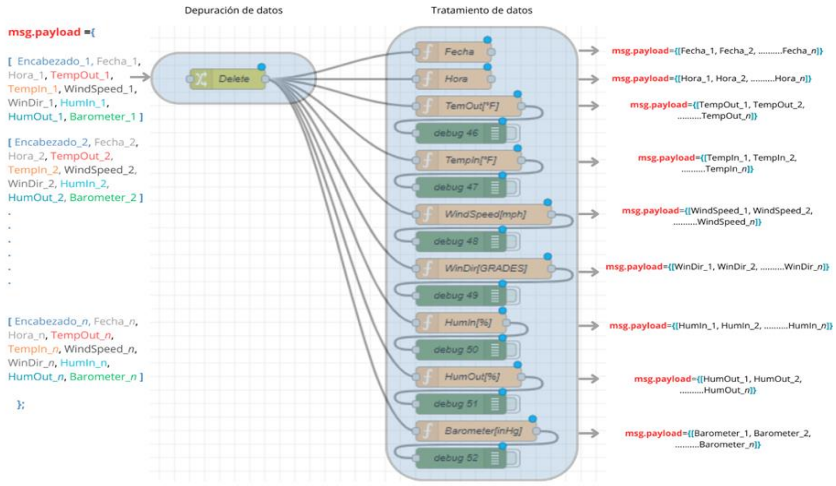
Con el propósito de optimizar la gestión de datos y facilitar el flujo de trabajo, se tomó la decisión de realizar ajustes en el formato de tratamiento de los datos. El nodo GSheet extrae los

datos por filas, lo que resulta en una disposición poco óptima para su posterior visualización gráfica o procesamiento. La práctica más aconsejable es organizar los datos por columnas, como se puede observar en la figura 19.

Para llevar a cabo esta adaptación, se implementaron varios nodos Function como se ilustra en la figura 19. Diseñados para extraer los primeros valores de cada array y consolidarlos en conjuntos de datos individuales. Estos conjuntos reflejan una organización por columnas, siendo el primer array dedicado a la fecha, el segundo a la hora, el tercero a la temperatura exterior, y así sucesivamente. Este enfoque asegura una disposición ordenada y coherente de los datos, facilitando su posterior análisis y representación.

Figura 19.

Reestructuración de datos.



Nota. Reestructuración de datos.

El código presentado en la figura 20 y ejecutado en un nodo Function tiene como objetivo principal optimizar la manipulación de los datos contenidos en msg. payload. El proceso detallado es el siguiente:

- Inicia la creación de un array denominado primeraColumna con la finalidad de almacenar los valores correspondientes a la tercera columna, específicamente para los datos de temperatura exterior. A través de un bucle for, cada valor de la tercera columna es recopilado y agregado a este array.
- Posteriormente, se establece un nuevo array llamado datosSensados. Utilizando la función map, se lleva a cabo una transformación en cada elemento de primeraColumna, sustituyendo las comas por puntos y convirtiendo cada valor en un tipo de dato numérico mediante Number().

Este proceso se replica de manera análoga para cada una de las variables de interés en un nodo function distinto, considerando su posición en msg.payload.

Figura 20.

Código en JavaScript para la extracción de datos.

```
var primeraColumna = [];
for (var i = 0; i < msg.payload.length; i++) {
        primeraColumna.push(msg.payload[i][3]);
}
var datosSensados = primeraColumna.map(function (dato) {
        return Number(dato.replace(',', '.'));
});
msg.payload = datosSensados;
return msg;
```

Nota. Código en JavaScript para la extracción de datos.

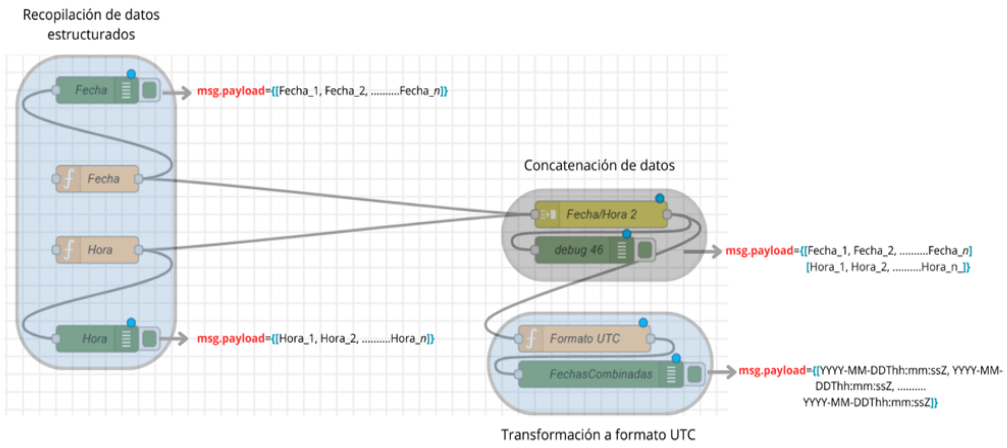
2.3.3.2.2 Formato UTC.

Utilizar el formato UTC (Tiempo Universal Coordinado) en Node-RED fue crucial para el desarrollo de la solución porque ofreció una referencia de tiempo estandarizada en todo el mundo. Esto significa que no se vio afectado por cambios de zona horaria locales, lo que asegura que las fechas y horas se entiendan de la misma manera en cualquier lugar. Esta uniformidad es esencial cuando trabajamos con sistemas distribuidos y conexiones globales, evitando confusiones y

garantizando que los eventos y datos temporales se sincronicen correctamente en Node-RED, por ello la implementación de un flujo específico para la transformación de datos de fecha y hora al formato UTC es fundamental. Para abordar esta tarea se diseñó e implementó un flujo dividido en tres etapas como se ilustra en la figura 21. En la primera fase, se procesan por separado los datos de fecha y hora. Luego, en la segunda etapa, se concatenan mediante un nodo Join en un único mensaje que contiene dos arrays, uno para la fecha y otro para la hora. Finalmente, mediante un nodo Function, se accede a este mensaje consolidado, permitiendo la conversión de los datos al formato UTC.

Figura 21.

Flujo para la conversión de fechas a formato UTC.



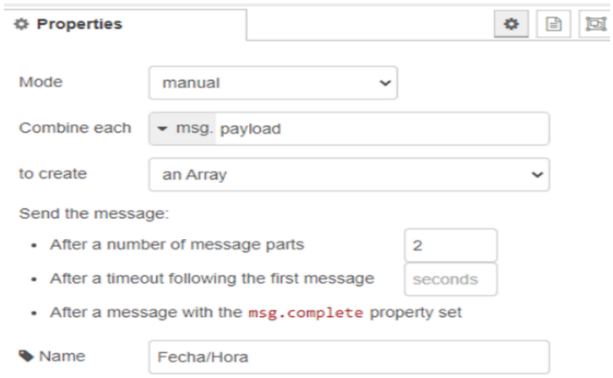
Nota. Flujo para la conversión de fechas a formato UTC.

La configuración necesaria en el nodo Join, visible en la figura 22, se diseñó para concatenar los arrays en un único mensaje sin alterar el orden de los datos. Para lograr esto, se optó por el modo manual, el cual fusiona los mensajes provenientes de msg.payload de dos fuentes

diferentes y los almacena en un array. La transmisión de este array se programó para ocurrir cuando ambas partes estuvieran completas, como se detalla en la figura 22.

Figura 22.

Configuración del bloque nodo Join para concatenar buses de datos.



Nota. Configuración del bloque nodo Join para concatenar buses de datos

Después de haber consolidado los datos en un solo mensaje, se inició la creación de un código en JavaScript, como se muestra en la figura 23, con el propósito de transformar dos arrays de fechas y horas en uno solo, asegurando que el formato de estos valores estuviera en UTC. Este código, implementado en un nodo function, trabaja fusionando los dos conjuntos de datos provenientes del mensaje de entrada (msg.payload).

Durante su ejecución, cada pareja de fechas y horas se combina en un único string según el formato ISO 8601. Acto seguido, se crea un objeto de fecha a partir de esta combinación, el cual se ajusta a la hora coordinada universal (UTC) mediante la adición de 5 horas, considerando la diferencia con la zona horaria colombiana. Tras este ajuste, la fecha y hora se transforman de nuevo

a un string en formato ISO 8601 y se almacenan en un nuevo array denominado "fechasCombinadas". Para facilitar la identificación posterior de otros mensajes, se incluyó la creación de un nuevo campo en el mensaje llamado "msg.topic", asignándole el valor "Fecha". Esto garantiza la coherente representación de los conjuntos de fechas y horas en formato UTC, así como su fácil identificación dentro del entorno de Node-RED.

Figura 23.

Código en JavaScript para la conversión de fechas a formato UTC

```
var fechasArray = msg.payload[0];
var horasArray = msg.payload[1];
var fechasCombinadas = [];
for (var i = 0; i < Math.min(fechasArray.length, horasArray.length); i++) {
        var fechaHoraString = fechasArray[i] + 'T' + horasArray[i] + '.000Z';
        var fechaHoraUTC = new Date(fechaHoraString);
   fechaHoraUTC.setHours(fechaHoraUTC.getHours() + 5);
        var fechaHoraAjustadaString = fechaHoraUTC.toISOString();
   fechasCombinadas.push(fechaHoraAjustadaString);
}
msg.payload = fechasCombinadas;
msg.topic = "Fecha";
return msg;
```

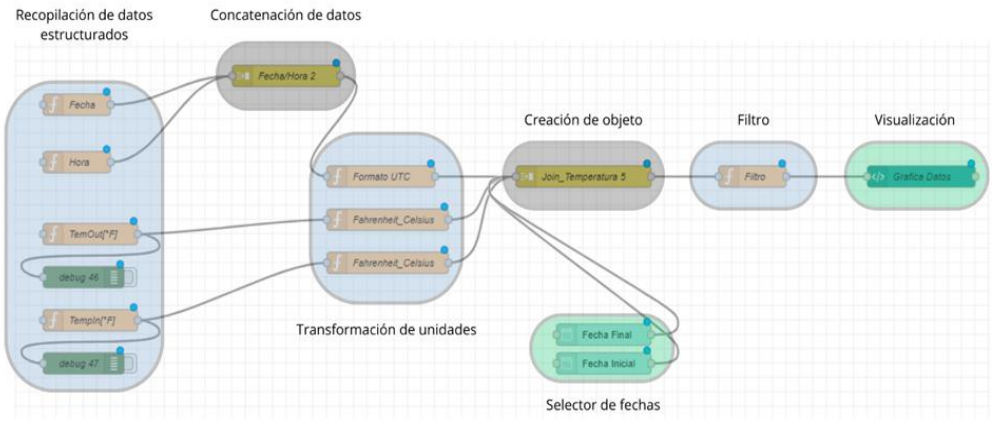
Nota. Código en JavaScript para la conversión de fechas a formato UTC.

2.3.3.2.3 Grafica del archivo histórico de datos y filtro.

En el proceso de implementación de la gráfica correspondiente al archivo histórico de datos en el panel de control, junto con su función de filtrar fechas, se abordaron diversos desafíos en la manipulación de la información. Con el objetivo de superar estos desafíos, se llevó a cabo la ejecución de múltiples etapas mediante la aplicación de nodos function y Join. Finalmente, para la representación visual en el panel de control, se implementaron nodos adicionales, específicamente los nodos Dashboard Template y Date Picker, como se detall de manera ilustrativa en la figura 24.

Figura 24.

Flujo para la gráfica del archivo histórico de datos y filtro



Nota. Flujo para la gráfica del archivo histórico de datos y filtro.

En la figura 24, se presenta el diagrama del flujo diseñado para la manipulación de la información y la visualización del historial de datos de las variables de temperatura interior y exterior, captadas por la estación. A continuación, se examinará en detalle cada fase del proceso, ofreciendo una explicación detallada de su funcionalidad y su operación específica:

2.3.3.2.4 Transformación de unidades.

Con el objetivo de satisfacer las necesidades de los interesados, se tomó la decisión de convertir las unidades de temperatura de grados Fahrenheit a grados Celsius. Para llevar a cabo esta conversión, se emplearon nodos function que iteran sobre cada valor del array, aplicando la siguiente ecuación:

$$℃ = (°\mathrm{F} - 32°) \times \frac{5}{9}$$

Donde "°F" representa los valores de temperatura en grados Fahrenheit que ingresan al nodo function, y "°C" representa los valores de temperatura en grados Celsius obtenidos a la salida

del mismo. Este proceso se implementó mediante un código en JavaScript y a su vez se le asignó un tópico para su identificación, como se muestra detalladamente en la figura 25.

Figura 25.

Código en JavaScript para la conversión de unidades.

```
function convertirFahrenheitACelsius(fahrenheit) {
        var celsius = (fahrenheit-32)/(1.8);
        return parseFloat(celsius.toFixed(2));
}
var datosFahrenheit = msg.payload;
var datosCelsius = datosFahrenheit.map(function (fahrenheit) {
        return convertirFahrenheitACelsius(fahrenheit);
});
msg.payload = datosCelsius;
msg.topic="TempOut";
return msg;
```

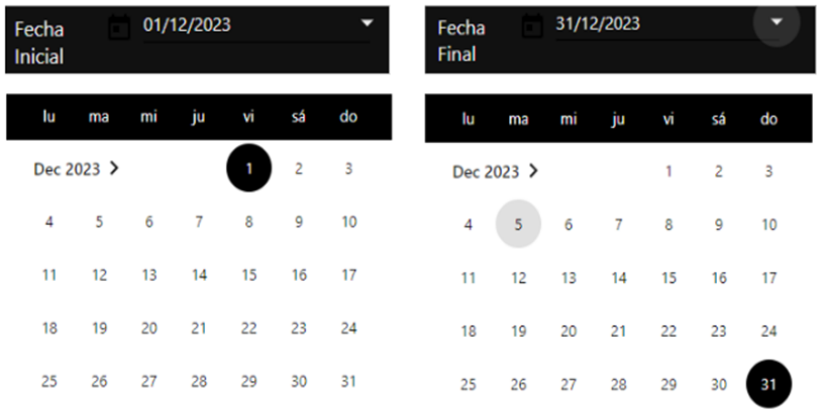
Nota. Código en JavaScript para la conversión de unidades**.**

2.3.3.2.4.1 Selector de fechas.

Para facilitar la implementación de un filtro de fechas en el gráfico del archivo histórico de datos, se incorporaron nodos de dashboard denominados "Date_Picker". Estos nodos ofrecen a los usuarios la capacidad de seleccionar una fecha específica en el Panel de control de Node-RED (Figura 26).

Figura 26.

Apariencia visual de los nodos Date_Picker en el dashboard

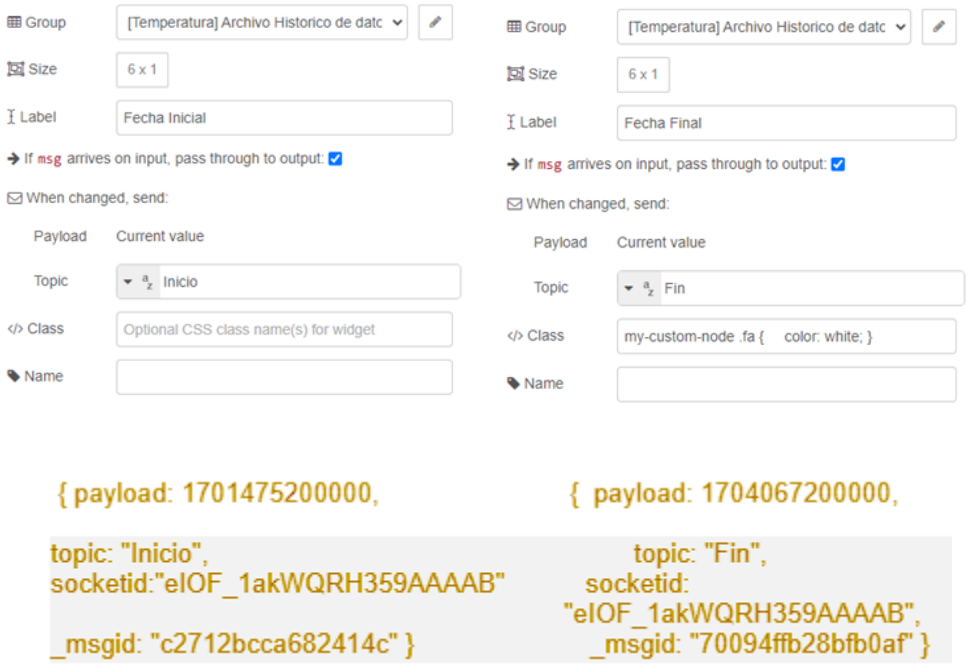


Nota. Apariencia visual de los nodos Date_Picker en el dashboard.

Al seleccionar una fecha, el nodo "Date_Picker" emite un número de tiempo Unix en la carga útil del objeto msg correspondiente a la fecha elegida. Adicionalmente, el nodo proporciona la opción de incluir el atributo "tópico" en el mensaje, lo cual resulta útil para su uso posterior. La configuración de estos nodos y el mensaje generado por ellos se ilustra en la figura 27, donde se pueden observar la carga útil que tiene la fecha en tiempo Unix y los tópicos designados como "Inicio" y "Fin". Estos tópicos facilitan la distinción entre la fecha de inicio y la fecha de finalización, mejorando así la claridad y organización del proceso de selección de fechas.

Figura 27.

Configuración y mensaje generados por los nodos Date_Picker.



Nota. Configuración y mensaje generados por los nodos Date_Picker.

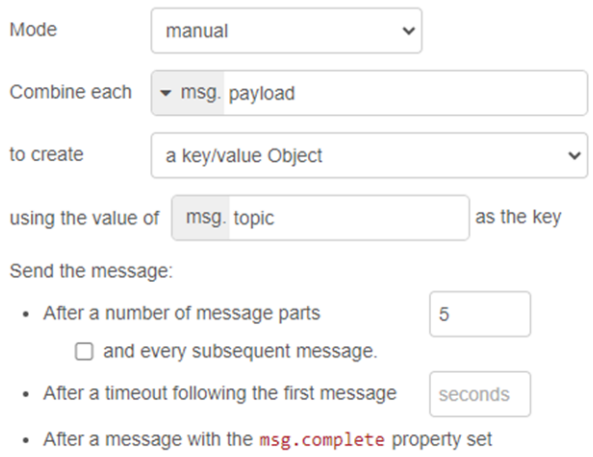
2.3.3.2.4.2 Creación de objeto.

Antes de realizar el filtrado de datos según las fechas iniciales y finales seleccionadas en el panel de control, se llevó a cabo la consolidación de todas las fuentes de datos en un objeto. Esta medida fue crucial para mantener una estructura coherente en el mensaje y permitir la identificación clara de cada conjunto de datos. En contraste, si los datos se hubieran unido en un array, la única manera de distinguir entre ellos sería mediante su posición, y esta cambiaría en función del momento en que el usuario elige una fecha en los date picker.

El objeto fue creado mediante el uso del nodo Join configurado en modo manual. En esta configuración, se estableció que el nodo uniera todas las cargas útiles (payload) de los mensajes en un único objeto. Este objeto presenta la particularidad de tener los nombres de los tópicos de los mensajes como etiquetas (labels), destacando la importancia de que cada mensaje tenga su tópico para la identificación de su contenido. La configuración del nodo Join incluyó la especificación para que el objeto se enviara únicamente cuando las cinco fuentes de información, en este caso, de la sección de temperatura (Fechas en formato UTC, Fecha inicial del filtrado, Fecha final del filtrado, Temperatura Interior y Temperatura Exterior), estuvieran completas. Los detalles específicos de esta configuración se encuentran ilustrados en la figura 28 y la carga útil resultante en la figura 29.

Figura 28.

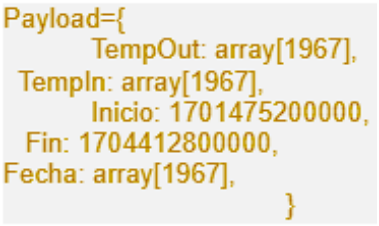
Configuración del nodo Join para la producción del objeto.



Nota. Configuración del nodo Join para la producción del objeto**.**

Figura 29.

Objeto resultante a la salida del bloque Join.



Nota. Objeto resultante a la salida del bloque Join.

2.3.3.2.4.3 Filtrado.

Teniendo una estructura organizada, con las fuentes de datos con una etiqueta, se procedió a hacer el filtrado de valores, para ello se usó un código en JavaScript ilustrado en la figura 30 implementado en un nodo function. Este código realiza un filtrado de datos de temperatura basado en el rango que se seleccionó en el panel de control. En primer lugar, se obtienen las fechas inicial y final del mensaje de entrada y se asignan a las variables fechaInicial y fechaFinal respectivamente. Además, se extraen las temperaturas de entrada y salida, así como las fechas del mensaje de entrada y se almacenan en las variables correspondientes.

Posteriormente, se inicia un bucle que recorre todas las fechas disponibles. Para cada fecha, se intenta convertirla a un objeto de fecha en JavaScript. Si la conversión es exitosa y la fecha cae dentro del rango establecido por fechaInicial y fechaFinal, se añaden las temperaturas correspondientes y la fecha al conjunto de datos filtrados, en caso de que ocurra un error al convertir alguna fecha, se maneja el error y se registra un mensaje de error indicando la fecha problemática.

Finalmente, el mensaje de salida (msg.payload) se actualiza con los conjuntos de datos filtrados de las temperaturas de salida, temperaturas de entrada y fechas. Este conjunto filtrado se utiliza para representar únicamente los datos que cumplen con el criterio de fecha establecido, con el fin de eliminar los datos fuera del rango para su posterior visualización.

Figura 30.

Código en JavaScript para el filtrado de fechas y datos.

```
var fechaInicial = new Date(msg.payload.Inicio);
var fechaFinal = new Date(msg.payload.Fin);
var datosTemperaturaSalida = msg.payload.TempOut;
var datosTemperaturaEntrada = msg.payload.TempIn;
var fechas = msg.payload.Fecha;
var datosFiltradosSalida = [];
var datosFiltradosEntrada = [];
var fechasFiltradas = [];
for (var i = 0; i < fechas.length; i++) {
   try {
         var fechaActual = new Date(fechas[i]);

         if (fechaActual > fechaInicial && fechaActual <= fechaFinal) {
         datosFiltradosSalida.push(datosTemperaturaSalida[i]);
         datosFiltradosEntrada.push(datosTemperaturaEntrada[i]);
         fechasFiltradas.push(fechas[i]);
         }
   } catch (error) {
         node.error("Error al convertir fecha: " + fechas[i]);
   }
}

msg.payload = {
   TempOut: datosFiltradosSalida,
   TempIn: datosFiltradosEntrada,
   Fecha: fechasFiltradas,
};
return msg;
```

Nota. Código en JavaScript para el filtrado de fechas y datos.

2.3.3.2.4.4 Visualización.

Para la visualización, se incluyó un nodo ui_template, el cual tiene el código HTML ilustrado en la figura 31 para llevar a cabo el grafico de las variables de temperatura interna y

externa, con los requerimientos de diseño que se plantearon en un inicio como lo fueron el indicador de fecha hora y valor numérico de las muestras,

El código HTML presenta una página web que utiliza las bibliotecas Moment.js, D3.js y Chart.js para visualizar datos de temperatura en dos gráficos. La estructura del documento incluye elementos para la presentación flexible de gráficos, y la sección de scripts incorpora una función autoejecutable que observa cambios en el mensaje recibido. La función formatear fechas y datos de temperatura, y configura dos gráficos de línea utilizando Chart.js. Los gráficos representan la temperatura de entrada y salida de la estación meteorológica CASIRI.

En cuanto al diseño del tooltip, este se personaliza mediante configuraciones específicas de Chart.js en la sección de opciones. El tooltip se muestra en un formato claro y atractivo, con información detallada sobre las fechas y los valores de temperatura. Se ajusta a un estilo visual coherente con la presentación general de los gráficos, utilizando colores y fuentes que mejoran la legibilidad. Además, se implementa una función de hover que permite resaltar y ampliar la información al pasar el cursor sobre los puntos del gráfico.

Figura 31.

Código HTML para la gráfica de los datos filtrados.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Temperatura de Entrada y Salida</title>
                                                                                <scrip
src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.1/moment.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/d3@6"></script>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <style>
        #tablaTemperaturas {
            color: black;
        }
        canvas {
            width: 100%;
            height: auto;
        }
    </style>
</head>
<body style="display: flex; flex-direction: column; align-items: center;">
    <div style="margin-bottom: 20px; overflow-x: auto;">
        <canvas id="miGrafico" width="600" height="200"></canvas>
        <canvas id="miOtraGrafico" width="600" height="200"></canvas>
    </div>
    <script>
        (function (scope) {
            var savedPayload;
            scope.$watch('msg', function (msg) {
                if (msg) {
                    savedPayload = msg.payload;
                    console.log('Mensaje guardado:', savedPayload);
                    var fechasFormateadasX = savedPayload.Fecha.map(function (fecha) {
                                                                    return    moment(fecha
'YYYY-MM-DDTHH:mm:ss.SSSZ').format('MMM YYYY');
                    });
                        var fechasFormateadasTooltip = savedPayload.Fecha.map(functior
(fecha) {
                                                                    return    moment(fecha
'YYYY-MM-DDTHH:mm:ss.SSSZ').format('dddd, MMMM D, YYYY h:mm:ss A');
                    });
                    var datosEntrada = savedPayload.TempIn;
                    var datosSalida = savedPayload.TempOut;
                    var ctx = document.getElementById('miGrafico').getContext('2d');
                    var config = {
                        type: 'line',
                        data: {
```

Nota. Código HTML para la gráfica de los datos filtrados.

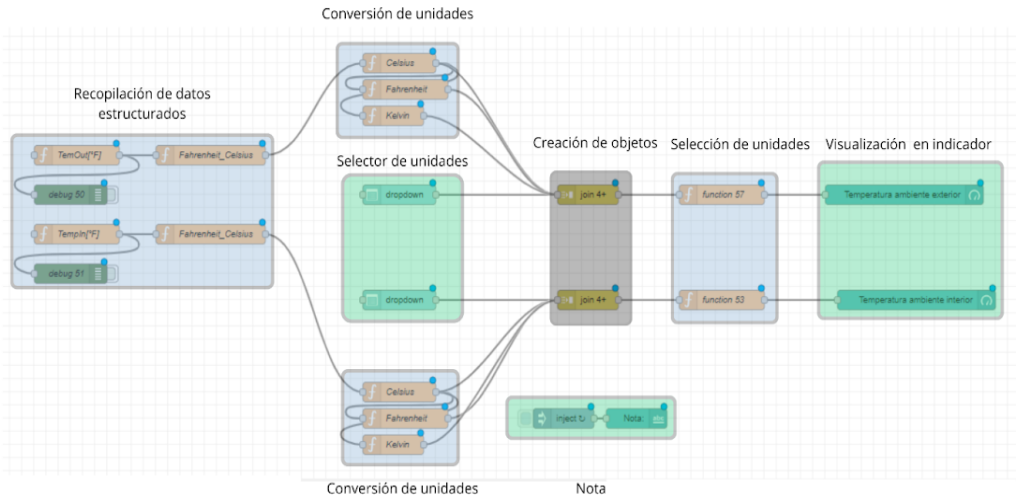
Nota: Es importante destacar que la estructura para el desarrollo del servicio de consulta del historial de datos y filtrado de las demás variables sigue un patrón similar, variando principalmente en la transformación de unidades y en los aspectos visuales que percibe el usuario en la gráfica.

2.3.3.3 Servicio de consulta en línea del último envió de datos.

Con el objetivo de permitir a los usuarios visualizar el último envío de datos almacenado en la base de datos en la nube a través de la API de Google Sheets, se ha desarrollado un nuevo flujo, detallado en la figura 32. Este flujo se ha diseñado para satisfacer las necesidades específicas de los usuarios y se compone de diversas etapas. Este diseño se fundamenta en la recopilación de datos y la modificación de la estructura realizada previamente para el servicio de historial de datos.

Figura 32.

Flujo que proporciona el servicio de consulta de ultimo envío de datos



Nota. Flujo que proporciona el servicio de consulta de ultimo envío de datos.

Una vez reestructurados los buses de datos, se implementaron varias etapas en el flujo, a continuación, se entrará a detalle en cada una de estas etapas:

2.3.3.3.1 Conversión de unidades.

En esta fase, se estructuró en tres bloques de funciones que contienen código JavaScript para extraer el último dato del array resultante de la etapa de recopilación de datos y convertirlo a la unidad correspondiente, según el bloque. Para los datos de temperatura en grados Celsius, se mantiene el dato sin cambios, ya que se asume que el dato ya está en Celsius debido a la implementación de la (Ec.1) en la etapa de recopilación de datos. Para los datos en Fahrenheit, se aplica la ecuación (Ec.2), mientras que, para los datos en Kelvin, se utiliza la ecuación (Ec.3). Estas ecuaciones (Ec.2) y (Ec.3) se implementan en los bloques “”Fahrenheit” y “Kelvin” con los códigos en JavaScript mostrados en las figuras 33 y 34 respectivamente.

$$°\mathrm{F} = \left(℃ \times \frac{9}{5}\right) + 32$$

Figura 33.

Código en JavaScript para la conversión de datos de grados Celsius a Fahrenheit.

```
var celsius = msg.payload;
var fahrenheit = (celsius * 9 / 5) + 32;
var numeroLimitado = parseFloat(fahrenheit).toFixed(2);
msg.payload = numeroLimitado;
msg.topic="Fahrenheit";
return msg;
```

Nota. Código en JavaScript para la conversión de datos de grados Celsius a Fahrenheit.

$$K = ℃ + 273{,}15°$$

Figura 34.

Código en JavaScript para la conversión de datos de grados Celsius a Kelvin.

```
var celsius = msg.payload;
var kelvin = celsius + 273.15;
var numeroLimitado = parseFloat(kelvin).toFixed(2);
msg.payload = numeroLimitado;
msg.topic="Kelvin";
return msg;
```

Nota. Código en JavaScript para la conversión de datos de grados Celsius a Kelvin.

Es importante destacar que la extracción del último dato del array se realiza mediante el código en JavaScript (Figura 35) en el bloque de funciones denominado "Celsius". Su salida se conecta a los bloques "Fahrenheit" y "Kelvin". Esto se emplea con el fin de procesar únicamente el último dato y evitar sobrecargar la máquina con datos innecesarios. Adicionalmente a esto también se agrega un tópico a cada bloque para su posterior identificación del mensaje.

Figura 35.

Código en JavaScript para la extracción del último dato.

```
var datosCelsius = msg.payload || [];
var ultimoDatoCelsius = datosCelsius[datosCelsius.length - 1];
console.log("Último dato en Celsius:", ultimoDatoCelsius);
msg.payload = ultimoDatoCelsius;
msg.topic="Celsius";
return msg;
```

Nota. Código en JavaScript para la extracción del último dato.

2.3.3.3.2 Selector de unidades.

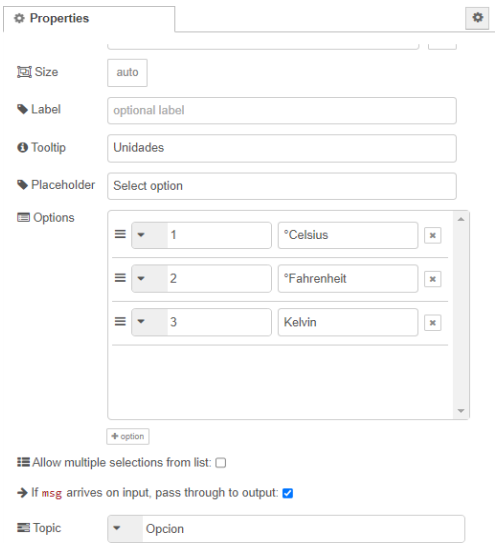
Para permitir al usuario elegir la unidad en la que desea visualizar el último envío de datos, se emplearon dos nodos "dropdown" en el panel de control del HMI. Estos nodos, integrados en

el dashboard de Node-RED, proporcionan una interfaz gráfica con menús desplegables configurables, los cuales se utilizaron para la selección de unidades.

En la configuración de los nodos, detallada en la figura 36, se destaca la sección "options", donde se pueden agregar las opciones que aparecerán en los menús desplegables. Cada opción tiene un campo para el nombre visible en el panel de control y otro para el mensaje que se enviará en la carga útil (payload) cuando se seleccione esa opción. En este caso, se incluyeron las opciones "°Celsius", "°Fahrenheit" y "Kelvin" con cargas útiles de tipo entero 1, 2 y 3, respectivamente. Además, los nodos cuentan con una opción de "topic", permitiendo identificar el mensaje mediante un atributo de tópico en el objeto msg para un uso posterior en el flujo de trabajo. Estos nodos se insertaron en el grupo de "Último envío de datos" y se configuraron para las variables de Temperatura ambiente Interior y Exterior con los tópicos llamados "opción".

Figura 36.

Configuración del nodo dropdown



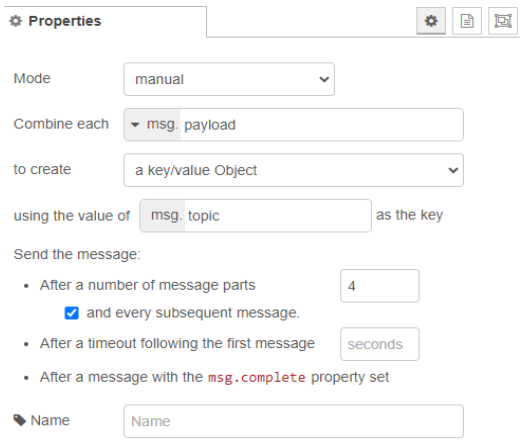
Nota. Configuración del nodo dropdown.

2.3.3.3.3 Creación de objetos.

Con el objetivo de consolidar todas las fuentes de datos en un único mensaje, se implementaron dos nodos "Join" en modo manual, siguiendo un enfoque similar al configurado para el servicio de consulta en línea del historial de datos y filtrado. La diferencia radica en que cada nodo "Join" se ajustó para enviar datos a su salida cada vez que se recibían 4 partes distintas, considerando que a su entrada llegan 4 fuentes de datos diferentes, cada una identificada por su tópico. La configuración detallada de estos nodos se presenta en la figura 37.

Figura 37.

Configuración del bloque Join para la construcción de un objeto con 4 labels.



Nota. Configuración del bloque Join para la construcción de un objeto con 4 labels.

2.3.3.3.4 Selección de unidades.

Con un objeto que almacena datos identificados por sus tópicos, procedimos a filtrar la salida del objeto según el valor del tópico "opción" en la carga útil del mensaje "msg". Esta tarea se realizó utilizando un bloque de función con un código JavaScript básico, como se muestra en la

figura 38. A través de condicionales "if", se evaluó el valor de este tópico y se asignó a la salida el dato correspondiente.

Figura 38.

Algoritmo en JavaScript para la selección de unidades.

```
if (msg.payload.Opcion === 1) {
    msg.payload = msg.payload.Celsius;
}
if (msg.payload.Opcion === 2) {
    msg.payload = msg.payload.Fahrenheit;
}
if (msg.payload.Opcion === 3) {
    msg.payload = msg.payload.Kelvin;
}
return msg;
```

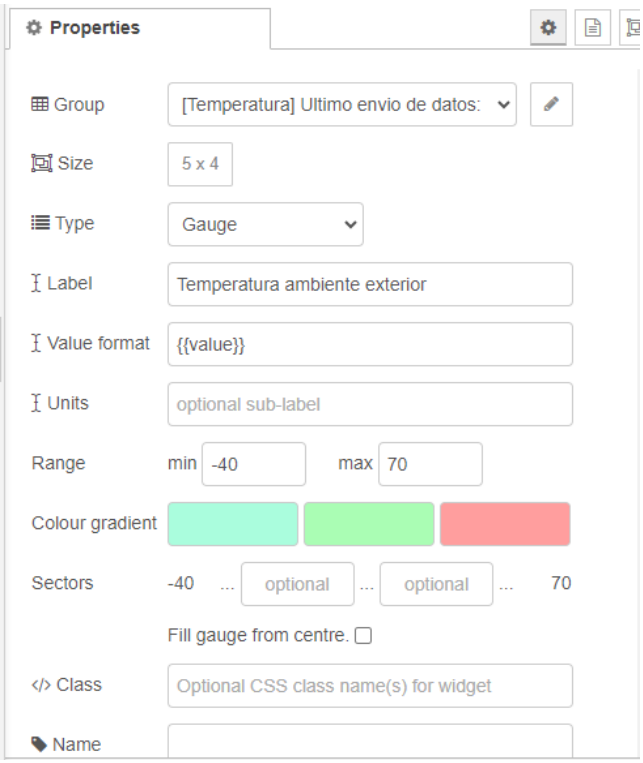
Nota. Algoritmo en JavaScript para la selección de unidades.

2.3.3.3.5 Visualización en indicador.

Contando con el dato seleccionado por el usuario, el siguiente paso consistió en presentarlo de manera intuitiva y fácil de entender. Para lograr esto, incorporamos el nodo de dashboard "ui_gauge" dentro del grupo "Último envío de datos" en el panel de control. Este nodo visualiza el dato instantáneo que recibe en varios tipos de indicadores, como manómetros, indicadores de nivel, brújulas, entre otros. En nuestra implementación, optamos por el manómetro y configuramos tonos pasteles que van desde el azul hasta el rojo para representar diferentes valores. El nodo proporciona la opción de asignar valores a este rango de colores, y en nuestro caso, establecimos el rango de -40 a 70, correspondientes a los límites de grados Celsius proporcionados por el fabricante de la estación meteorológica Davis. La configuración detallada de este bloque se ilustra en la figura 39.

Figura 39.

Configuración del nodo gauge



Nota. Configuración del nodo gauge.

Además, se incorporó una nota informativa al grupo de dashboard utilizando el nodo "ui_text". El texto de la nota es el siguiente: "Nota: El indicativo de color solo está disponible para unidades en grados Celsius [°C]". Esta adición tiene como objetivo informar al usuario sobre la limitación de la interfaz de usuario, ya que el nodo "ui_gauge" no permite la opción de ajustar los rangos de colores dinámicamente en función de la unidad seleccionada.

Nota: Es importante destacar que la estructura para el desarrollo del Servicio de consulta en línea de último envío de datos de las demás variables sigue un patrón similar, variando

principalmente en la transformación de unidades y en los aspectos visuales que percibe el usuario en el indicador de color.

2.3.3.4 Servicio de programación de eventos con Google Calendar.

Para establecer un servicio que permita programar eventos a través de la interfaz con Google Calendar, se tuvo que tomar decisiones sobre qué tipo de eventos se podrían programar, y se identificaron algunas limitaciones. En el caso de la consola Davis Vantage Pro 2, carece de un sistema para recibir comandos u otro tipo de comunicación que permita controlar el dispositivo. En lugar de ello, la consola se dedica a medir datos meteorológicos y transmitirlos a través del puerto serial.

Dada esta limitación, se decidió implementar la programación de eventos basada en la medición de datos, como el envío de informes al correo electrónico de los promedios de datos en intervalos de tiempo específicos. Para ello fue necesario diseñar primero una conexión entre el Google Calendar y Node-RED haciendo uso de una Web App.

Este servicio web, desarrollado como una interfaz de conexión entre un Google Calendar dedicado a la programación de eventos y la interfaz de Node-Red, se implementó utilizando Google Apps Script. Este script ilustrado en la figura 40, escrito en JavaScript, funciona como una implementación pública accesible mediante solicitudes HTTP. Al ejecutarse, extrae la información de la fecha inicial, fecha final y nombre del próximo evento en el calendario, basándose en la fecha actual del servidor. Posteriormente, devuelve estos datos en formato JSON, utilizando etiquetas para identificar cada valor. La funcionalidad principal está encapsulada en las funciones doGet() y obtenerSiguienteEvento(), las cuales desempeñan un papel clave en la obtención y formateo de la información necesaria.

Figura 40.

Código implementado en una Web App de Google Apps Script

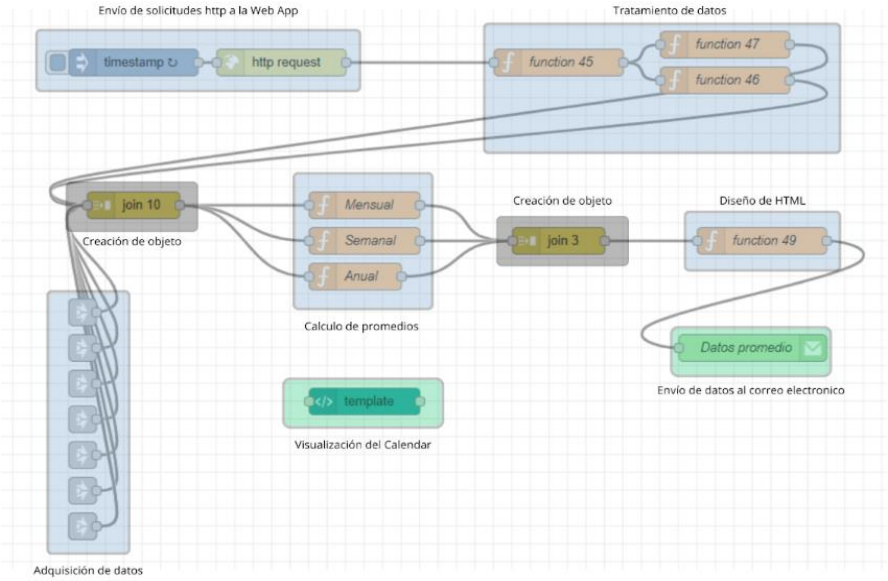
```
function doGet() {
return
ContentService.createTextOutput(obtenerSiguienteEvento()).setMimeType(ContentS
ervice.MimeType.JSON);
}

function obtenerSiguienteEvento() {
var calendarioId=
'31ffeef628be2bc0e3c0f014218d04f0ef54c627b34ceda50624aee339350822@group.
calendar.google.com';

 // Accede al calendario por su ID
 var calendario = CalendarApp.getCalendarById(calendarioId);

 var ahora = new Date();
 var eventos = calendario.getEvents(ahora, new Date(ahora.getTime() + 3655 * 24 *
60 * 60 * 1000));

 // Filtra el próximo evento futuro o el evento que está ocurriendo en el momento
 var proximoEvento = eventos.find(function(evento) {
  return evento.getStartTime() <= ahora && ahora <= evento.getEndTime();
 });

 // Objeto para almacenar la información del evento
 var eventoObjeto = {};

 // Verifica si hay un próximo evento
 if (proximoEvento) {
  // Almacena la información en el objeto
    eventoObjeto['FechaInicial'] = Utilities.formatDate(proximoEvento.getStartTime(),
'GMT-5', 'yyyy-MM-dd HH:mm:ss');
    eventoObjeto['FechaFinal'] = Utilities.formatDate(proximoEvento.getEndTime(),
'GMT-5', 'yyyy-MM-dd HH:mm:ss');
  eventoObjeto['Titulo'] = proximoEvento.getTitle();
 } else {
  Logger.log('No hay eventos futuros en el calendario.');
 }
 return JSON.stringify(eventoObjeto);
}
```

Nota. Código implementado en una Web App de Google Apps Script que sirve de intermediario para la conexión entre Node-RED y Google Calendar.

Posteriormente, se procedió a diseñar un flujo en Node-Red ilustrado en la figura 41, destinado a recibir la información proveniente de esta Web App y realizar un procesamiento posterior.

Figura 41.

Flujo del servicio de programación de eventos con Google Calendar



Nota. Flujo para proporcionar el servicio de programación de eventos con Google Calendar e integración con el correo electrónico.

Este flujo comprende varias etapas, desde la recopilación inicial de datos hasta su presentación visual. A continuación, se detallarán cada una de estas instancias existentes en el flujo.

2.3.3.4.1 Envió de solicitudes http

El inicio del proceso para establecer una comunicación entre la Web App y Node-RED involucró la generación periódica de solicitudes HTTP a la URL de la Web App. Para lograr esto, se programó periódicamente un estímulo al bloque de solicitudes HTTP proporcionado por Node-RED, utilizando un nodo inject. Esta configuración permite evaluar de manera constante los datos proporcionados por la Web App implementada en Google Apps Script. Como resultado de este

flujo continuo, se obtienen datos en formato JSON en cadena de texto, que incluyen la fecha de inicio y final del evento, así como su nombre.

2.3.3.4.2 Tratamiento de datos.

Tras obtener los datos en formato de cadena, se llevó a cabo la conversión inicial a formato JSON, seguida de la separación del dato correspondiente a la fecha inicial y la fecha final. Este proceso demandó la implementación de tres bloques function, cada uno con códigos básicos en JavaScript diseñados para cumplir con estos objetivos específicos.

3.3.3.4.3 ***Adquisición de datos.***

Para recopilar los datos de las variables meteorológicas y sus correspondientes fechas de registro, se implementó una conexión con los flujos diseñados en los servicios previos para el tratamiento de datos. Esta conexión se estableció mediante los nodos "link in" y "link call", que posibilitan una interconexión entre flujos sin mostrar cables en la interfaz, contribuyendo así a mantener un diseño ordenado y libre de excesivos cables en el flujo principal.

2.3.3.4.3 Cálculo de promedios.

Una vez que se tuvo todos los buses de datos debidamente separados, se procedió a consolidarlos en un objeto único. Posteriormente, se llevó a cabo el cálculo de promedios mediante tres nodos function, cada uno encargado de calcular el promedio respectivo para períodos semanales, mensuales y anuales. Aunque el código en JavaScript implementado en los tres nodos es similar, la diferencia radica en los intervalos de tiempo evaluados. El código específico para el cálculo del promedio semanal se presenta en la figura 42, sirviendo como ejemplo del utilizado en los demás bloques.

Figura 42.

Código en JavaScript para el cálculo de promedios semanales, mensuales y anuales.

```
var fechaActual = Math.floor(new Date().getTime());
  var datoWindSpeed = msg.payload.WindSpeed;
    var datoWinDir = msg.payload.WinDir;
    var datoHumIn = msg.payload.HumIn;
    var datoHumOut = msg.payload.HumOut;
    var datoPression = msg.payload.Presion;
    var datoTempOut = msg.payload.TempOut;
    var datoTempIn = msg.payload.TempIn;
    var cantidadMuestras=0;
var FechaInicial = Math.floor(new Date(msg.payload.FechaInicial).getTime());
var FechaFinal = Math.floor(new Date(msg.payload.FechaFinal).getTime());
var limiteInferior = fechaActual - 7 * 24 * 60 * 60*1000;
  var WindSpeed = [];
    var WinDir = [];
    var HumIn = [];
    var HumOut = [];
    var Pression = [];
    var TempOut = [];
    var TempIn = [];
    var Fecha = msg.payload.Fecha;
if (fechaActual >= FechaInicial && fechaActual <= FechaFinal) {
    for (var i = 0; i < Fecha.length; i++) {
        var fechaDato = new Date(Fecha[i]);
        var fechaDatoTiempoUnix = fechaDato.getTime();
       // Si la fecha del dato está dentro de los últimos 30 días
         if (fechaDatoTiempoUnix >= limiteInferior && fechaDatoTiempoUnix <=
fechaActual) {
        cantidadMuestras++;
          WindSpeed.push(datoWindSpeed[i]);
           WinDir.push(datoWinDir[i]);
           HumIn.push(datoHumIn[i]);
           HumOut.push(datoHumOut[i]);
           Pression.push(datoPression[i]);
           TempOut.push(datoTempOut[i]);
           TempIn.push(datoTempIn[i]);
```

Nota. Código en JavaScript para el cálculo de promedios semanales, mensuales y anuales.

2.3.3.4.4 Diseño de la componente HTML.

Con los promedios calculados, se generó un código HTML utilizando un bloque function con código en JavaScript que describe la estructura HTML. Este fragmento de código en JavaScript ilustrado en la figura 43, se encarga de procesar los promedios mensuales, semanales y anuales de datos meteorológicos y generar un código HTML estructurado que presenta esta información de manera organizada en tablas. Primero, se extraen los promedios de las variables meteorológicas de la carga útil del objeto msg.payload. Luego, se utiliza una plantilla HTML para crear una página que exhibe tablas separadas para cada conjunto de datos. El código HTML

incorpora estilos CSS para dar formato al contenido, y utiliza tablas para organizar los datos, con etiquetas y valores correspondientes. Para cada conjunto de datos, se emplea un bucle for...in para iterar sobre las etiquetas y valores, generando filas de tabla que muestran la etiqueta del aspecto meteorológico, su valor y la unidad de medida obtenida mediante la función obtenerUnidad(). Finalmente, el código HTML resultante se asigna al objeto msg.payload, permitiendo su integración en el flujo de Node-RED y su uso en el cuerpo de un correo electrónico.

Figura 43.

Diseño de de la componente HTML adjunta en el correo electrónico.

```
var Mensual = msg.payload.Mensual;

var Semanal = msg.payload.Semanal;
var Anual = msg.payload.Anual;
var htmlCode = `
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Promedios de Datos</title>
    <style>
        #titulo {
            font-size: 20px;
            margin-bottom: 10px;
            text-align: center;
            color: #B5B8B2;
        }

        #infoAdicional {
            font-size: 11px;
            color: #B5B8B2;
            margin-top: 20px;
            margin-left: auto;
            margin-right: auto;
```

Nota. Diseño de la componente HTML adjunta en el correo electrónico.

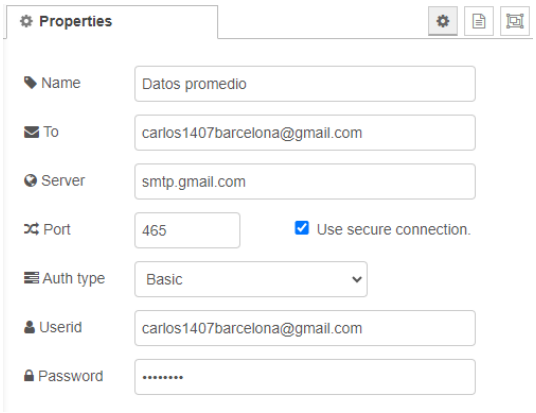
2.3.3.4.5 Envió de datos al correo electrónico.

Con el contenido del correo electrónico completado, se procedió a establecer la conexión entre la interfaz de Node-RED y un servicio de correos, como Gmail. Para ello, se empleó el nodo "email", cuya configuración se ilustra en la figura 44. Esta configuración permite modificar el servidor de envío; en nuestro caso, se utilizó el servidor SMTP por defecto, comúnmente empleado para enviar correos electrónicos en internet. Además, se especificó el puerto predeterminado 465, que garantiza la cifra de la información entre el cliente y el servidor. Se proporcionó la dirección

de correo electrónico del remitente y del destinatario. Para ello, fue necesario generar una contraseña de aplicación en la cuenta de correo electrónico para permitir el acceso a agentes externos. El nodo, con esta configuración, lee los datos ingresados y envía el contenido del correo, que en nuestro caso es el HTML con las tablas detallando los promedios calculados.

Figura 44.

Configuración del nodo email.



Nota. Configuración del nodo email.

2.3.3.4.6 Visualización.

Para incorporar la visualización del Calendar en el panel de control de la interfaz, se empleó el nodo "ui_template" del dashboard de Node-RED. Este nodo facilitó la inserción de código HTML utilizando las bibliotecas D3.js y Chart.js. El código, representado en la figura 45, introduce un marco con el contenido del Google Calendar, registrando su identificador. Además, agrega una nota que sugiere revisar un informe detallado de los promedios en el correo registrado en el bloque, junto con un hipervínculo que dirige al Google Calendar para su edición.

Figura 45.

Código HTML para la visualización del calendar en el dashboard de Node-RED.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Google Calendar Claro</title>
</head>
<body>
    <style>
        iframe {
            border: 2px solid #333;
        }
        #nota {
            margin-top: 20px;
            font-size: 11px;
        }
        #nota strong {
            font-weight: bold;
        }
        #nota a {
            color: #00f;
            /* Color azul para el enlace */
            text-decoration: underline;
            /* Subrayado para indicar que es un enlace */
        }
    </style>
    <iframe

src="https://calendar.google.com/calendar/embed?src=31ffeef628be2bc0e3c0
f014218d04f0ef54c627b34ceda50624aee339350822%40group.calendar.goo
gle.com&ctz=America%2FBogota"
                style="border: 0" width="1100" height="800" frameborder="0"
scrolling="no"></iframe>

    <div id="nota">
        <p><strong>Nota:</strong></p>
            <p>Para un informe completo consulte en el correo electrónico
<strong>carlos1407barcelona@gmail.com</strong>, en
            donde se enviará un reporte semanal, mensual y anual de los datos
sensados por la estación meteorológica
            CASIRI. Para configurar la periodicidad del envío de estos datos al
correo electrónico puede hacer uso del
            Google Calendar y puede acceder a él haciendo clic <a

href="https://calendar.google.com/calendar/u/0?cid=MzFmZmVlZjYyOGJlMm
JjMGUzYzBmMDE0MjE4ZDA0ZjBlZjU0YzYyN2IzNGNlZGE1MDYyNGFlZTM
zOTM1MDgyMkBncm91cC5jYWxlbmRhci5nb29nbGUuY29t"
                target="_blank">aquí</a>. (Recuerde que para poder configurar el
Google Calendar debe pedir permisos al
            administrador)</p>
    </div>
</body>
</html>
```

Nota. Código HTML para la visualización del calendar en el dashboard de Node-RED.

3. Resultados

3.1 Validación del funcionamiento del sistema

Para comprobar el funcionamiento del sistema, fue necesario tener en cuenta toda la estructura que comprende la estación de radioastronomía CASIRI. Este sistema engloba la adquisición de imágenes, RFI y variables ambientales, empleadas en la caracterización de posibles

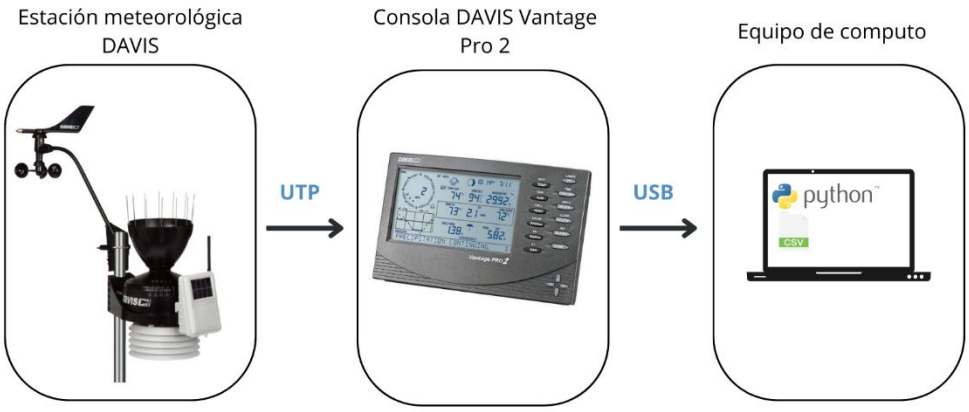
sitios para la radioastronomía. Es relevante destacar que el enfoque central de esta iniciativa fue la creación de un sistema IoT destinado a la visualización, almacenamiento y procesamiento en la nube de las variables ambientales.

La estación de radioastronomía CASIRI se encuentra actualmente en las instalaciones del laboratorio de investigación RadioGis, donde se llevó a cabo la verificación del funcionamiento del sistema mediante una serie de pruebas experimentales.

Inicialmente, se realizaron las conexiones ilustradas en la figura 46. Este esquema proporciona una descripción superficial de las conexiones físicas necesarias para establecer la comunicación entre la estación meteorológica Davis, la consola y el equipo de cómputo utilizado en la recopilación de datos.

Figura 46.

Esquema de conexiones físicas para la recopilación de datos.



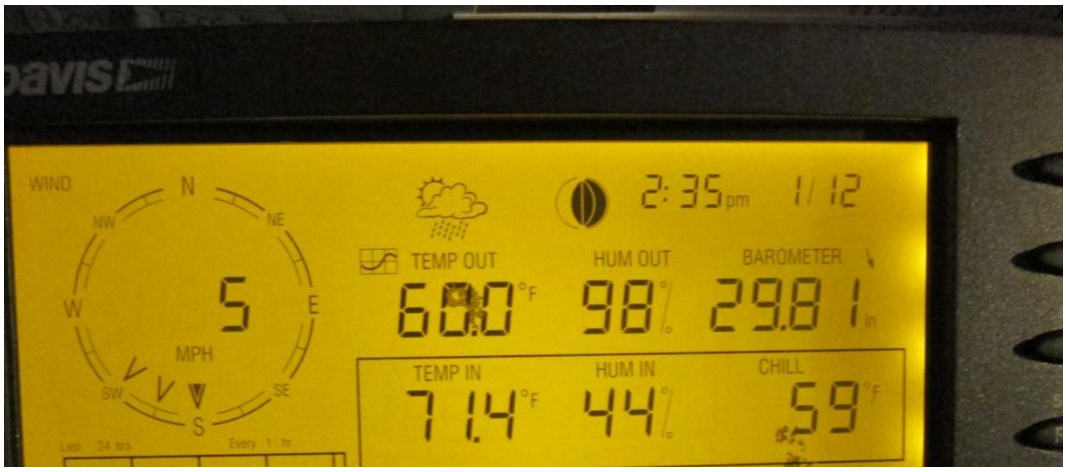
Nota. Esquema de conexiones físicas para la recopilación de datos.

Una vez establecida una conexión física entre cada componente, se llevó a cabo la verificación para asegurar la correcta captura de datos. Inicialmente, se encendió la consola y se

verificó que todos los periféricos (sensor de temperatura, anemómetro, veleta y barómetro) estuvieran registrando datos de manera adecuada en la pantalla LCD de la consola, tal como se muestra en la figura 47. Para iniciar la sincronización de la estación, es necesario presionar el botón "done" de la consola dos veces y mantenerlo presionado hasta que se visualicen las variables de medida.

Figura 47.

Datos registrados en la consola Davis Vantage Pro 2.



Nota. Datos registrados en la consola Davis Vantage Pro 2.

Para verificar que hubo una comunicación entre la consola y el equipo de cómputo se activó el entorno virtual "Estacion_ambiental" en una terminal. Posteriormente, se ejecutó el python “toma_estacion.py”, se digito la cantidad de tomas deseadas y el intervalo de tiempo entre ellas, tal como se indica en el apéndice C “Protocolo de toma de datos”. Los datos recopilados se almacenaron en un archivo "datos.csv" como se observa en la figura 48 dando a entender que la comunicación entre la consola y el equipo fue exitosa.

Figura 48.

CSV resultante de la toma de datos

15731	,,,
15732	2023-12-01,14:35:13,TemOut[Â°F],60.0
15733	,,TempIn[Â°F],71.4
15734	,,WindSpeed[mph],5
15735	,,WinDir[GRADES],268
15736	,,HumIn[%],98
15737	,,HumOut[%],44
15738	,,Barometer[inHg],29.81
15739	,,,

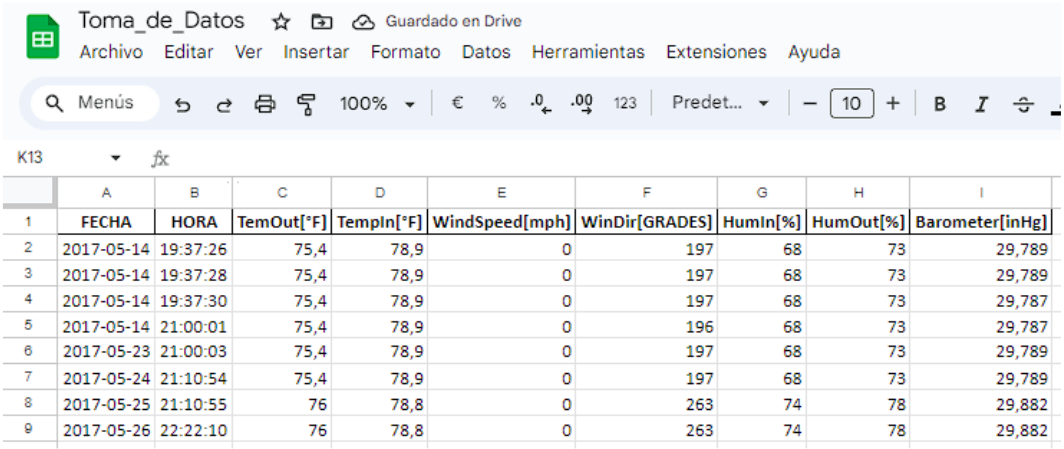
Nota. CSV resultante de la toma de datos

4.2 Validación del sistema IoT

Para comprobar el funcionamiento del sistema IoT, se realizaron varias pruebas durante la instancia final del proyecto. Inicialmente se ejecutaron los archivos Python como se especificó en la sección 3.3.1 y 3.3.2 para la organización, actualización y subida de los datos tomados a la nube mediante la API de Google Sheet de la consola de desarrolladores de Google. Obteniendo como resultado un almacenamiento en la nube como se planteó inicialmente los objetivos del proyecto, esto se evidencia en la figura 49.

Figura 49.

Almacenamiento de los datos en la nube.



	A	B	C	D	E	F	G	H	I
1	FECHA	HORA	TemOut[°F]	TempIn[°F]	WindSpeed[mph]	WinDir[GRADES]	HumIn[%]	HumOut[%]	Barometer[inHg]
2	2017-05-14	19:37:26	75,4	78,9	0	197	68	73	29,789
3	2017-05-14	19:37:28	75,4	78,9	0	197	68	73	29,789
4	2017-05-14	19:37:30	75,4	78,9	0	197	68	73	29,787
5	2017-05-14	21:00:01	75,4	78,9	0	196	68	73	29,787
6	2017-05-23	21:00:03	75,4	78,9	0	197	68	73	29,789
7	2017-05-24	21:10:54	75,4	78,9	0	197	68	73	29,789
8	2017-05-25	21:10:55	76	78,8	0	263	74	78	29,882
9	2017-05-26	22:22:10	76	78,8	0	263	74	78	29,882

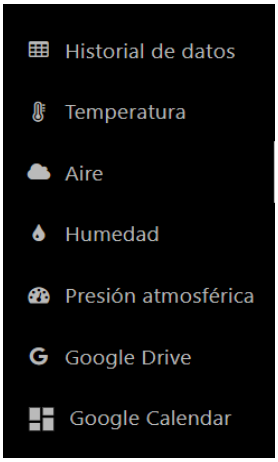
Nota. Almacenamiento de los datos en la nube.

Es importante resaltar que este servicio de almacenamiento en la nube está disponible en cualquier momento sin depender de la ejecución de la máquina proporcionada por AWS. Después de realizar esta prueba, se llevaron a cabo otras pruebas de los distintos servicios propuestos en la interfaz de Node RED. Para acceder a la interfaz como se detalla en el apéndice D “Manual del usuario” fue necesario conocer la IP pública de la máquina virtual.

Una vez en la interfaz se hizo uso del menú lateral que nos permitió acceder a cada uno de los servicios diseñados como se evidencia en la figura 50.

Figura 50.

Menú lateral de la interfaz de Node RED.



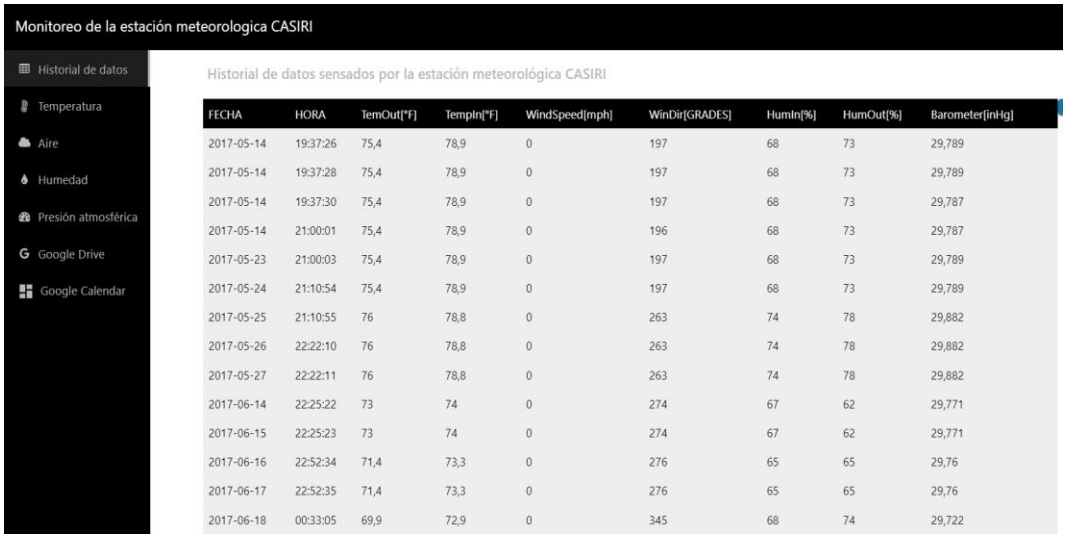
Nota. Menú lateral de la interfaz de Node RED.

4.2.1 *Servicio de consulta de historial general de datos registrados.*

La figura 51 muestra cómo el usuario puede acceder al historial completo de datos registrados por la estación CASIRI. En esta sección, se visualizó los datos registrados en la base de datos de Google Sheets, integrada en el panel de control de Node-RED lo que posibilito la interacción directa entre el usuario y el historial de datos.

Figura 51.

Historial de datos sensados



FECHA	HORA	TemOut[°F]	TempIn[°F]	WindSpeed[mph]	WinDir[GRADES]	HumIn[%]	HumOut[%]	Barometer[inHg]
2017-05-14	19:37:26	75,4	78,9	0	197	68	73	29,789
2017-05-14	19:37:28	75,4	78,9	0	197	68	73	29,789
2017-05-14	19:37:30	75,4	78,9	0	197	68	73	29,787
2017-05-14	21:00:01	75,4	78,9	0	196	68	73	29,787
2017-05-23	21:00:03	75,4	78,9	0	197	68	73	29,789
2017-05-24	21:10:54	75,4	78,9	0	197	68	73	29,789
2017-05-25	21:10:55	76	78,8	0	263	74	78	29,882
2017-05-26	22:22:10	76	78,8	0	263	74	78	29,882
2017-05-27	22:22:11	76	78,8	0	263	74	78	29,882
2017-06-14	22:25:22	73	74	0	274	67	62	29,771
2017-06-15	22:25:23	73	74	0	274	67	62	29,771
2017-06-16	22:52:34	71,4	73,3	0	276	65	65	29,76
2017-06-17	22:52:35	71,4	73,3	0	276	65	65	29,76
2017-06-18	00:33:05	69,9	72,9	0	345	68	74	29,722

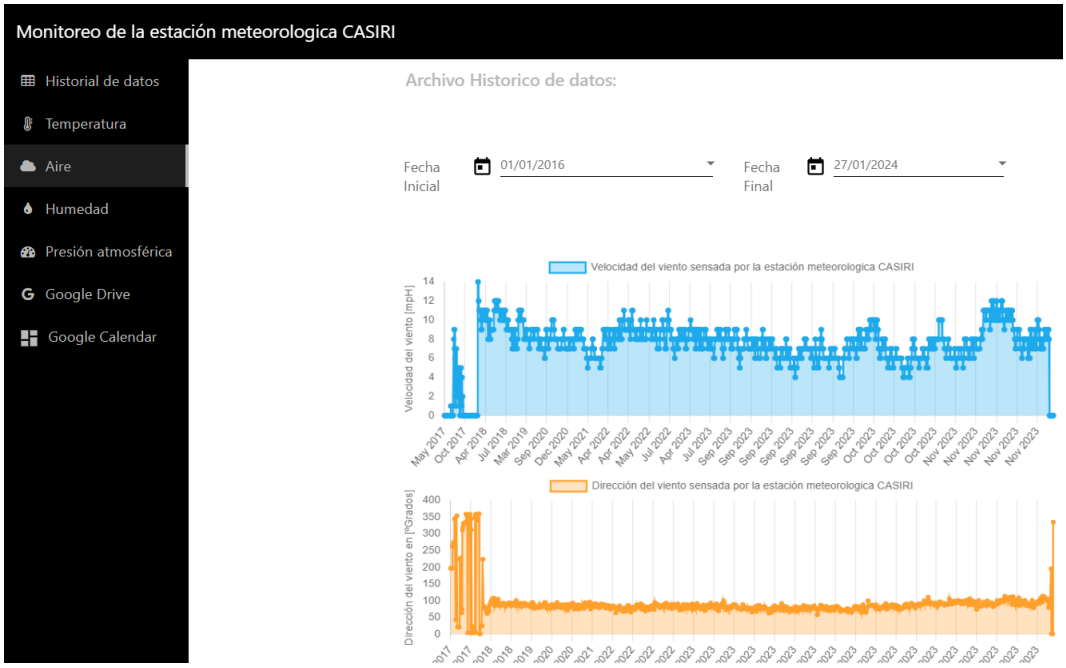
Nota. Historial de datos sensados.

4.2.2 Servicio de consulta en línea del historial de datos y filtrado por fecha.

En cuanto al servicio de consulta en línea del historial de datos con filtrado por fecha, se ha diseño una sección dedicada a cada grupo de las variables registradas (temperatura, humedad, presión y viento), simplificando su visualización. Como se mencionó anteriormente, cada sección cuenta con un gráfico que facilita la selección de un intervalo temporal específico como se evidencio en la figura 52.

Figura 52.

Grafica filtrada por el intervalo de tiempo seleccionado.

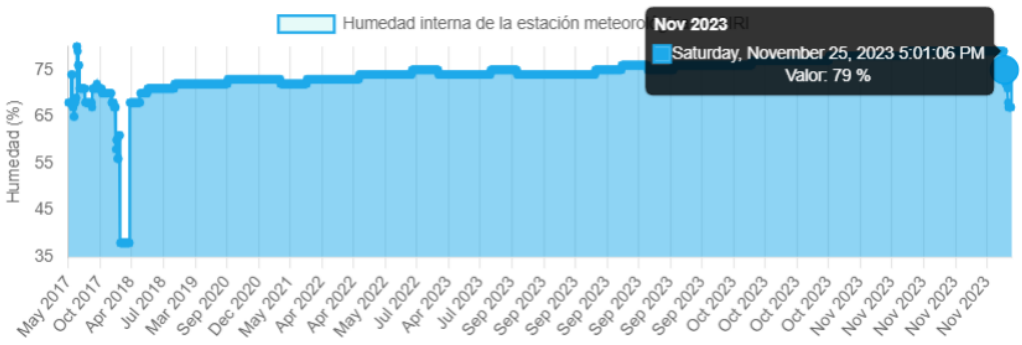


Nota. Grafica filtrada por el intervalo de tiempo seleccionado.

Para tener un control del registro de las muestras filtradas se hizo uso de la herramienta de visualización que tiene la gráfica. Al colocar el cursor sobre cada muestra del grafico permite visualizar el valor de la variable, así como la fecha y hora de su registro como se observa en la figura 53.

Figura 53.

Información detallada de una muestra seleccionada.



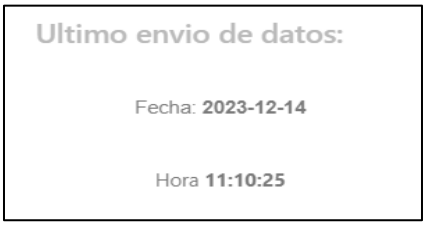
Nota. Información detallada de una muestra seleccionada en el HMI.

4.2.3 Servicio de consulta en línea del último dato

Para la validación del servicio de consulta en línea del último dato registrado, se diseñó un grupo de dashboard en cada sección de las variables, en donde se visualiza la hora de envió del último dato y su respectiva fecha como se observa en la figura 54.

Figura 54.

Información detallada de una muestra seleccionada.



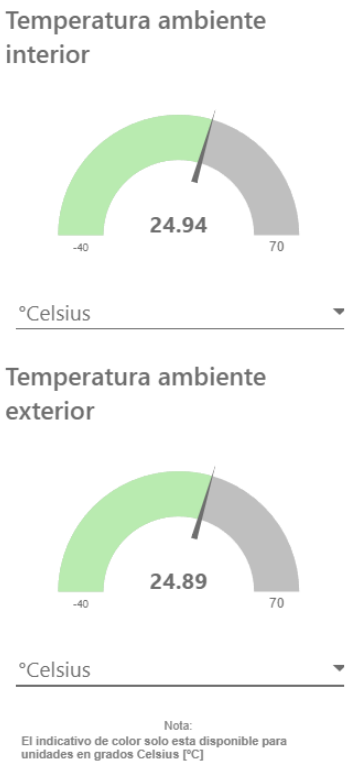
Nota. Fecha y hora del último envío de datos.

Además, se aprovecharon los manómetros integrados en los dashboards de Node-RED para la representación visual de los datos. Con el objetivo de adecuarse a los rangos óptimos de

medición de la estación de radioastronomía CASIRI, se ajustaron los manómetros para variar su tonalidad, adoptando tonos azules en los límites inferiores y una tonalidad roja en los límites superiores. Para mantener la funcionalidad de cambio de unidades que ofrece la consola Davis Vantage Pro 2, se diseñaron selectores de unidades específicos para cada variable medida. La disposición de los manómetros y los selectores de unidades se presenta detalladamente en la figura 55.

Figura 55.

Manómetros y selector de unidades para visualizar la última muestra.



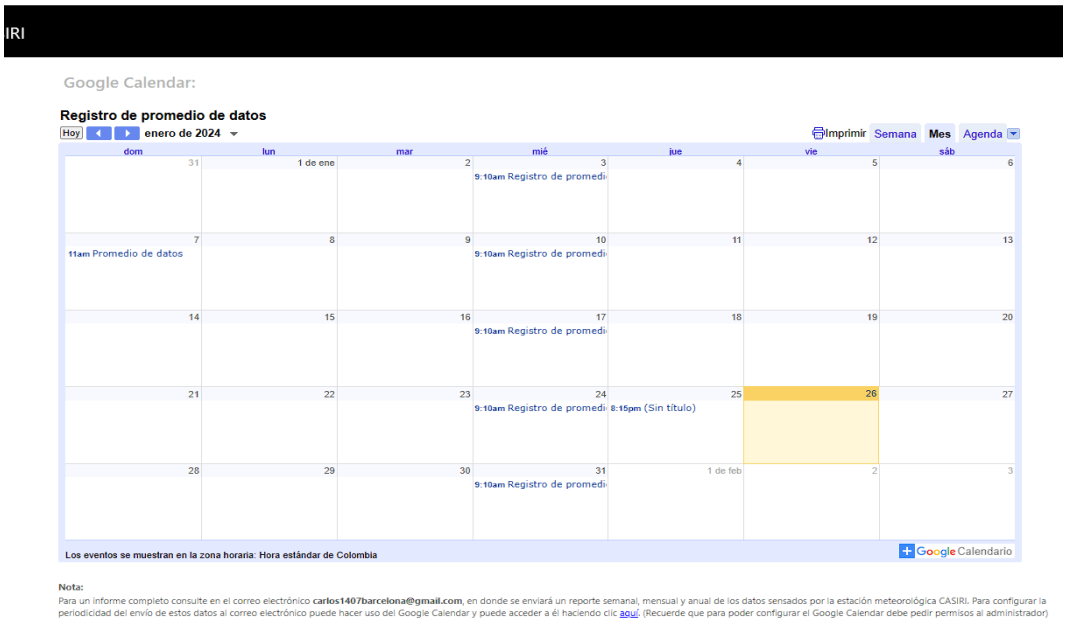
Nota. Manómetros y selector de unidades para visualizar la última muestra.

4.3 Servicio de programación de eventos con Google Calendar.

Para observar la funcionalidad del servicio de programación de eventos con Google Calendar, se procedió a ingresar a través de la sección dedicada en el panel de control, como se detalla en la figura 56. Dentro de esta sección, se visualizaron los eventos programados en el calendario, permitiendo su listado por mes o semana, además de facilitar la impresión del calendario con las fechas relevantes.

Figura 56.

Sección del Google Calendar

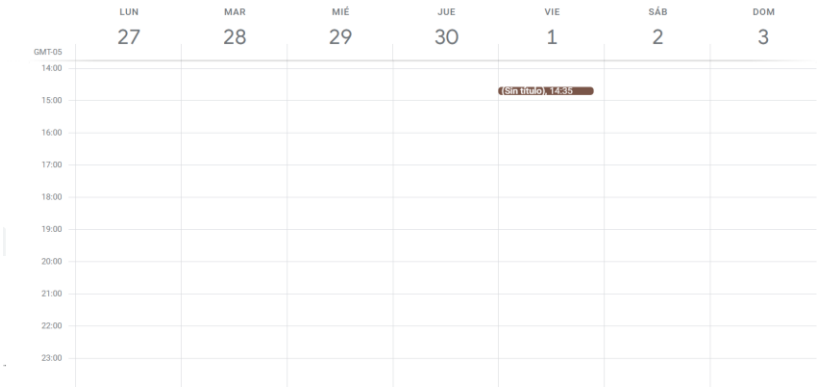


Nota. Sección para programar eventos en el Google Calendar.

Al ingresar a la sección destinada al Google Calendar, se procedió a realizar ediciones haciendo clic en el hipervínculo ubicado en la nota en la parte inferior. Dentro del Calendar, fue esencial configurar los permisos para la edición y asignar un evento cercano a la fecha, tal como se visualiza en la figura 57.

Figura 57.

Calendar designado para la programación de eventos

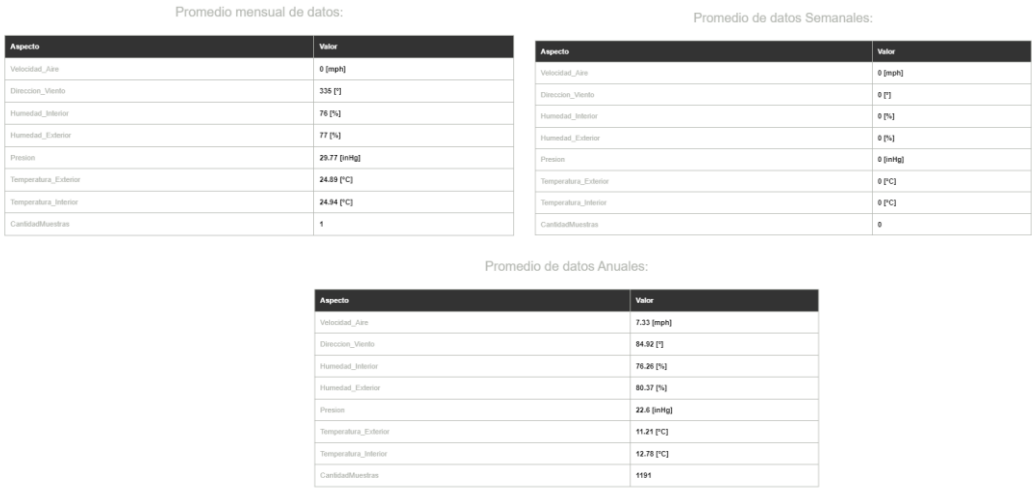


Nota. Calendar designado para la programación de eventos en el HMI.

Tras asignar el evento, se esperó a que ocurriera y se verificó en la bandeja de entrada. En este lugar, se encontraron los datos promedio derivados de las últimas fechas registradas hasta la fecha de la prueba, presentados en forma de promedios semanales, mensuales y anuales, dando así muestra de la interconexión que tiene Node-RED con otras tecnologías en la nube como lo puede ser el servicio de correos electrónicos. El correo electrónico reveló el resultado de la componente HTML, mostrando tablas con los valores de interés, como se ilustra en la figura 58.

Figura 58.

Tablas en el correo electrónico



Promedio mensual de datos:

Aspecto	Valor
Velocidad_Aire	0 [mph]
Direccion_Viento	335 [°]
Humedad_Interior	76 [%]
Humedad_Exterior	77 [%]
Presion	29.77 [inHg]
Temperatura_Exterior	24.89 [°C]
Temperatura_Interior	24.94 [°C]
CantidadMuestras	1

Promedio de datos Semanales:

Aspecto	Valor
Velocidad_Aire	0 [mph]
Direccion_Viento	0 [°]
Humedad_Interior	0 [%]
Humedad_Exterior	0 [%]
Presion	0 [inHg]
Temperatura_Exterior	0 [°C]
Temperatura_Interior	0 [°C]
CantidadMuestras	0

Promedio de datos Anuales:

Aspecto	Valor
Velocidad_Aire	7.33 [mph]
Direccion_Viento	84.92 [°]
Humedad_Interior	76.26 [%]
Humedad_Exterior	80.37 [%]
Presion	22.6 [inHg]
Temperatura_Exterior	11.21 [°C]
Temperatura_Interior	12.78 [°C]
CantidadMuestras	1191

Nota. Tablas adjuntas en el correo electrónico programado con Google Calendar donde se muestran los datos promedios.

4.4 Rendimiento de la máquina.

Al evaluar el rendimiento de la máquina AWS durante la ejecución del proyecto, es posible realizar un análisis del comportamiento de la misma en el desarrollo del diagrama en Node-RED. Durante esta etapa, se lleva a cabo la transmisión de datos registrados por la estación, los cuales son cargados en la interfaz de Node-RED. La figura 59 ilustra que la máquina mantiene un rendimiento óptimo, lo que permite al usuario o desarrollador ejecutar todo el sistema desarrollado, reduciendo así el consumo o la necesidad de recursos de la máquina.

Figura 59.

Utilización de la CPU (%) de la maquina AWS.

Nota. Grafica de la utilización de la CPU (%) de la maquina AWS en las mediciones que proporciona la plataforma.

5 Conclusiones

El proyecto ha logrado con éxito mejorar la estación meteorológica CASIRI mediante la implementación de Node-RED, consolidándola como una solución avanzada para la visualización y procesamiento en la nube de datos atmosféricos. La adopción de Node-RED ha permitido una transición eficiente hacia un entorno de desarrollo visual y nodal, brindando flexibilidad y escalabilidad a la estación.

La estrategia de desarrollo, enfocada inicialmente en la implementación remota, ha demostrado ser acertada al establecer una base sólida para futuras expansiones. La validación del sistema en su totalidad, desde la adquisición de datos hasta la visualización remota a través de la interfaz HMI, ha respaldado la viabilidad y eficacia de la solución propuesta.

La interfaz HMI diseñada con Node-RED ha mejorado significativamente la accesibilidad y comprensión de los datos meteorológicos. La implementación de funciones como la consulta en

línea del historial de datos, la visualización en tiempo real y la programación de eventos mediante Google Calendar ha ampliado las capacidades y utilidad de la estación para los usuarios.

El rendimiento del sistema en la nube, respaldado por servicios de Amazon Web Services (AWS), ha demostrado ser óptimo, proporcionando una plataforma robusta para el procesamiento y almacenamiento eficiente de datos. Este proyecto ha consolidado a la Universidad Industrial de Santander (UIS) como líder en el campo de IoT, fortaleciendo la investigación y desarrollo tecnológico en el ámbito de la radioastronomía.

Referencias bibliográficas.

E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, “Industrial Internet of Things: Challenges, Opportunities, and Directions,” IEEE Transactions on Industrial Informatics, vol. 14, no. 11, pp. 4724–4734, Nov. 2018, doi:10.1109/TII.2018.2852491.

Dario, J. (2016, 27 abril). Programación visual con Node-Red: conectando el Internet de las cosas con facilidad. Toptal Engineering Blog. https://www.toptal.com/nodejs/programacion-visual-con-node-red-conectando-el-internet-de-las-cosas-con-facilidad#:~:text=Node%2DRED%20es%20un%20editor,que%20se%20comuniquen%20entre%20ellos.

BIBLIOTECA VIRTUAL UIS. (s. f.). https://ieeexplore-ieee-org.bibliotecavirtual.uis.edu.co/document/10315514

The core nodes: Node-RED. (s. f.). https://nodered.org/docs/user-guide/nodes

Node-red-dashboard. (s. f.). https://flows.nodered.org/node/node-red-dashboard

Portal, T. (2023, 7 diciembre). Servidores. TIC Portal. https://www.ticportal.es/glosario-tic/servidores

What is Python? Executive Summary. (s. f.). Python.org. https://www.python.org/doc/essays/blurb/

¿Qué es una API? - Explicación de interfaz de programación de aplicaciones - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/api/

Smith, J., & Jones, A. (2008). "Human-Machine Interface Design." En Handbook of Human Factors and Ergonomics (pp. 1433-1462). Wiley.

BIBLIOTECA VIRTUAL UIS. (s. f.-b). https://ieeexplore-ieee-org.bibliotecavirtual.uis.edu.co/document/9928973 (Nota: Figura HMI)

Infante, D. C. H., & Infante, D. C. H. (2023, 19 abril). Qué es Node.Js: casos de uso comunes y cómo instalarlo. Tutoriales Hostinger. https://www.hostinger.co/tutoriales/que-es-node-js#:~:text=Conclusi%C3%B3n-,Node.,de%20una%20solicitud%20con%20Node.

What-is-AWs. (s. f.). [Vídeo]. Amazon Web Services, Inc. https://aws.amazon.com/es/what-is-aws/#:~:text=Amazon%20Web%20Services%20(AWS)%20es,de%20datos%20a%20nivel%20global.

Ramírez, I. (2020, 31 enero). Máquinas virtuales: qué son, cómo funcionan y cómo utilizarlas. Xataka. https://www.xataka.com/especiales/maquinas-virtuales-que-son-como-funcionan-y-como-utilizarlas

Qué es una dirección IP: definición y explicación. (2023, 19 diciembre). latam.kaspersky.com. https://latam.kaspersky.com/resource-center/definitions/what-is-an-ip-address

De Zúñiga, F. G. (2023, 31 octubre). Servicios web: qué son y qué tecnología usar en su desarrollo. Blog de arsys.es. https://www.arsys.es/blog/web-services-desarrollo#:~:text=Un%20servicio%20web%20(o%20web,entre%20m%C3%A1quinas%20conectadas%20en%20Red.

Smart City: significado, características y ejemplos | Enel X. (s. f.). Enel X. https://corporate.enelx.com/es/question-and-answers/what-is-a-smart-city#:~:text=Una%20smart%20city%20es%20una,Redes%20inteligentes%20de%20transporte%20urbano

¿Qué es IoT? - Explicación del Internet de las cosas - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/iot/#:~:text=El%20t%C3%A9rmino%20IoT%2C%20o%20Internet,como%20entre%20los%20propios%20dispositivos.

¿Qué significa HMI? Interfaz humano-máquina | COPA-DATA. (2023, 12 diciembre). https://www.copadata.com/es/productos/zenon-software-platform/visualizacion-control/que-significa-hmi-interfaz-humano-maquina-copa-data/#:~:text=HMI%20son%20las%20siglas%20de,para%20las%20de%20entornos%20industriales.

Acerca de JavaScript | MDN. (s. f.). MDN Web Docs. https://developer.mozilla.org/es/docs/conflicting/Web/JavaScript

Coppola, M. (2023, 12 abril). Frontend y backend: qué son, en qué se diferencian y ejemplos. Hubspot. https://blog.hubspot.es/website/frontend-y-backend

Flores, F. (2023, 18 abril). Qué es Serverless, ventajas y servicios. OpenWebinars.net. https://openwebinars.net/blog/que-es-serverless-ventajas-y-servicios/

Beservices. (2023, 28 noviembre). ¿Qué es Google Apps Script y para qué sirve? beservices. https://blog.beservices.es/blog/que-es-google-apps-script-para-que-sirve

Apéndices

Apéndice A. Manual de instalación y configuración de la máquina virtual proporcionada por AWS

Se elaboró este apéndice con la finalidad de ofrecer al usuario una guía lógica para la instalación y configuración de cualquier máquina virtual proporcionada por AWS. En este contexto, se contempla la idea de trasladar la interfaz HMI del proyecto a esta máquina virtual y se entra a detalle de las configuraciones previas para un correcto funcionamiento de la interfaz.

Apéndice B. Generalidades de Node RED.

El propósito de esta guía fue proporcionar a los usuarios una comprensión detallada de las generalidades de Node-RED, desde la instalación inicial hasta la personalización avanzada de los dashboards que podían ser creados con esta plataforma. Se exploraron conceptos fundamentales en Node-RED, incluyendo el transporte de datos, y se enfocó en los nodos más relevantes para el procesamiento de datos y su posterior visualización en los dashboards. Esta guía sirvió como una sólida base para comprender conceptos esenciales en la utilización de esta herramienta de programación visual basada en flujos, ofreciendo a los usuarios las habilidades necesarias para aprovechar al máximo las capacidades de Node-RED.

Apéndice C. Protocolo Toma de Datos.

Este documento presentó el protocolo de toma de datos seguido durante la "VIII EXPEDICIÓN CIENTÍFICA DE COLOMBIA A LA ANTÁRTICA VERANO AUSTRAL 2021– 2022", con el objetivo de proporcionar una guía para llevar a cabo las mediciones en el campo y lograr una toma de datos correcta y eficiente. Se incluyeron recomendaciones para la revisión de equipos antes de iniciar las mediciones y una descripción del software utilizado para

la toma de datos. Finalmente, se proporcionó el protocolo con las instrucciones para que los asistentes al evento llevaran a cabo la toma de datos.

Apéndice D. Manual de usuario.

La finalidad de esta guía es orientar a los usuarios a lo largo de una ruta lógica para el uso efectivo de la HMI (Interfaz Persona-Máquina) diseñada con el propósito de supervisar la estación de radioastronomía CASIRI. Se abordarán aspectos detallados que van desde la instalación de la copia de seguridad hasta la utilización de cada uno de los servicios que esta interfaz proporciona.

Universidad industrial de Santander

Instalación y configuración de la maquina virtual proporcionada por AWS.

Enero de 2024

Carlos Eduardo Silva Sepúlveda

Brayan Hernando Gonzales Mendoza

Tabla de contenido

Introducción

En la era de la computación en la nube, la plataforma Amazon Web Services (AWS) se ha destacado como un líder indiscutible, ofreciendo una amplia gama de servicios para satisfacer las necesidades de empresas y desarrolladores. En este contexto, la implementación de una máquina virtual en AWS se convierte en un paso esencial para desplegar y ejecutar diversas aplicaciones y servicios de manera eficiente.

Una herramienta valiosa que puede potenciar la funcionalidad y la automatización en tu máquina AWS es Node-RED. Node-RED es un entorno de programación basado en nodos de código abierto que facilita la conexión de dispositivos, APIs y servicios en un flujo de trabajo visual. Este entorno intuitivo permite a los usuarios diseñar flujos de trabajo fácilmente, lo que resulta especialmente útil para la creación rápida de aplicaciones IoT (Internet de las cosas) y la automatización de tareas.

Conceptos previos

VM: Significa Virtual Machine o máquina virtual. En general, una máquina virtual se refiere a una réplica virtualizada de un entorno de hardware de computadora. Esto incluye recursos como CPU, memoria, almacenamiento y red. Una VM se utiliza para ejecutar sistemas operativos y aplicaciones, y puede ser equivalente a una computadora física en muchos aspectos. Las plataformas en la nube, como Google Cloud, AWS, Azure, Alibaba usan ese concepto para referirse a un servicio que ofrecen para que tu cuentes con algo que hace lo mismo que un computador, pero está en la nube de alguna de esas plataformas.

Instancia: Este término es conocido en la programación orientada a objetos, donde cada objeto es una instancia de una clase. Pero aquí tiene una connotación un poco diferente porque hablar de una instancia en AWS es hablar de una máquina virtual que se ha creado a imagen de otra que ofrece la plataforma. Imagina que el profesor te da la guía de un taller, tu le sacas una copia y luego trabajas sobre esa copia. Eso significa que tú has instanciado la guía del taller y ahora trabajas en una instancia del mismo. A diferencia de las VM tradicionales, las instancias son flexibles en términos de poder reconfigurar sus componentes. En adelante, en este trabajo, cuando usemos el término instancia o VM nos referiremos a lo mismo.

EC2: EC2 significa Elastic Cloud. Pero en realidad es un tipo de máquinas virtuales que ofrece AWS que permiten ser adaptadas aún después de creadas para contener mayores o menores recursos como CPU, memoria, almacenamiento y rendimiento de red, sistemas operativos, reglas de seguridad, el tipo de IP elástica o estática, para satisfacer tus necesidades específicas. Eso las hace flexibles y escalables. Es en el fondo una idea de negocios, donde la persona paga justo por lo que usa, ni más ni menos. EC2 es uno de los servicios más populares de AWS y es ampliamente utilizado para alojar aplicaciones web, ejecutar cargas de trabajo de análisis de datos, ejecutar servidores de aplicaciones, alojar sitios web y más. AWS también cuenta con una amplia gama de EC2 preconfiguradas para ser instanciadas para diversos casos de uso, como cómputo general, cómputo de alto rendimiento, cómputo optimizado para memoria, y más.

EBS: Significa Elastic Block Store. Es un tipo de almacenamiento que se le puede dar a una EC2 que es flexible.

AMI: "Amazon Machine Image" (Imagen de Máquina Amazon). Una AMI es una plantilla preconfigurada que contiene la información necesaria para lanzar una instancia de EC2 (Elastic Compute Cloud), que es una máquina virtual en la nube de AWS. Las amis son fundamentales en AWS, ya que te permiten crear instancias de EC2 de manera eficiente y reproducible.

IP estática: Una máquina tiene una IP estática si cuando se reinicia mantiene siempre la misma IP.

IP dinámica: Una máquina tiene una IP dinámica cuando se reinicia cambia la misma IP.

IP elástica: AWS es fanático del término “elastic”, quieren proyectar que todo es elástico. Por eso, cuando hablan de IP elástico se refieren en realidad a una IP estática, pero que el usuario puede usar a su antojo, es decir, la puede usar en una VM o trasladarla a otras VM o a recursos en la nube, como instancias EC2 o balanceadores de carga. La principal ventaja de una IP elástica en AWS es que se mantiene constante incluso cuando detienes y luego reinicias la instancia a la que está asociada. Esto permite una mayor flexibilidad y facilita la alta disponibilidad y la recuperación ante desastres. En conclusión, AWS no ofrece un servicio que se llame “IP estática”, eso lo suple con un servicio de IP estática. Desde ese punto de vista “IP estática” no es un tipo de IP, sino un servicio AWS.

Tarea 1: Inscripción a AWS

Proceso a seguir
Https://aws.amazon.com/ > registra tus datos

Tarea 2: Crear una VM (instancia) de tipo EC2

VM es la sigla que usaremos para referirnos a una máquina virtual, El término que usa AWS para referirse a lo mismo es “instancia” que significa que estamos instanciando un modelo de máquina que ya tiene AWS.

Paso 1. Entra a la consola. Es decir, inicia sesión en AWS
Https://aws.amazon.com/ > Sign in to the Console >
Paso 2. Creación de la VM
Lunch a virtual machine With EC2 > Name and tags (das un nombre que permita diferenciar esta VM de otras)> Entre las imágenes disponibles selecciona “Ubuntu”> **OJO:** No oprima “Launch Instance” hasta que no se le indique.

Paso 3. Creación de par de claves y lanzamiento (osea que la VM quede corriendo).

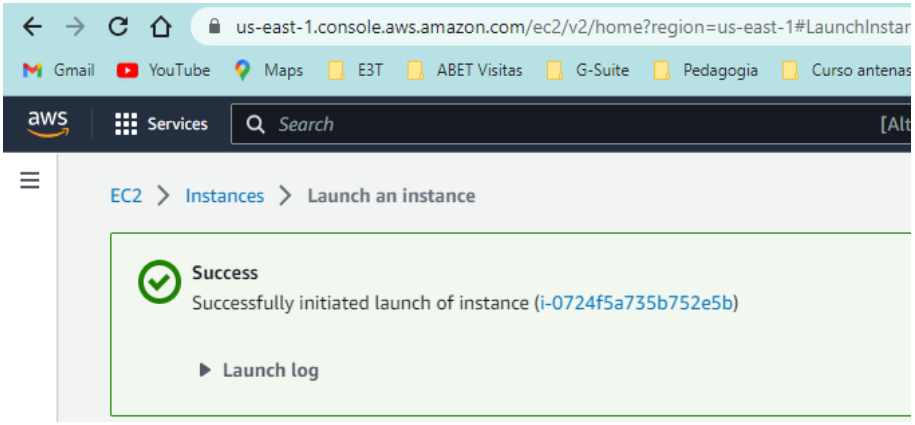
Algunos considerandos previos:

- Esta parte es muy necesaria en AWS. Está pensada en la seguridad, para que personas no autorizadas no puedan acceder a una VM

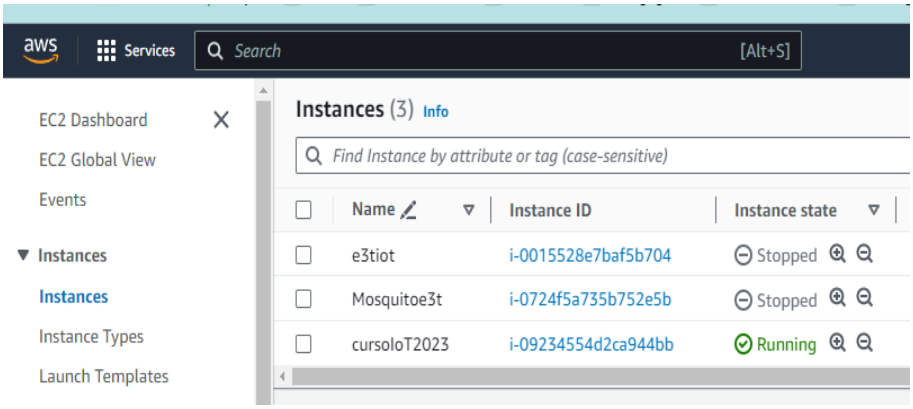
- Esas claves son necesarias cuando se intente acceder a la VM desde terminales remotos usando aplicaciones como powershell, putty o VNC. En nuestra experiencia no usaremos esas aplicaciones, sino que aprovecharemos una terminal en línea que ofrece AWS. De esta manera evitaremos usar estas claves, pero las vamos a cuidar por si se requieren a futuro.

- Son dos claves, la pública queda en la VM y se puede compartir con otros usuarios o sistemas, se usa para cifrar datos que se envían a la VM. La privada baja al computador del usuario en forma de un archivo .pem. También es posible bajar como un archivo .ppk. **Pero es más recomendable el archivo en formato .pem**. Sirve a la hora de configurar una terminal de acceso remoto a la VM, debe mantenerse segura y se utiliza para descifrar los datos que se reciben en la instancia EC2.

- Una recomendación es que le des a ese archivo el mismo nombre de la VM, ya que en el futuro puedes tener varias VM y por consiguiente varios de estos archivos

Paso 3. Crea el par de claves
Continuando desde donde estás creando la instancia > haz click en “Create new key pair” > cuando te pida un nombre para el par de claves dale el mismo dado a la máquina en el paso anterior> Oprime “Create key par” > Revisa el disco de tu computadora, debe haber bajado un archivo con nombre igual al tag de la VM y con extensión .pem > Guarda ese archivo en una carpeta selecta para que no se te pierda.
Paso 4. Finaliza la creación de la instancia (VM)
Oprime “Launch instance”. Verás algo así como resultado: 
Revisa el estado de tu máquina

Menú lateral izquierdo > Instances > Instances. El siguiente es un ejemplo de lo que verás si la máquina creada se llamara cursoiot2023. Nota: si no vez tu VM refresca la vista del navegador con



Vemos que la máquina queda encendida por defecto “Running”

Prueba apagar y encender la máquina. NOTA: es importante apagarla cuando no la uses si no deseas consumir recursos.

Chequeas la instancia de interés > Menu horizontal > Instance state > “Stop instance” para apagarla o “Start Instance” para encenderla

Tarea 3. Configuración de la VM para soportar clientes externos iot.

Las VM en la nube vienen usualmente acompañadas con un conjunto de reglas de seguridad que permiten que el dueño de la máquina pueda prevenir el uso indebido. Básicamente, lo que se busca es que estén controlados los puertos en la VM. Se trata de:

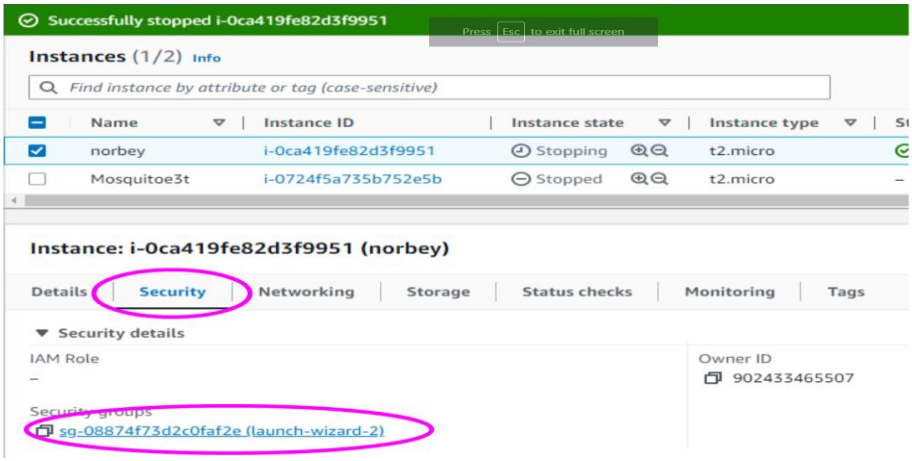
- Reglas de entrada (Inbound rules): sirven para permitir o denegar el tráfico entrante a través de ciertos puertos de red. Ellas controlan el acceso a la VM desde fuentes externas, bien sea desde otras VM, servidores web, computadores de usuarios, dispositivos iot, etc.
- Reglas de salida (Outbound rules): es similar a las reglas de entrada, pero para el control del tráfico de salida.
- La configuración por defecto de esa regla no es suficiente para permitir la entrada de los datos que envían dispositivos iot.

Paso 1. Revisar las reglas de seguridad de tu instancia o VM

Si te saliste del panel que lista de las instancias creadas, vuelve all. Si no sabes como, puede ser así:
Https://aws.amazon.com/ > Sign in > EC2 > Menu lateral > Instances > Instances

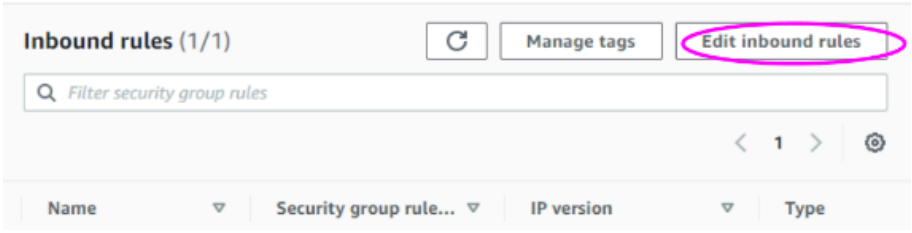
Paso 2. La configuración en sí de las reglas de seguridad

Chequeas la instancia de interés > **asegurate que la VM esté apagada** > Security > busca "Security Groups" > haz click en el enlace que aparece debajo de "Security groups"

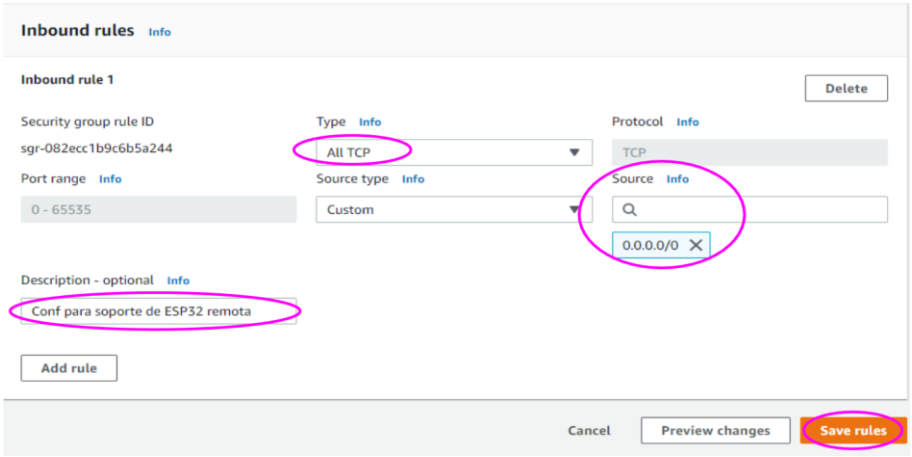


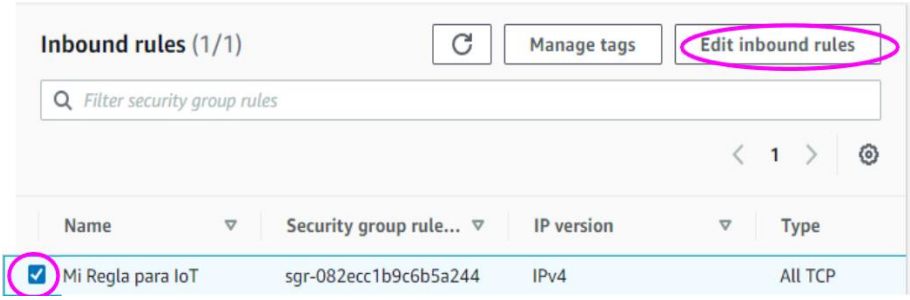
Editar la regla de seguridad de entradas y dejarla activa

En la nueva ventana selecciona la regla vigente > "Edit inbound rules"



Edita la configuración con los datos de la siguiente figura:

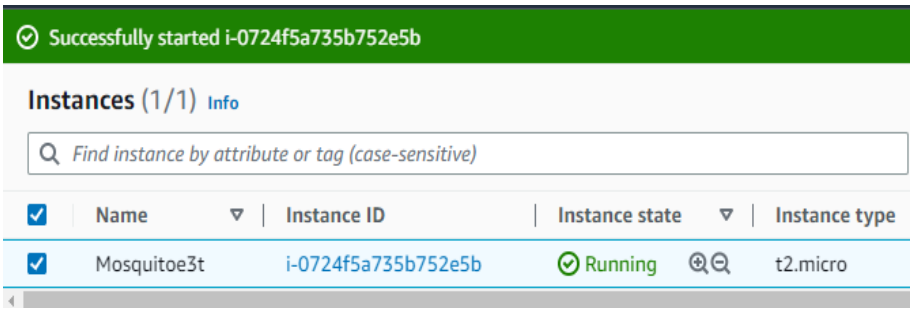


Observa la advertencia que aparece y dice: “Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.” Es algo que debemos recordar para tomar futuras acciones en pro de la seguridad. Por ahora está bien porque nos interesa que las comunicaciones fluyan desde cualquier IP de manera que una tarjeta ESP-32 no tenga problemas para enviarle datos a esta VM.
Oprime “Save rules”
Deja la regla activa así cuando desees acciones adicionales 
Puedes pasar a tu Instancia así: menú lateral izquierdo > Instances > Instances

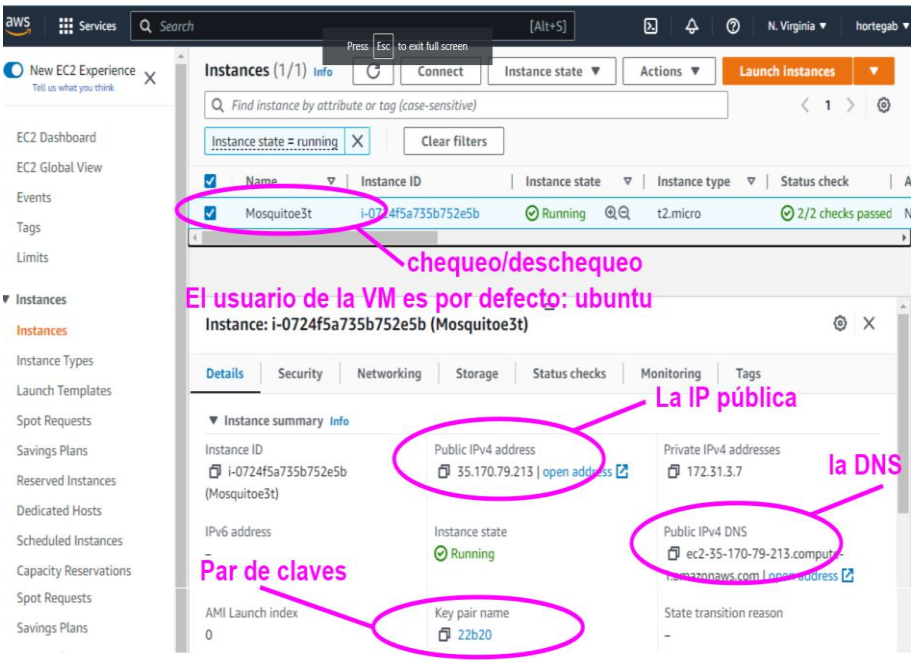
Tarea 4. El dia a dia para encender la máquina y contar con una terminal de trabajo

Paso 1. Encendemos la VM así:

Te logeas con tu cuenta AWS y quedas en la consola de gestión> entras a “EC2” > Menu lateral > Instances, Instances > chequeas la instancia de interés > Menu horizontal > Instance state > start instance > constatamos que la VM quede corriendo (puede tomar hasta un minuto o más. A veces la instancia ya está corriendo, pero la página web de AWS se demora en refrescar la información. Puedes refrescar la página manualmente con). Deberá quedar corriendo como en la Fig siguiente



Es importante identificar datos de relevancia de la VM, como se muestra en la siguiente figura



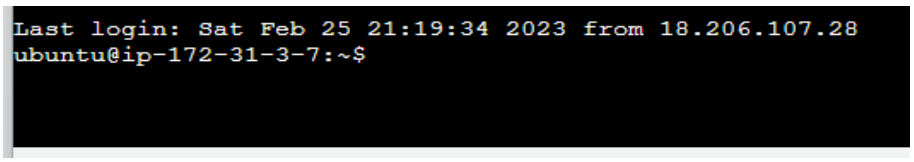
Paso 2. Abrimos una terminal

La opción de usar una terminal en línea (la recomendada)

- Es la recomendada porque no requiere instalar nada en un computador ni usar keys. Simplemente, en un navegador aparece el recuadro negro, osea el equivalente a una terminal, para escribir los comandos.

  La secuencia es:

- Menu horizontal > Connect > Connect
- Se abrirá una nueva pestaña en el navegador con una terminal de comandos en línea



La opción de usar una terminal putty

- Se puede seguir al pie de la letra el video de este enlace, excepto que en el minuto 6:45 dice que el usuario es ec2-user, pero hay que colocar allí el usuario de nuestra máquina, que últimamente se llama por defecto “Ubuntu”.

Tarea 5. Instalación de Node.js

Node.js representa una revolución en el desarrollo de aplicaciones web y servicios del lado del servidor, destacándose por su enfoque asincrónico basado en javascript. Su eficiencia en la gestión de múltiples solicitudes concurrentes lo posiciona como la elección ideal para aplicaciones en tiempo real y servicios que requieren actualizaciones continuas, como chats y juegos en línea.

Para aquellos familiarizados con Google Apps Script, Node.js comparte similitudes valiosas. Ambas plataformas utilizan javascript, permiten el desarrollo de servicios web, apis, y webhooks, y facilitan el cómputo en la nube, incluyendo el desarrollo tanto del backend como del frontend. Sin embargo, la distinción principal radica en que Node.js se instala en un servidor propio, otorgando un control más profundo sobre la infraestructura.

Cuando exploramos alternativas, Python, con su capacidad para crear servicios restful, se presenta como un competidor significativo. Mientras Node.js se destaca por su eficiencia y capacidad para manejar eventos en tiempo real, Python brilla en el desarrollo restful, ofreciendo un marco robusto y ampliamente utilizado llamado Flask. Este marco proporciona herramientas poderosas para la construcción de apis restful y servicios web eficientes.

En la elección entre Node.js y Python, se trata de sopesar las fortalezas específicas de cada tecnología en función de los requisitos particulares del proyecto. Ambas ofrecen soluciones sólidas para el desarrollo web y del lado del servidor, permitiendo a los desarrolladores elegir la herramienta que mejor se adapte a sus necesidades y preferencias.

Las siguientes son las instrucciones tomadas del archivo Readme que aparece en el repositorio de github en el año 2023, semestre 2 (2023-2). Este proceso también instala nmp que es un gestor de paquetes para node. Es decir para realizar instalaciones adicionales a Node se usará nmp. Hacemos esta aclaración porque las instrucciones pueden llegar a cambiar con el tiempo y muchas veces es importante ir a la fuente, sobre todo porque las versiones van cambiando, En el 2023-2, todo indica que la versión más madura y más apropiada para nuestra versión de Ubuntu es la 20.09, aunque ya existe la versión 21x.

Paso 1. Bajar el código fuente y la clave de seguridad Entrega todos los siguientes comandos de un solo golpe
Sudo apt-get update && sudo apt-get install -y ca-certificates curl gnupg && Sudo mkdir -p /etc/apt/keyrings && curl -fssl https://deb.nodesource.com/gpgkey/nodesource-repo.gpg.key | sudo gpg --dearmor -o /etc/apt/keyrings/nodesource.gpg
Paso 2. Crear el repositorio deb

Nota: con “NODE_MAJOR=20” estarás indicando que deseas instalar la versión 20.x. Otras opciones son NODE_MAJOR=16 NODE_MAJOR=18 NODE_MAJOR=20 NODE_MAJOR=21
NODE_MAJOR=20 && echo "deb [signed-by=/etc/apt/keyrings/nodesource.gpg] https://deb.nodesource.com/node_$NODE_MAJOR.x nodistro main" | sudo tee /etc/apt/sources.list.d/nodesource.list
Paso 3. Correr las actualizaciones e instalar Node.js
Sudo apt-get update && sudo apt-get install nodejs -y

Tarea 6. Instalación de Node Red

Node Red es una herramienta que se puede instalar en cualquier computadora, pero el verdadero sentido está en instalarlo en un servidor como es el caso de la nuestra VM en AWS. De manera similar a cuando Moodle se instala en un servidor, es posible invocar Node Red desde cualquier computadora como una página web. Desde otro ángulo, Node-RED también es comparable, en la forma en que se crean soluciones, con GNU radio en el sentido que ambos usan la programación visual, mediante flujogramas. Pero en Node-RED es más común el término nodo para referirse no sólo a los elementos que se usan sino también a un flujograma, que surge como producto de unir varios nodos para una solución. En este sentido hay una similitud con la programación orientada a objetos, donde varios objetos se pueden unir para componer un objeto mayor. Pero la concepción detrás de Node-RED va más allá de todo lo existente porque está pensada más bien para soluciones en la nube a nivel de servidor.

Más allá de todo eso, con Node-RED es posible crear, en línea, interfaces para visualización gráfica en línea. En este sentido es una competencia para otras herramientas que se usan en la UIS llamada Ignition con MQTT Engine, pero que tiene un alto costo para su uso.

Node-RED. Podemos decir que Node-RED busca llegar a ser el equivalente gráfico de Node.js, lo cual es mucho decir si recordamos que Node.js está entre las herramientas más competitivas para los desarrollos en la nube.

Recomendaciones iniciales:

- Visita la página de Node-RED para conocer novedades: https://nodered.org/

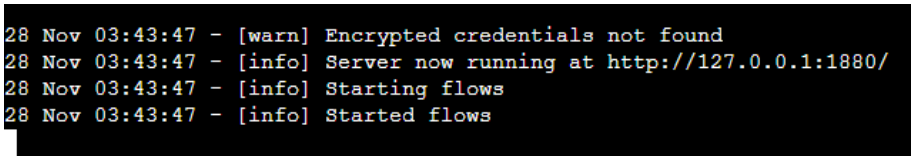
- Vemos que Node-RED tiene muchas opciones de uso, desde la instalación local, en PC, en una raspberry, en android y en la nube. Nos interesa esta última y que sea en AWS, de manera que, en la página https://nodered.org/ nos interesa explorar más bien el enlace: Amazon Web Services

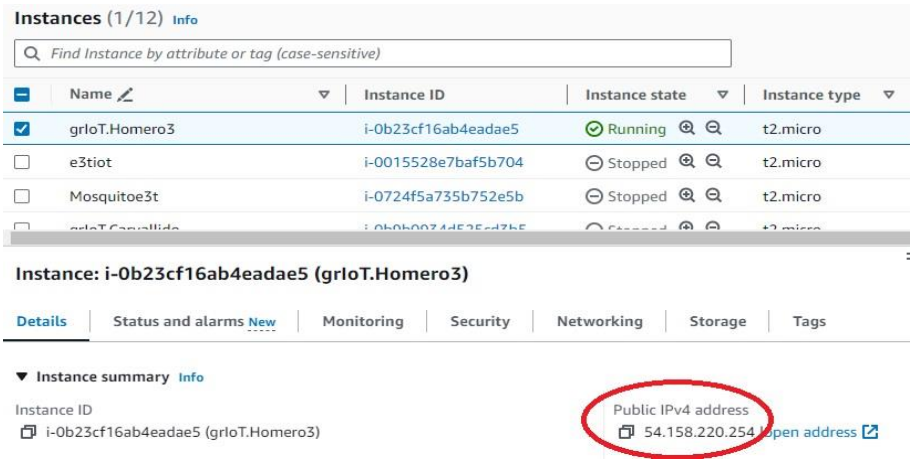
- El enlace anterior es el recomendado para conocer las últimas novedades en la instalación. Los comandos de la siguiente tabla es una experiencia de instalación vivida en febrero 2023, siguiendo este video

Sudo npm install -g --unsafe-perm node-red	Usando nmp hacemos la instalación de Node-RED
Sudo systemctl daemon-reload	Recarga la configuración del demonio del sistema para asegurarse de que los cambios realizados en la configuración o unidad systemd, hechos por la instalación de Node-RED, se apliquen correctamente.
Sudo apt-get upgrade	
Node-red -v	Lanzamiento de Node Red. Es como cuando uno abre el editor de python o gnuradio. Entonces la consola se queda en espera. No se debe cerrar la consola porque se cerrará node red.

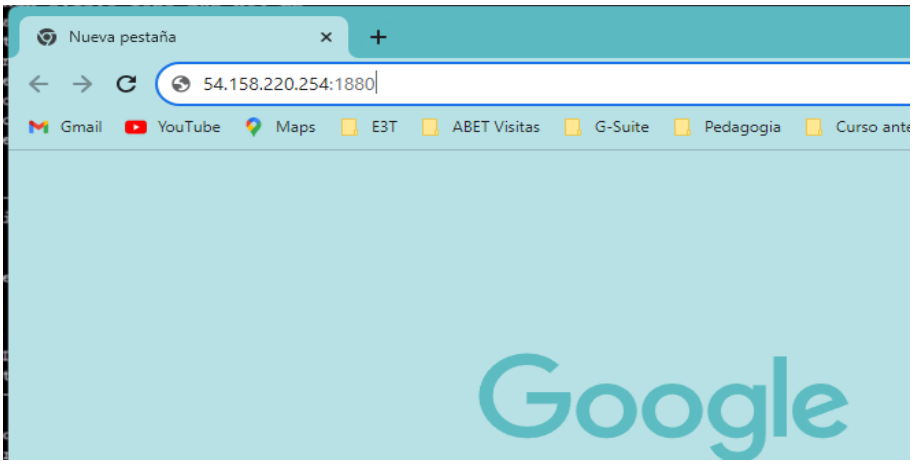
Tarea 7. Pruebas de reconocimiento a Node Red

Pruebas en la consola de node-RED

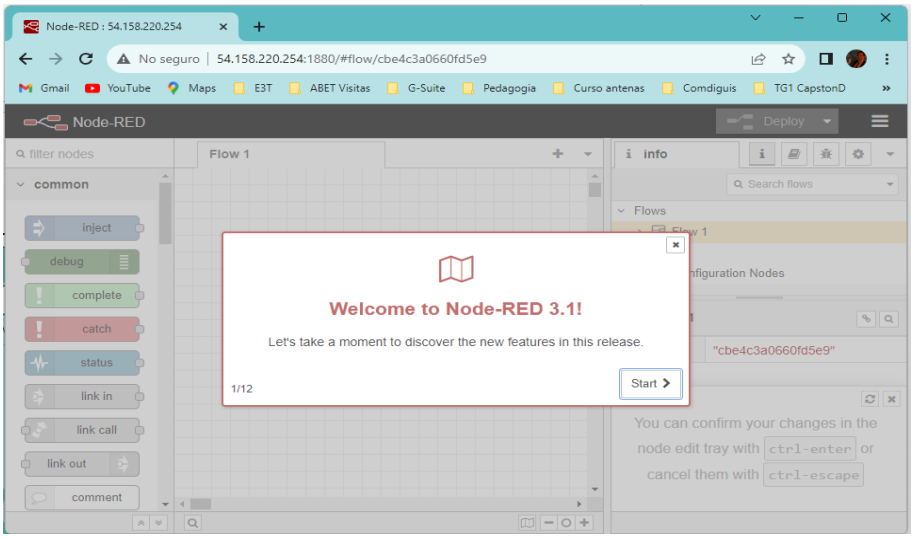
Abre node-RED con lo cual entras a la consola. Para ello usa el comando:
Node-red -v
Como resultado verás un prompt que te indica que ya no estás en Ubuntu sino en Node-RED

Ve a AWS y revisa cual es la IP pública de tu instancia



Ve a un navegador y escribe la ip seguida de ":" y el puerto de este servicio que es 1880



Como resultado estarás en la interfaz de programación visual de Node-RED



¿Comprueba que pasa si en tu instancia sales de la consola Node-RED con Ctrl+C. Puedes seguir viendo tu interfaz gráfica en el navegador como se hizo en el punto anterior? Debe refrescar el navegador para conocer qué pasa. Verás que ya no podrás trabajar en la interfaz gráfica, porque se ha cerrado Node-RED

En Ubuntu, comprueba si hay una instancia abierta de Node-RED, con el siguiente comando:

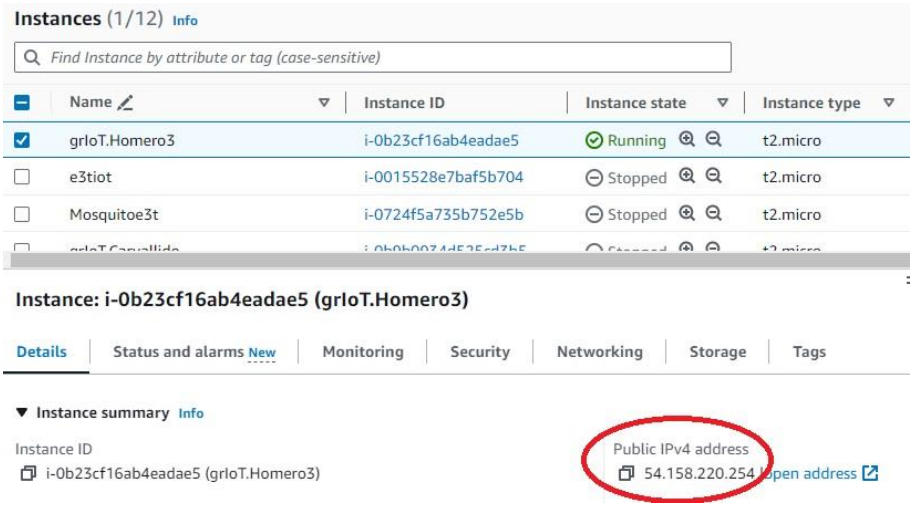
Ps aux | grep node-red

La siguiente respuesta indica que hay una instancia de grep, pero no de node-RED:

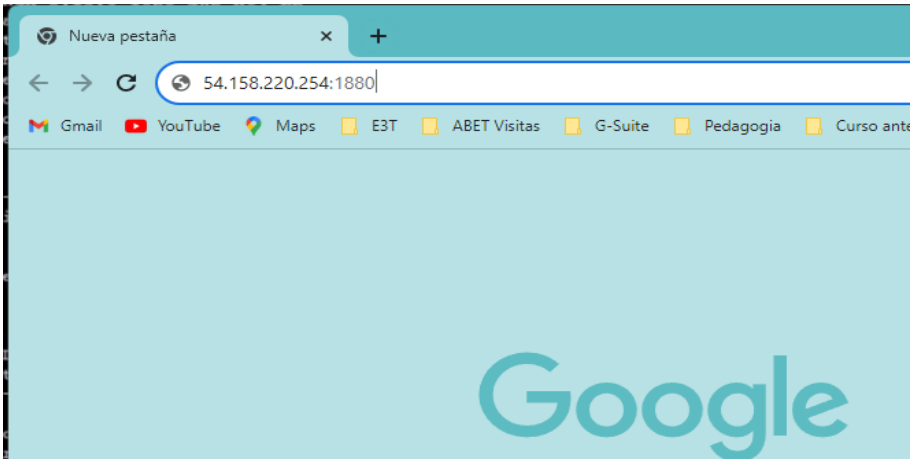
```
ubuntu@ip-172-31-43-14:~$ ps aux | grep node-red
ubuntu      1201  0.0  0.2   7004  2304 pts/0    S+   04:01   0:00 grep --color=auto node-red
ubuntu@ip-172-31-43-14:~$
```

Pruebas de funcionamiento de la interfaz gráfica

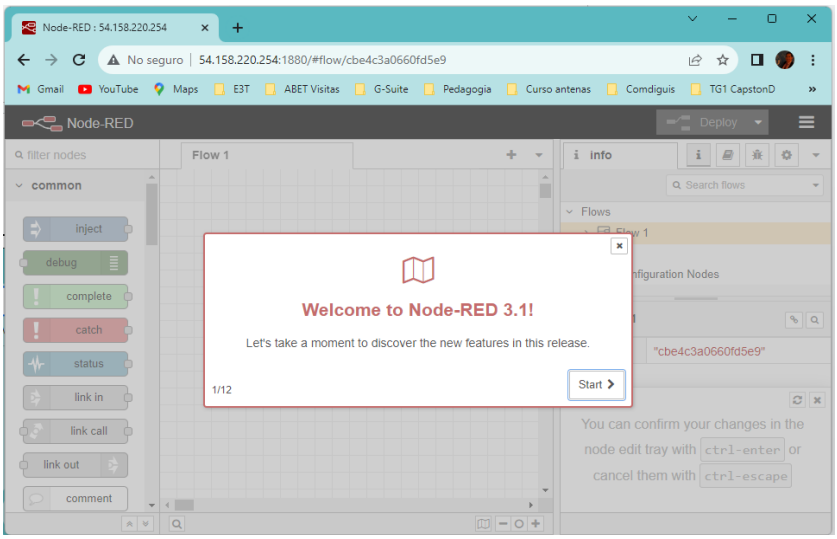
Ve a AWS y revisa cual es la IP pública de tu instancia



Ve a un navegador y escribe la ip seguida de “:” y el puerto de este servicio que es 1880



Como resultado estarás en la interfaz de programación visual de Node-RED



¿Comprueba que pasa si en tu instancia sales de la consola Node-RED con Ctrl+C. Puedes seguir viendo tu interfaz gráfica en el navegador como se hizo en el punto anterior? Debe refrescar el navegador para conocer qué pasa. Verás que ya no podrás trabajar en la interfaz gráfica, porque se ha cerrado Node-RED

Averiguar si Node-RED está corriendo. Es decir, si tu máquina tiene una instancia de Node-RED

En Ubuntu, comprueba si hay una instancia abierta de Node-RED, con el siguiente comando:

Ps aux | grep node-red

La siguiente respuesta indica que hay una instancia de grep, pero no de node-RED:

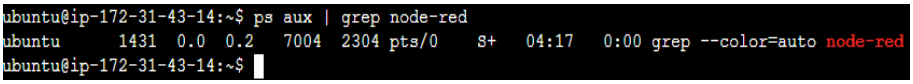
```
ubuntu@ip-172-31-43-14:~$ ps aux | grep node-red
ubuntu      1201  0.0  0.2   7004  2304 pts/0    S+   04:01   0:00 grep --color=auto node-red
ubuntu@ip-172-31-43-14:~$
```

Conclusión: si no estás en la consola, has salido de ella con Ctrl+C. No tendrás una instancia de Node-RED. La interfaz gráfica remota no funcionará. Pero es posible hacer una reconfiguración para que esto no ocurra como se explica en el siguiente capítulo

Averiguar qué pasa si tienes Node-RED corriendo, reinicias tu máquina, ¿con ello Node-RED también se reinicia?

Inicia node-RED con:

Node-red -v

Reinicia tu máquina desde la consola de AWS
Con el siguiente comando comprueba si hay una instancia de Node-RED corriendo:
Ps aux | grep node-red
La respuesta es que no, veremos esto, pero eso solo indica que hay una instancia de grep


Tarea 8. Configuración de Node Red.

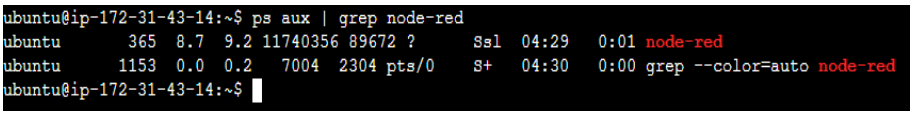
Paso 1. Configuración de Node-Red para que arranque automáticamente

En el paso anterior vimos que, con la instalación hecha, Node-RED no se reinicia automáticamente. Supongamos que tienes un servicio al público y por alguna razón se fue se apagó la máquina, entonces en adelante, el servicio no estará disponible, aunque la máquina se reinicie. De allí la importancia de una configuración para lograr que Node-RED arranque automáticamente. La siguiente es la configuración que recomienda chatgpt para lograr que Node-RED arranque automáticamente cuando se enciende la VM:

Comando	Explicación
Sudo nano /etc/systemd/system/node-red.service	Se crea un archivo para el control del servicio para Node-RED por parte de systemd. El archivo resulta abierto con el editor nano
[Unit] Description=Node-RED After=network.target [Service] Execstart=/usr/bin/node-red Restart=on-failure User=ubuntu [Install] Wantedby=multi-user.target	Es el texto que se debe pegar en el archivo creado. En nuestro caso el usuario se llama ubuntu y será con ese usuario que arrancará el servidor. Esto representa una orden a systemd para iniciar Node-RED como un servicio después de que la red esté disponible. Si Node-RED falla, systemd intentará reiniciarlo automáticamente. El servicio se ejecutará con el usuario ubuntu, y se instalará para arrancar automáticamente en el nivel de ejecución multiusuario. Parece ser un prerrequisito para hacer que Node-

	RED arranque automáticamente cuando arranca el sistema. Hay que guardar y salir con Ctrl+O, Ctrl+X
Sudo systemctl daemon-reload	Recargar la configuración de systemd
Sudo systemctl enable node-red.service	Habilita el servicio de Node-RED para que se inicie automáticamente al arrancar el sistema

Pruebas de arranque automático

Reinicia la máquina
Envia el comando para revisar si hay instancias corriendo
Ps aux | grep node-red
Resultado

Conclusión: Node-RED arranca automáticamente, el comando anterior muestra inas instancia de node-red corriendo, la cual tiene el IDE 365

Paso 2. Comprueba que, sin tocar la máquina, el navegador de internet que permite el acceso remoto a Node-RED. En el navegador escribe la IP pública seguida de ": 1880", por ejemplo:

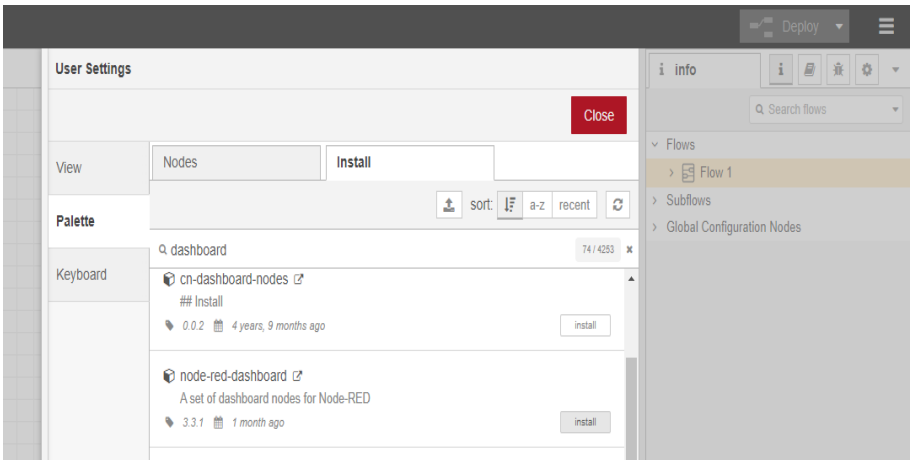
Http://54.236.56.149:1880

Una falla común es que: uno toma la IP que aparece en la instancia, pero resulta que puede ser la IP de una corrida anterior, debido a que al navegador le toma tiempo refrescar los datos. Para descartar esta falla, usa el botón de refrescar

Paso 3. Instalar complementos de Node-RED

> Manage Palette > Install > nombre de lo que desea instalar > Install> Install

- Node-red-dashboard



- De manera similar instalar: node-red-contrib-ui-led
- Instalar tambien: node-red-mysql

Tarea 9. Mantenimiento de Node RED.

Esta tarea se dedica a conquistar habilidades para hacerle mantenimiento a estos servidores en aspectos como: saber si se está presentando alguna falla, si hay que hacer configuraciones, etc. Básicamente, dominar el uso de comandos útiles

Node Red	
Comando	Explicación
node-red -log	Permite conocer el estado del servidor Node Red. Por ejemplo, si está en ejecución
sudo systemctl start node-red.service	Iniciar el servicio de Node-RED
sudo systemctl stop node-red.service	Detener el servicio de Node-RED
sudo systemctl restart node-red.service	Reiniciar el servicio de Node-RED

REFERENCIAS

N. U. Khuzairi, S. Saon, A. K. Mahamad, M. S. M. Zainordin, S. Yamaguchi and M. A. Bin Ahmadon. (s/f). Weather Station Monitoring System with Node-RED,.

Nodered.org. (s/f). *Https://nodered.org/docs/user-guide/*.

Universidad industrial de Santander

Generalidades de Node-RED.

Enero de 2024

Carlos Eduardo Silva Sepúlveda

Brayan Hernando Gonzales Mendoza

Contenido

Introducción

El propósito de esta guía es proporcionar a los usuarios una comprensión detallada de las generalidades de Node-RED, desde la instalación inicial hasta la personalización avanzada de los dashboards que pueden ser creados con esta plataforma. Exploraremos conceptos fundamentales en Node-RED, incluyendo el transporte de datos, y nos enfocaremos en los nodos más relevantes para el procesamiento de datos y su posterior visualización en los dashboards. Esta guía sirve como una sólida base para comprender conceptos esenciales en la utilización de esta herramienta de programación visual basada en flujos, ofreciendo a los usuarios las habilidades necesarias para aprovechar al máximo las capacidades de Node-RED.

Capítulo 1

1. Node-RED

1.1 ¿Qué es Node-RED?

Node-RED es un entorno de desarrollo de código abierto que facilita la creación de aplicaciones basadas en flujos (flow-based programming). Desarrollado por IBM, Node-RED proporciona una interfaz gráfica basada en navegador que permite a los usuarios conectar nodos predefinidos para construir lógica de programación de una manera visual e intuitiva.

En términos sencillos, Node-RED permite a los desarrolladores y usuarios crear aplicaciones mediante la interconexión de bloques de construcción llamados nodos. Cada nodo realiza una función específica y puede ser arrastrado y soltado en la interfaz gráfica, donde los usuarios pueden conectarlos para formar flujos de trabajo. Estos flujos pueden abordar una variedad de tareas, desde la automatización del hogar hasta la integración de sistemas complejos.

Los nodos en Node-RED representan servicios, dispositivos, funciones, o conexiones a bases de datos, entre otras cosas. La plataforma es altamente extensible, lo que significa que los usuarios pueden crear sus propios nodos o instalar nodos adicionales desarrollados por la comunidad para ampliar la funcionalidad.

Node-RED se ejecuta en Node.js y se integra fácilmente con una amplia gama de servicios web y dispositivos IoT (Internet de las cosas). Su enfoque visual y de arrastrar y soltar lo hace accesible para aquellos que pueden no tener experiencia en programación convencional, permitiendo la creación rápida de flujos de trabajo complejos con una curva de aprendizaje baja.

1.2 Arquitectura

La arquitectura de Node-RED se basa en un entorno de ejecución Node.js y una interfaz web intuitiva. Su editor gráfico permite a los usuarios arrastrar y soltar nodos para construir flujos de trabajo. La comunicación entre nodos se realiza mediante conexiones que transportan mensajes en formato JSON. Con un runtime

para ejecución y un gestor de procesos para administración, Node-RED destaca por su modularidad y extensibilidad, facilitando la creación de aplicaciones basadas en flujos en diversos contextos.

Capítulo 2

2. Instalación y ejecución

2.1 Requisitos previos

Asegúrate de tener Node.js instalado en tu sistema. Puedes descargar la última versión desde nodejs.org.

2.2 Instalación

Para instalar Node-RED puedes usar el comando npm que viene con node.js:

```
sudo npm install -g --unsafe-perm node-red
```

Si está utilizando Windows, no inicie el comando con sudo. Ese comando instalará Node-RED como un módulo global junto con sus dependencias. Puede confirmar que se ha realizado correctamente si el resultado final del comando es similar a:

```
+ node-red@1.1.0
added 332 packages from 341 contributors in 18.494s
found 0 vulnerabilities
```

2.3 Ejecución

Una vez instalado como módulo global, puede usar el comando node-red para iniciar Node-RED en su terminal. Puede usar Ctrl-C o cerrar la ventana de terminal para detener Node-RED.

```
$ node-red

Welcome to Node-RED
===================
```

```
30 Jun 23:43:39 - [info] Node-RED version: v1.3.5
30 Jun 23:43:39 - [info] Node.js  version: v14.7.2
30 Jun 23:43:39 - [info] Darwin 19.6.0 x64 LE
30 Jun 23:43:39 - [info] Loading palette nodes
30 Jun 23:43:44 - [warn] rpi-gpio : Raspberry Pi specific node
set inactive
30 Jun 23:43:44 - [info] Settings file  : /Users/nol/.node-
red/settings.js
30 Jun 23:43:44 - [info] HTTP Static    : /Users/nol/node-
red/web
30 Jun 23:43:44 - [info] Context store  : 'default'
[module=localfilesystem]
30 Jun 23:43:44 - [info] User directory : /Users/nol/.node-red
30 Jun 23:43:44 - [warn] Projects disabled : set
editorTheme.projects.enabled=true to enable
30 Jun 23:43:44 - [info] Creating new flows file :
flows_noltop.json
30 Jun 23:43:44 - [info] Starting flows
30 Jun 23:43:44 - [info] Started flows
30 Jun 23:43:44 - [info] Server now running at
http://127.0.0.1:1880/red/
```

La salida del registro le proporciona varios datos:

- Las versiones de Node-RED y Node.js
- Se produjo algún error al intentar cargar los nodos de la paleta.
- La ubicación de su archivo de configuración y directorio de usuarios
- El nombre del archivo de flujos que está utilizando.

2.4 Acceso a la Interfaz de Usuario

Abre tu navegador web y accede a http://localhost:1880. Aquí encontrarás la interfaz de usuario de Node-RED.

Capítulo 3

3. Interfaz

Una vez instalado y ejecutado Node-RED, podemos acceder a ella por el puerto 1880 en donde se visualizará la interfaz de programación. La estructura de la interfaz de desarrollo de Node-RED se organiza en diversas pestañas, cada una con un propósito específico, como se observa en la ilustración 1.

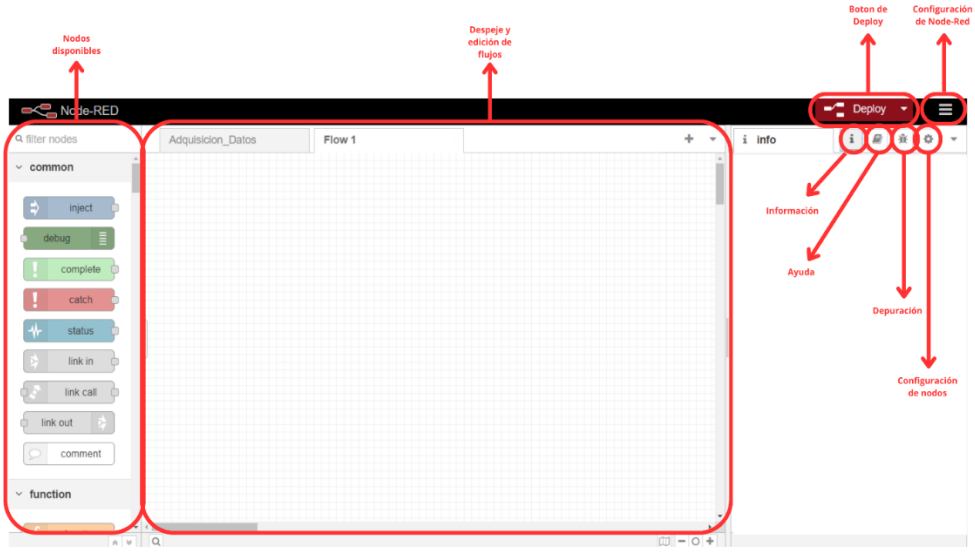


Ilustración 1. Plataforma de desarrollo de Node-Red

La pestaña de depuración es esencial para rastrear y verificar el flujo de datos en tiempo real, permitiendo la observación de mensajes de depuración, la inspección de variables y la verificación del correcto funcionamiento de los nodos. La pestaña de flujos es el lugar donde se construyen y organizan los flujos de trabajo. Aquí, se pueden arrastrar y soltar nodos, conectarlos y definir la lógica de la aplicación deseada. La pestaña de ayuda proporciona información detallada sobre cada nodo y su funcionalidad, facilitando la comprensión y el uso eficiente de Node-RED. Adicionalmente, existen dos pestañas más dedicadas a la configuración e información detallada de los nodos, donde se pueden realizar modificaciones en aspectos específicos de cada nodo. Por último, el botón "Deploy" permite guardar toda la configuración nueva en el flujo y, al mismo tiempo, ejecutarla. Node-RED permite la instalación de paquetes y extensiones para ampliar sus capacidades, respaldado por una sólida comunidad de usuarios.

Capítulo 4

4. Transporte de datos entre nodos

En Node-RED, el msg.payload es un campo específico de un objeto msg que se utiliza comúnmente para transportar datos a lo largo de los nodos en un flujo. El objeto msg (mensaje) es la unidad básica de información en Node-RED y puede contener varios campos, siendo msg.payload uno de los más utilizados ya que allí se encuentra la carga útil que podemos procesar y manipular dependiendo de nuestras necesidades. En este objeto msg también se encuentra otros atributos como el msgid, que indica el id único del mensaje y el tópico, atributo que sirve para identificar y clasificar mensajes por esta variable que puede ser de tipo string, entero, decimal y todo tipo de variables que soporta Node-Red. En el siguiente código se puede observar los atributos típicos en un objeto msg.

```
msg = { _msgid: “a735a576745c39d0”,
        payload: ”La carga útil está aquí”,
        topic: “El tópico está aquí”
      }
```

Capítulo 5

5. Nodos centrales

La paleta de Node-RED cuenta con un conjunto predefinido de nodos, los cuales son elementos fundamentales para la creación de flujos. En esta sección, se resaltan los nodos esenciales que es importante conocer.
Cada nodo incorpora documentación que se encuentra disponible en la pestaña de Información, accesible desde la barra lateral al seleccionar un nodo específico.

5.1 Nodo Inject

Ilustración 2.Nodo Inject

Una de las herramientas más utilizadas para generar los mensajes es el nodo “Inject”, ya que permite a los usuarios inyectar mensajes manualmente en el flujo de trabajo en momentos específicos o de acuerdo con una programación establecida. Este nodo puede configurarse para enviar mensajes en intervalos regulares, en momentos específicos del día o en respuesta a eventos externos. Este nodo es útil para simular eventos, programar ejecuciones o controlar el flujo de datos en aplicaciones Node-RED.

5.2 Nodo Function

Ilustración 3. Nodo Function

El nodo "Function" en Node-RED es una utilidad versátil que permite a los usuarios agregar lógica de programación personalizada a sus flujos. Al utilizar JavaScript, los usuarios pueden escribir código para procesar y transformar datos entre nodos. Este nodo es esencial para realizar tareas específicas que no están cubiertas por los nodos predefinidos, ofreciendo una flexibilidad considerable en el desarrollo de flujos personalizados en Node-RED. Con el nodo "Function", los usuarios pueden adaptar y extender la funcionalidad de sus flujos de manera eficiente y adaptarlos a sus necesidades específicas.

5.3 Nodo Join

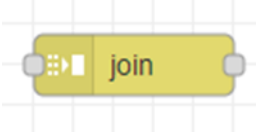


Ilustración 4. Nodo Join.

El nodo "Join" en Node-RED es una herramienta que permite a los usuarios combinar y sincronizar mensajes de diferentes flujos para facilitar la manipulación de datos. Con este nodo, los usuarios pueden unir mensajes basándose en distintos criterios, como el tiempo o la secuencia de eventos. Esto resulta especialmente útil cuando se trabaja con datos provenientes de múltiples fuentes y se necesita consolidar la información antes de su procesamiento adicional. El nodo "Join" proporciona una manera eficaz de gestionar y coordinar datos en flujos complejos, mejorando la flexibilidad y funcionalidad de las aplicaciones desarrolladas en Node-RED.

5.4 Nodo Change

Ilustración 5. Nodo Change

El nodo "Change" en Node-RED es una herramienta versátil que permite a los usuarios realizar modificaciones y transformaciones específicas en los mensajes que fluyen a través de sus flujos. Con este nodo, se pueden realizar diversas operaciones, como agregar, modificar o eliminar propiedades de los mensajes. Es esencial para adaptar y ajustar los datos según los requisitos específicos del flujo, brindando a los usuarios un control preciso sobre la manipulación de la información en sus aplicaciones Node-RED.

5.5 Nodo Debug

Ilustración 6. Nodo Debug

El nodo "Debug" en Node-RED es una herramienta esencial para el desarrollo y la depuración de flujos. Permite a los usuarios visualizar información detallada sobre los mensajes que fluyen a través de sus nodos, facilitando la identificación de problemas y la comprensión del flujo de datos. Este nodo se utiliza para imprimir mensajes, variables o información específica en la ventana de depuración de Node-RED. Su función principal es ayudar a los desarrolladores a

comprender el comportamiento de sus flujos y a diagnosticar cualquier problema que pueda surgir durante la ejecución del programa.

5.6 Nodo Template

Ilustración 7. Nodo template

El nodo "template" en Node-RED permite la creación de contenido personalizado mediante el uso de plantillas. Con este nodo, los usuarios pueden generar datos o mensajes complejos utilizando lenguaje de marcado, variables y funciones JavaScript. Esto facilita la manipulación avanzada de datos antes de enviarlos a otros nodos en el flujo, ofreciendo flexibilidad para personalizar la salida de acuerdo con las necesidades específicas del usuario. El nodo "template" es especialmente útil para la generación de mensajes estructurados y formateados según requerimientos particulares en el flujo de Node-RED.

Capítulo 6

6. Nodos dashboard

Los nodos "Dashboard" en Node-RED son componentes fundamentales para la creación de interfaces gráficas de usuario (GUI) como se oberva en la ilustración 8. Estos nodos permiten a los usuarios diseñar paneles de control interactivos para visualizar y controlar datos en tiempo real. Al utilizar widgets predefinidos como gráficos, medidores y botones, los nodos Dashboard simplifican la creación de aplicaciones web interactivas sin necesidad de codificación adicional. Son esenciales para la construcción de aplicaciones de monitoreo, control y visualización de datos, proporcionando una interfaz fácil de usar para interactuar con los flujos de Node-RED.

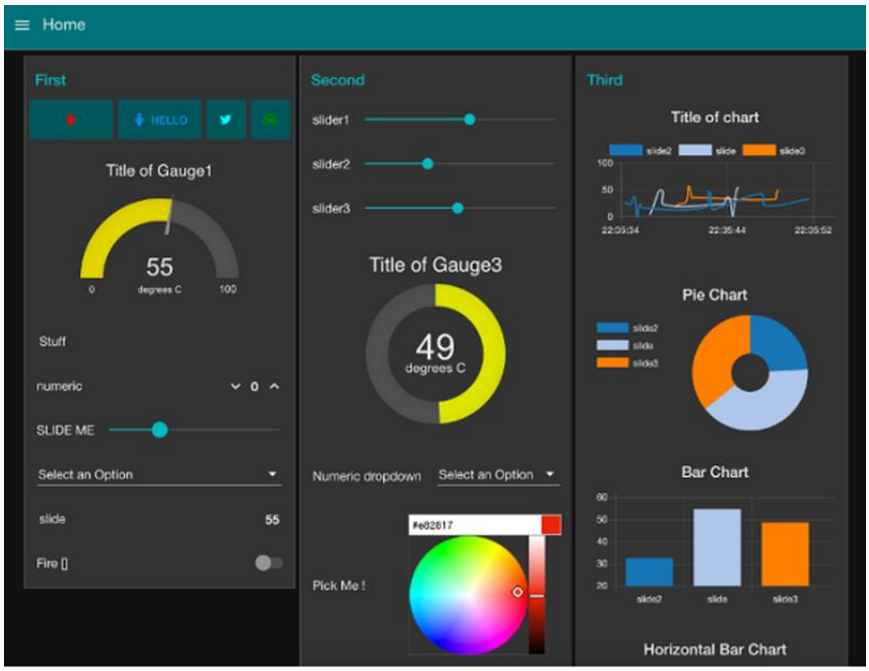


Ilustración 8. Interfaz hecha con nodos dashboard en Node-RED.

6.1 Nodo ui_button

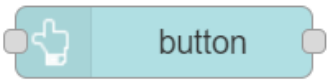


Ilustración 9. Nodo ui_button.

El nodo ui_button de Node-RED es una herramienta versátil que permite la creación de botones interactivos en el panel de control. Configurable con diversas opciones de estilo y comportamiento, este nodo se convierte en una solución eficaz para ejecutar acciones específicas en el flujo mediante la interacción directa del usuario. Al presionar el botón, puede desencadenar eventos, facilitando la implementación de interfaces de usuario dinámicas y de fácil manipulación.

6.2 Nodo ui_switch

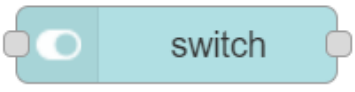


Ilustración 10. Nodo ui_switch.

El nodo ui_switch introduce un interruptor de encendido/apagado en el panel de control, brindando una solución visual e intuitiva para controlar dispositivos o funciones con estados binarios. Los usuarios pueden cambiar el estado del interruptor directamente desde la interfaz web, y el nodo puede generar salidas en el flujo en respuesta a estos cambios, proporcionando una herramienta valiosa para el control remoto de dispositivos y sistemas.

6.3 Nodo ui_slider

Ilustración 11. Nodo ui_slider.

Con el nodo ui_slider, los usuarios pueden ajustar valores de manera intuitiva mediante un control deslizante en el panel de control. Este nodo es ideal para la configuración de parámetros variables, ya que puede recibir entradas para establecer la posición del deslizador y generar salidas cuando se ajusta. Ofrece una interfaz de usuario interactiva y eficiente para manipular y controlar valores numéricos en tiempo real.

6.4 Nodo ui_text

Ilustración 12. Nodo ui_text.

El nodo ui_text se utiliza para mostrar texto o datos en tiempo real en el panel de control. Esencial para presentar información dinámica, este nodo permite a los usuarios monitorear datos actualizados de manera continua y ofrece una forma clara y legible de proporcionar retroalimentación visual.

6.5 Nodo ui_chart

Ilustración 13. Nodo ui_chart.

Para la visualización efectiva de datos, el nodo ui_chart en Node-RED permite crear gráficos interactivos en el panel de control. Diseñado para representar series temporales y datos numéricos, este nodo proporciona una representación visual clara y fácil de entender, mejorando la capacidad del usuario para interpretar información compleja.

6.6 Nodo ui_gauge

Ilustración 14. Nodo ui_gauge.

El nodo ui_gauge facilita la creación de medidores visuales en el panel de control, ofreciendo una representación gráfica intuitiva de valores numéricos. Adecuado para mostrar información como niveles de temperatura, humedad o cualquier variable numérica, este nodo contribuye a una interfaz de usuario más comprensible y fácil de interpretar.

6.7 Nodo ui_dropdown

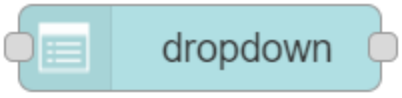


Ilustración 15. Nodo ui_dropdown.

Diseñado para la selección de opciones específicas, el nodo ui_dropdown presenta una lista desplegable en el panel de control. Permite a los usuarios elegir entre varias opciones predeterminadas, y el nodo puede generar salidas cuando se realiza una selección, proporcionando una forma eficiente y estructurada de interactuar con datos o configuraciones.

6.8 Nodo ui_template

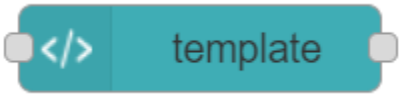


Ilustración 16. Nodo ui_template.

El nodo ui_template brinda una versatilidad excepcional al permitir la creación de interfaces de usuario altamente personalizadas mediante plantillas HTML, CSS y JavaScript. Este nodo es ideal para aquellos que buscan adaptar la apariencia y funcionalidad del panel de control a requisitos específicos y necesidades de diseño, proporcionando un lienzo creativo para la implementación de elementos visuales únicos y dinámicos.

6.9 Nodo ui_date_picker

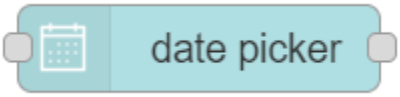


Ilustración 17. Nodo ui_date_picker.

El nodo ui_date_picker facilita la interacción con datos temporales al proporcionar un selector de fecha en el panel de control. Los usuarios pueden seleccionar fechas de manera intuitiva, y el nodo puede generar salidas cuando se selecciona una fecha específica. Es útil para programar eventos basados en el tiempo y para trabajar con datos temporales de manera eficiente.

Capítulo 7

7. Aspecto visual del dashboard

Los Dashboards ofrecidos por Node-RED están conformados por diversas secciones que encapsulan los elementos de interfaz proporcionados por los nodos mencionados anteriormente. Las secciones primordiales se conocen como "tabs" y son visualizadas en la barra vertical derecha de la página web. Estos "tabs" contienen grupos que albergan los nodos con los que los usuarios pueden interactuar y sobre los cuales realizan operaciones. En la Ilustración 18 se observa un Dashboard para el monitoreo de variables ambientales con cada uno de sus componentes.

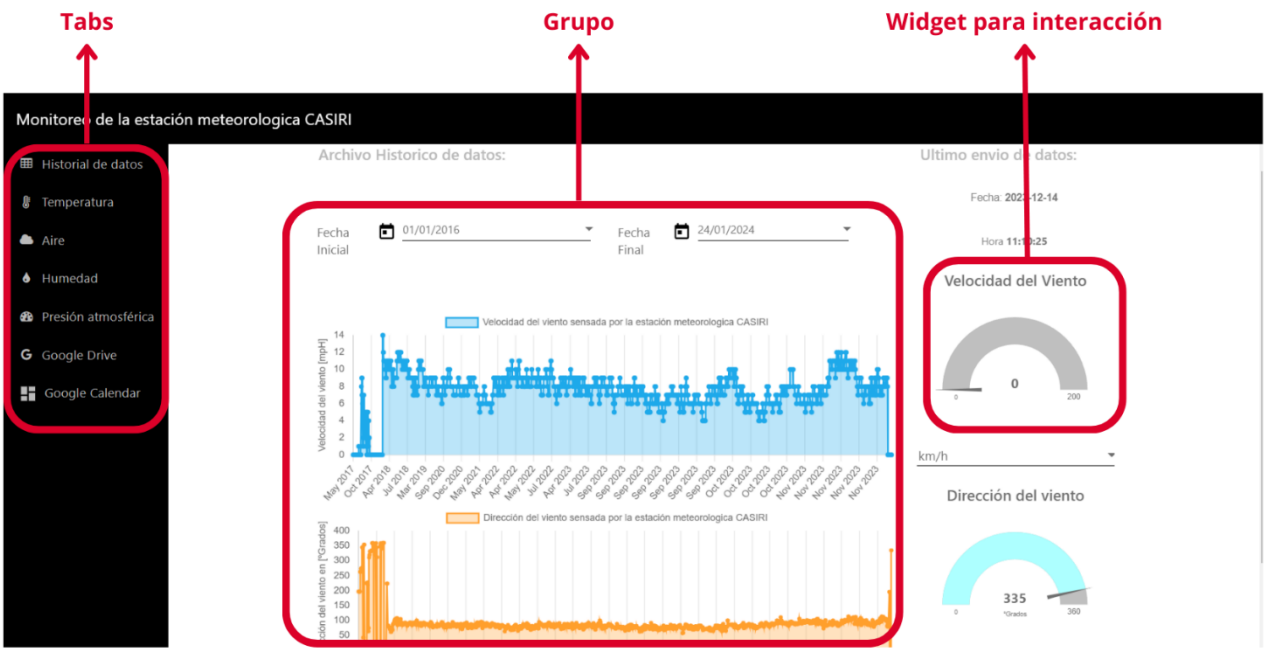


Ilustración 18. Vista general de un dashboard.

7.1 Personalización de dashboard

Node-RED permite a los usuarios personalizar el diseño de su dashboard según sus preferencias, brindando opciones que van desde la modificación del color de cada sección, desde la barra hasta los objetos en el dashboard, hasta la posibilidad de ajustar el tipo y color de fuente de letra para cada elemento. Para acceder a estas configuraciones, simplemente se hace clic en la pestaña ubicada a la derecha, se selecciona la sección "dashboard" y posteriormente se elige la opción "Theme", como se muestra en la ilustración 19. Esto proporciona un control detallado sobre la apariencia visual del dashboard, permitiendo una experiencia personalizada para el usuario.

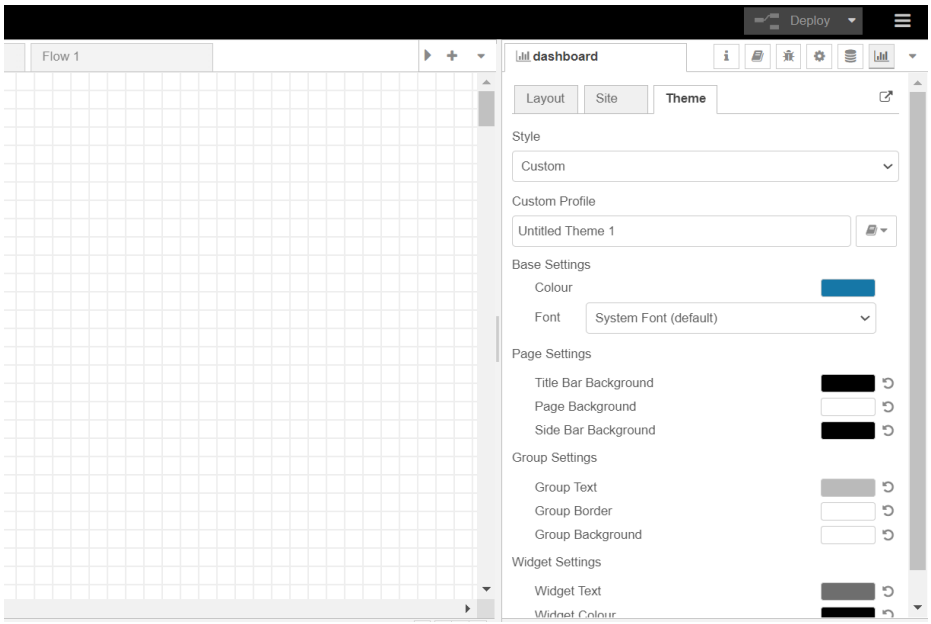


Ilustración 19. Vista general de la configuración del tema de dashboard.

7.2 Posición del Layout

Node-RED también permite al usuario personalizar la posición de los objetos que están en el dashboard, así como su tamaño. Para ello se accede a la pestaña de "dashboard" y luego clic en layout, allí se selecciona el tab a configurar y se ajusta el tamaño y la posición de cada objeto, tal como se observa en la ilustración 20.

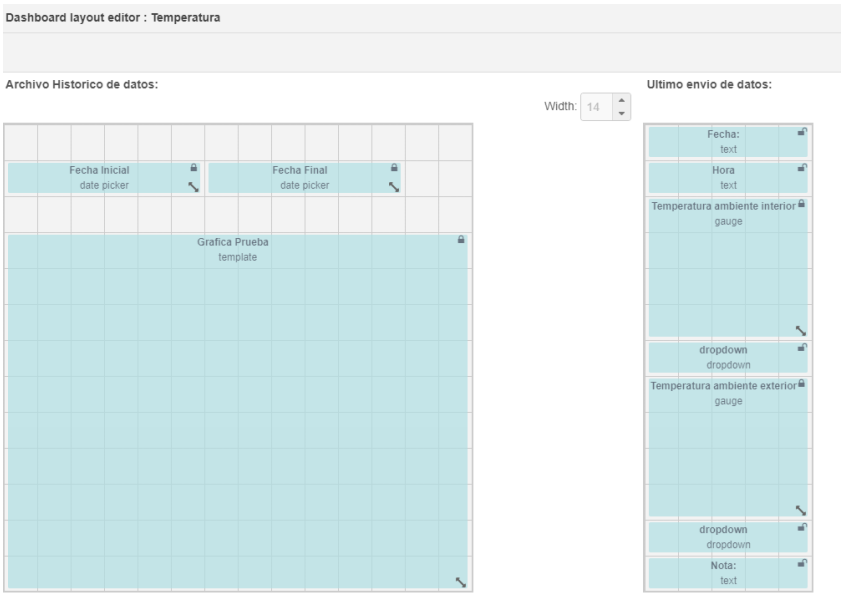


Ilustración 20. Ajuste de posición y tamaño dentro de un tab en un dashboard de Node-RED.

REFERENCIAS

N. U. Khuzairi, S. Saon, A. K. Mahamad, M. S. M. Zainordin, S. Yamaguchi and M. A. Bin Ahmadon. (s/f). Weather Station Monitoring System with Node-RED,.

nodered.org. (s/f). *https://nodered.org/docs/user-guide/*.

3.3. Protocolo de Toma de Datos

Este documento presenta el protocolo de toma de datos a seguir durante la **“VIII EXPEDICIÓN CIENTÍFICA DE COLOMBIA A LA ANTÁRTICA VERANO AUSTRAL 2021– 2022”**, con el fin de proveer una guía al momento de realizar las mediciones en campo para lograr una correcta y ágil toma de datos. En la sección 3.3.1. se encuentra una recomendación para revisión de equipos antes de comenzar las mediciones, en la sección 3.3.2 se presenta una descripción del software a utilizar para la toma de datos. Por último, en la sección 3.3.3, se muestra el protocolo con las instrucciones que deben seguir los asistentes al evento para la toma de datos.

3.3.1. Revisión de equipos.

Antes de comenzar a tomar datos se debe realizar una última valoración para verificar que todo se encuentre en condiciones óptimas para empezar, verifique que los equipos se encuentren bien conectados, que las antenas tengan una correcta posición y un buen anclaje al suelo.

3.3.2. Software a Utilizar.

Para la toma de datos se utilizará el software GNU Radio con el cual se obtiene la medición de señales que se encuentren presentes en el ambiente en donde se estén realizando las mediciones.

3.3.2.1 Sistema RFI

Descripción del software

La arquitectura de software implementada utiliza GNU Radio como software para la configuración del SDR y para la definición del tratamiento de los datos. En primera instancia se diseñó un diagrama de bloques en GNURadio y a partir de este se genera un archivo en lenguaje Python (.py) donde se obtiene la codificación de los bloques utilizados. Por otra parte se genera un archivo binario (.bit), utilizando un bloque file sink.

Se realizó un paquete de software para la Ettus E310.

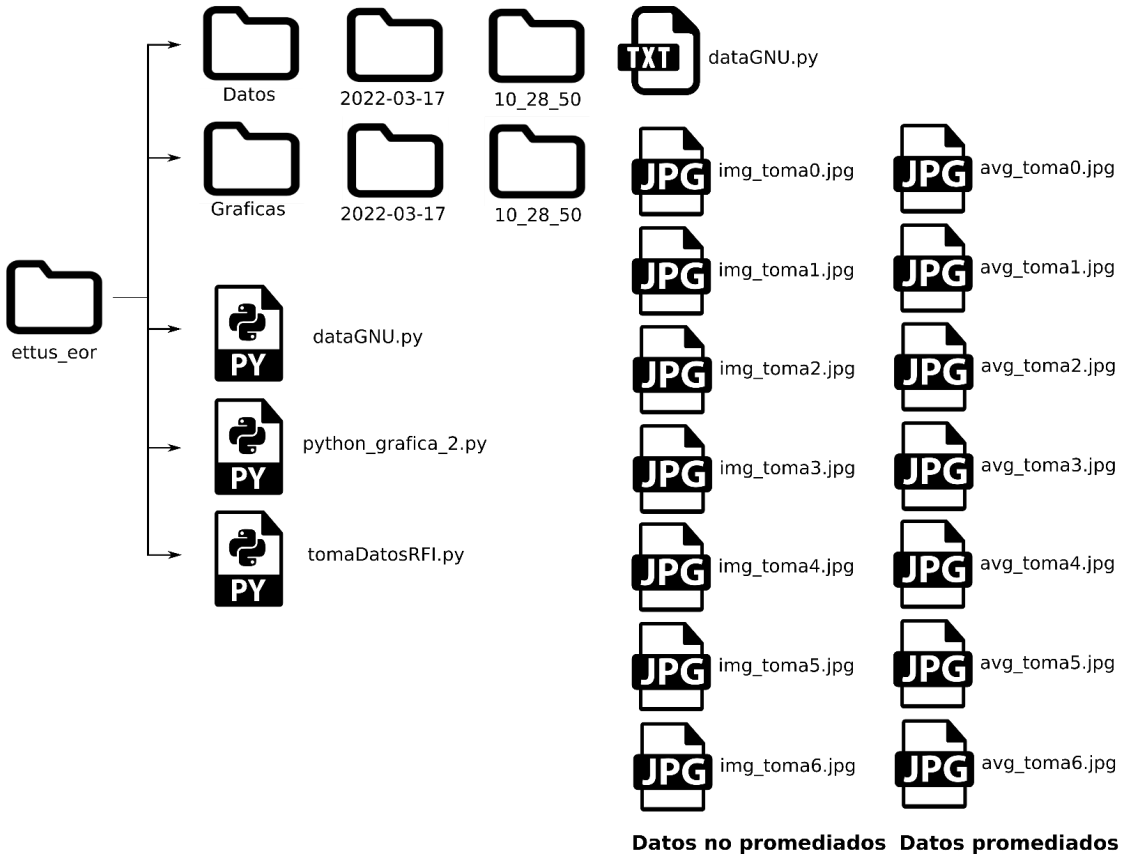
Se tiene una carpeta principal “etus_rfi” la cual contiene 3 archivos con extensión .py, y dos carpetas nombradas “datos” y “graficas”. Los tres archivos .py utilizados son: “python_graficas_2.py”, y “dataGNU.py” y “tomaDatosRFI.py”.

- python_graficas_2.py: es el archivo generado por GNU a partir del diagrama de bloques implementado.
- dataGNU.py: extrae del archivo python_graficas.py , la frecuencia central a partir de 108 MHz. Se define un sample rate de 16 Ms/s equivalentes a obtener ventanas de visualización de 16 MHz y un FFT size de 6144. Este archivo invoca y lee el archivo binario .bin y genera los plots de visualización. En dicho archivo se invoca el archivo .bit y se convierten los valores a vectores por medio de numpy. A continuación se calcula la

longitud del vector y se asigna el tamaño de los datos de potencia correspondientes al eje y. Posteriormente se definen las frecuencias de operación y los límites para la ventana de visualización. Luego se define el incremento o pasos de el eje x y se genera un vector que rellena y organiza los paquetes de datos; se pasa a un arreglo vectorial (numpy). Posteriormente se realiza la graficación donde se define y ubica la frecuencia central y los límites en los ejes *x* y *y*, así como la cantidad de ventanas de visualización.

- tomaDatosRFI.py: toma los datos de GNU e itera 6 veces variando la frecuencia central para cubrir el rango de 100 a 200 MHz. Extrae las fechas y las horas de las tomas de datos y genera los archivos que alimentan las carpetas de gráficas y datos. En las líneas 24 a 45 de dicho archivo se debe definir la ruta de archivos según la ubicación de la carpeta raiz utilizada e.g. con el fin de instalar y compilar el software desde cualquier dispositivo Linux. Al compilar este archivo se generan 6 gráficas en formato .jpg. Cada una de estas gráficas corresponde a una ventana de frecuencias diferentes así: de 100 a 116 MHz, de 116 a 132 MHz, de 132 a 148 MHz, de 148 a 164 MHz , de 164 a 180 MHz, de 180 a 196 MHz y de 196 a 212 MHz. Al finalizar la toma de datos restablece la frecuencia central a 108 MHz para poder efectuar una nueva toma de datos.

Las gráficas obtenidas quedan organizadas y clasificadas por subcarpetas de fecha y hora de la toma de datos. La data cruda queda almacenada en un archivo .txt la cual es sensible de ser post-procesada.



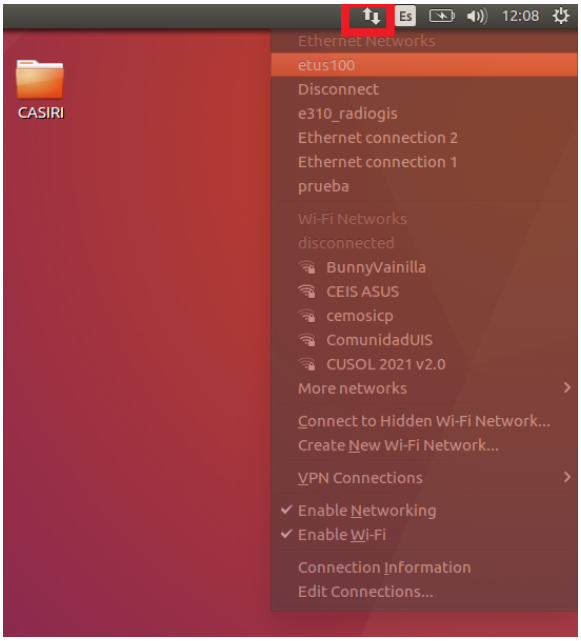
3.3.3. Toma de Datos.

Para comenzar las mediciones realice los siguientes pasos:

Sistema RFI

1. Conecte la Ettus a un tomacorriente y al PC mediante cable Ethernet, se prenderá automáticamente.

2. Establecer conexión en la red local etus100.



3. Abrir un terminal (Ctrl + Alt + T) y ejecutar “sudo arp-scan --localnet”, solicitará la contraseña”ieeeieee” debe salir una dirección IP y el nombre del instrumento (national instruments)

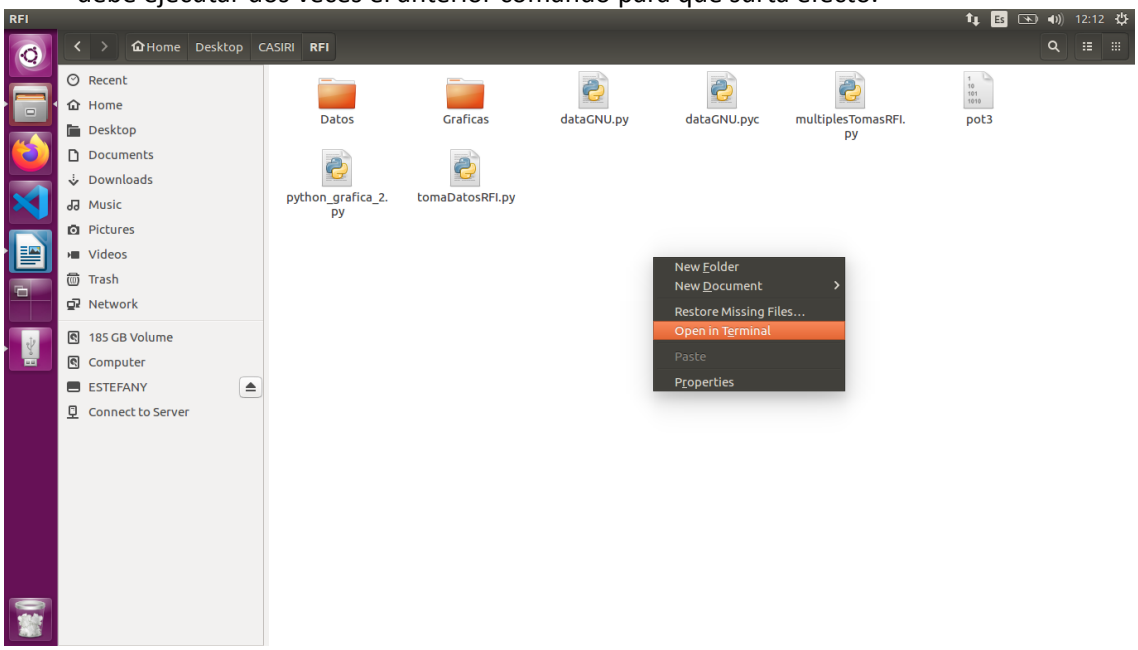
4. Luego digite el siguiente comando “ssh root@#####(dirección IP del paso anterior)”, luego de esto digite el comando “usrp_e3x0_network_mode”. Estos pasos permitirán reconocer el dispositivo para la toma de datos.

5. Abra la carpeta llamada RFI que se encuentra en la carpeta CASIRI ubicada en el escritorio o en la ruta “/home/comdiguis/Desktop/CASIRI/RFI”. Si no está utilizando un pc de Radiogis edite el archivo “tomaDatosRFI.py” para definir la ruta de las carpetas en donde se guardaran los datos, para ello: En las líneas 24 a 45 del archivo se debe definir la ruta de las carpetas “Datos” y “Graficas”, según la ubicación de la carpeta raíz utilizada, con el fin de compilar el software desde cualquier dispositivo Linux. Al compilar este archivo se generan 14 gráficas en formato .jpg (7 img_toma# y 7 avg_toma#) y 7 archivos .txt.

6. En la línea 54 del archivo “tomaDatosRFI.py” se establece la frecuencia central (newFrecuency) de la primera toma, si se desea cambiar esta frecuencia edite esta línea.

7. Desde un terminal ubicado en la carpeta RFI escriba el siguiente comando “python tomaDatosRFI.py”. Esto generará una sola toma de datos. Si cambió la frecuencia central

debe ejecutar dos veces el anterior comando para que surta efecto.



8. Si desea hacer múltiples tomas de datos ejecute “python multiplesTomasRFI.py”, le pedirá el número de tomas y el tiempo entre ellas en segundos.

9. Los archivos se guardarán en las carpetas “Datos” y “Graficas” ubicadas en la carpeta RFI, dentro de estas carpetas se crearán carpetas con la fecha en que se toman los datos y dentro de estas carpetas se crean otras con la hora en la que se tomaron.

Subsistema Cámara OASC

1. Abrir un terminal y ejecutar “sudo arp-scan --localnet” debe salir una dirección IP y el nombre del instrumento (Nvidia).

2. Luego digite el siguiente comando “ssh radiogis@#####(dirección IP del paso anterior)”. Le pedirá la contraseña que es “radio”. Estos pasos permitirán ingresar a la tarjeta Jetson.

3. Desde este momento el terminal se encuentra ejecutado en la tarjeta.

4. Ejecute el comando “ps -A |grep indiserver” esto arroja los procesos y sus números de identificación. Luego escriba el comando “kill ##(el número que identifique el proceso indiserver)”. Con esto se asegura que ningún servidor indi se encuentre ejecutándose.

5. Ejecute el servidor con el comando “indiserver -v -m 100 indi_sx_ccd”. Después de ejecutar el comando este terminal queda bloqueado solo para ver el estado del proceso.

6. Sin cerrar el terminal anterior, abra uno nuevo y vuelva a realizar el paso 2.

7. Ahora se debe ejecutar el comando “cd Desktop/radioastronomia” y luego el comando “python captura.py”. Luego de esto se le pide que coloque cuantas capturas desea que se realicen y el tiempo (segundos) entre ellas.

8. En un nuevo terminal digite el siguiente comando “sftp radiogis@#####(dirección IP del primer paso)”. Le pedirá la contraseña que es “radio”. Estos pasos permitirán ingresar a la tarjeta Jetson en modo transferencia de archivos.

9. Para mover las carpetas contenedoras de las imágenes digite el comando “get -r /home/radiogis/Desktop/radioastronomia/imagenes/ /home/comdiguis/Desktop/CASIRI/CAMARA”

10. Para mover la carpeta contenedora de los timelapses digite el comando “get -r /home/radiogis/Desktop/radioastronomia/timelapses/ /home/comdiguis/Desktop/CASIRI/CAMARA”

Sistema Estación Meteorológica

1. Para iniciar la sincronización de la estación se debe presionar el boton “done” de la consola dos veces y mantenerlo presionado hasta que muestre las variables de medida.

2. Activar el entorno virtual Estacion_ambiental, para esto abra una terminal y ejecute el comando “source estacion_ambiental/bin/activate”.

3. Luego el comando “cd /home/comdiguis/Desktop/CASIRI/ESTACION” luego de esto el comando “python toma_estacion.py”. Luego de esto se le pedirá que coloque cuantas tomas desea que se realicen y el tiempo entre ellas (valor expresado en segundos).
4. En la ruta “/home/comdiguis/Desktop/CASIRI/ESTACION” se encuentra el archivo datos.csv en el cual queda almacenado el histórico de los datos tomados.

Nota (configuración inicial de la consola):

Entre cada ítem presionar “Done” y en el último ítem mantener Done presionado

- Receiving from: Debe aparecer una X que significa que está recibiendo datos.
- On (iss)
- Retransmit off
- Enter time: Configuración de fecha y hora
- Enter latitude: Configurar latitud
- Enter longitude: Configurar longitud
- Configurar zona horaria
- Daylight savings auto
- enter elevation: configurar altitud
- wind cup size
- rain collector
- rain season begins: fecha de inicio de lluvias
- serial baud rate: 19200

Universidad industrial de Santander

Manual de usuario (NOREC-23)

Enero de 2024

Carlos Eduardo Silva Sepúlveda

Brayan Hernando Gonzales Mendoza

Contenido

Introducción

La finalidad de esta guía es orientar a los usuarios a lo largo de una ruta lógica para el uso efectivo de la HMI (Interfaz Persona-Máquina) diseñada con el propósito de supervisar la estación de radioastronomía CASIRI. Se abordarán aspectos detallados que van desde la instalación de la copia de seguridad hasta la utilización de cada uno de los servicios que esta interfaz proporciona.

Capítulo 1

1. Acceso a la interfaz de programación

Para acceder a la interfaz de programación se requiere de acceso a internet y el conocimiento de la dirección Ip donde se desea instalar la imagen de la interfaz.
Una vez teniendo este dato, abra su navegador de preferencia y digite en el buscador la ip seguida del puerto 1880 de la siguiente manera.

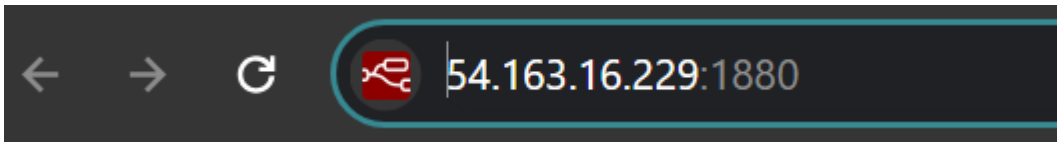


Ilustración 1. Ruta de acceso a la interfaz de programación.

Capítulo 2

2. Instalación de la copia de seguridad

Una vez dentro de la plataforma de desarrollo de Node-RED, diríjase a la pestaña de la parte superior derecha, y haga clic en la opción “import” como se observa en la ilustración 2.

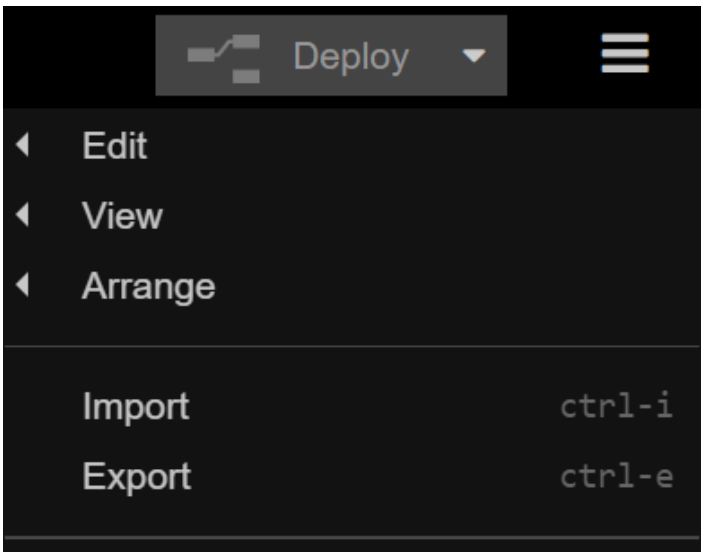


Ilustración 2. Apartado para importar la copia de seguridad.

Estando dentro, importe el archivo. Json que se anexo en los entregables del proyecto llamado “Copia_Seguridad.Json” el cual contiene la copia de seguridad de la interfaz del proyecto, como se observa en la ilustración 3.

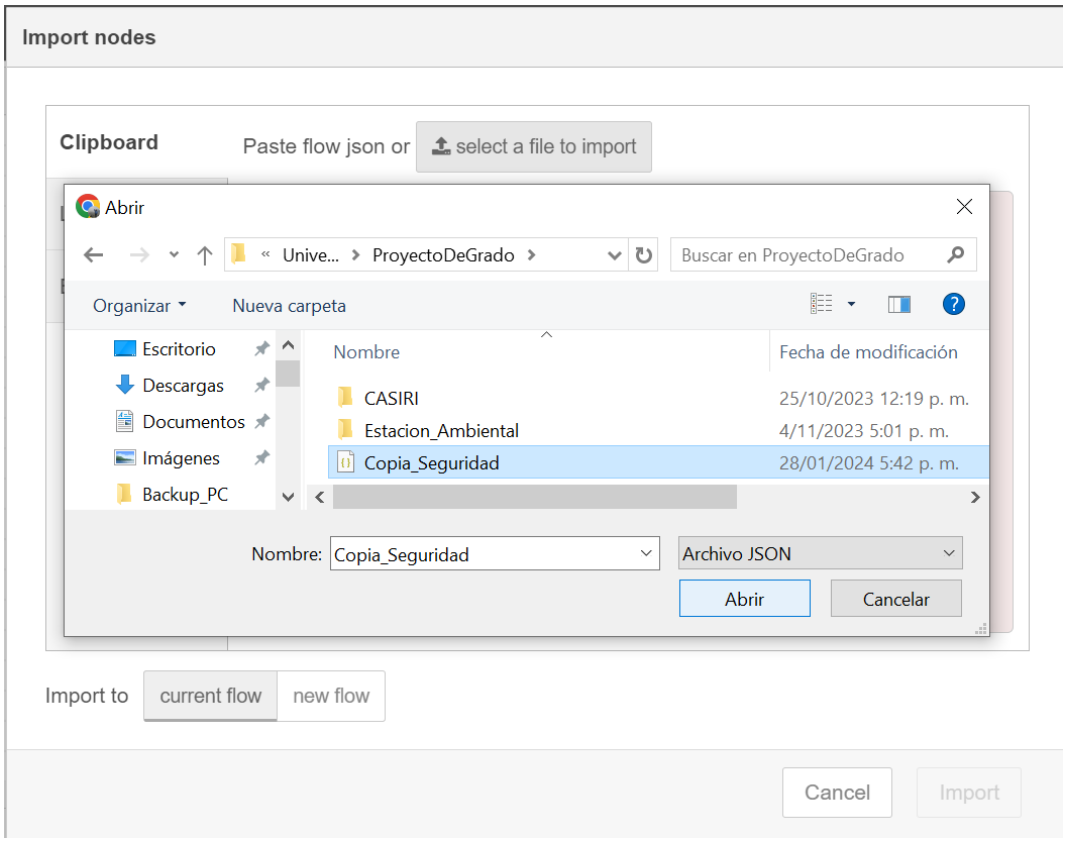


Ilustración 3. Importación del archivo “Copia_Seguridad.Json”

Capítulo 3

3. Acceso a la HMI (human-machine interface)

Habiendo instalado la copia de seguridad, ahora es posible acceder a la HMI. Para ello diríjase a la barra de navegación de su navegador de preferencia y digite la Ip publica de la maquina virtual, seguido del puerto 1880 y “/ui”, como se observa en la ilustración 4.

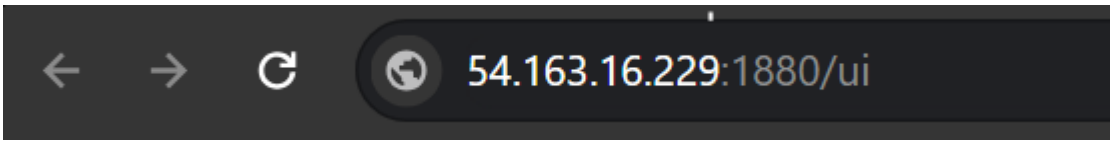


Ilustración 4. Ruta de acceso a la HMI.

Una vez dentro de la interfaz podrá ver una barra lateral en la parte izquierda de su pantalla como se observa en la ilustración 5, en esta ruta podrá navegar entre los diferentes servicios ofrecidos por la interfaz.

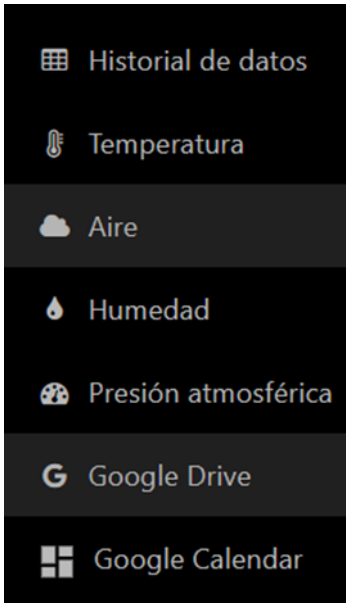
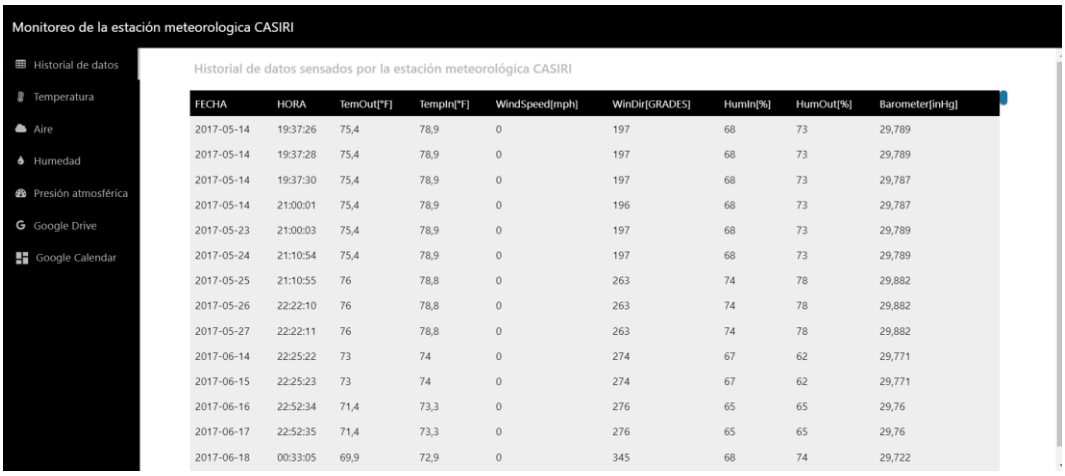


Ilustración 5. Menú de navegación de la HMI.

3.1 Servicio de consulta del historial de datos

Una vez dentro de la HMI podemos explorar todos los servicios, como lo puede ser el historial de datos, para acceder a él, diríjase a la sección “Historial de datos” de la barra lateral izquierda. En esta sección se podrá ver una tabla con el historial de datos como se muestra en la ilustración 6.



FECHA	HORA	TemOut[°F]	TempIn[°F]	WindSpeed[mph]	WinDir[GRADES]	HumIn[%]	HumOut[%]	Barometer[inHg]
2017-05-14	19:37:26	75,4	78,9	0	197	68	73	29,789
2017-05-14	19:37:28	75,4	78,9	0	197	68	73	29,789
2017-05-14	19:37:30	75,4	78,9	0	197	68	73	29,787
2017-05-14	21:00:01	75,4	78,9	0	196	68	73	29,787
2017-05-23	21:00:03	75,4	78,9	0	197	68	73	29,789
2017-05-24	21:10:54	75,4	78,9	0	197	68	73	29,789
2017-05-25	21:10:55	76	78,8	0	263	74	78	29,882
2017-05-26	22:22:10	76	78,8	0	263	74	78	29,882
2017-05-27	22:22:11	76	78,8	0	263	74	78	29,882
2017-06-14	22:25:22	73	74	0	274	67	62	29,771
2017-06-15	22:25:23	73	74	0	274	67	62	29,771
2017-06-16	22:52:34	71,4	73,3	0	276	65	65	29,76
2017-06-17	22:52:35	71,4	73,3	0	276	65	65	29,76
2017-06-18	00:33:05	69,9	72,9	0	345	68	74	29,722

Ilustración 6. Historial de datos.

3.2 Servicio de consulta de las gráficas del historial de datos y filtrado por fecha

Para hacer uso del servicio de gráficos del historial de datos y filtrado por fechas, diríjase a cualquier sección de las variables presentes en el HMI. Una vez dentro seleccione una fecha inicial de filtrado y otra final como se observa en la ilustración 7, luego de ello recargue la página web y podrá observar la gráfica del historial de datos como se observa en la figura 8, con un selector de muestras donde se observa el valor de la muestra con la fecha y hora en la que se tomó.

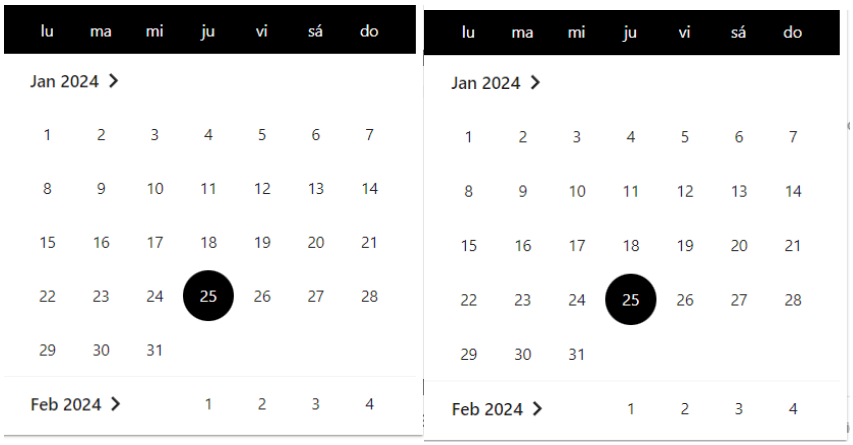


Ilustración 7. Selector de fecha inicial y final de la gráfica.

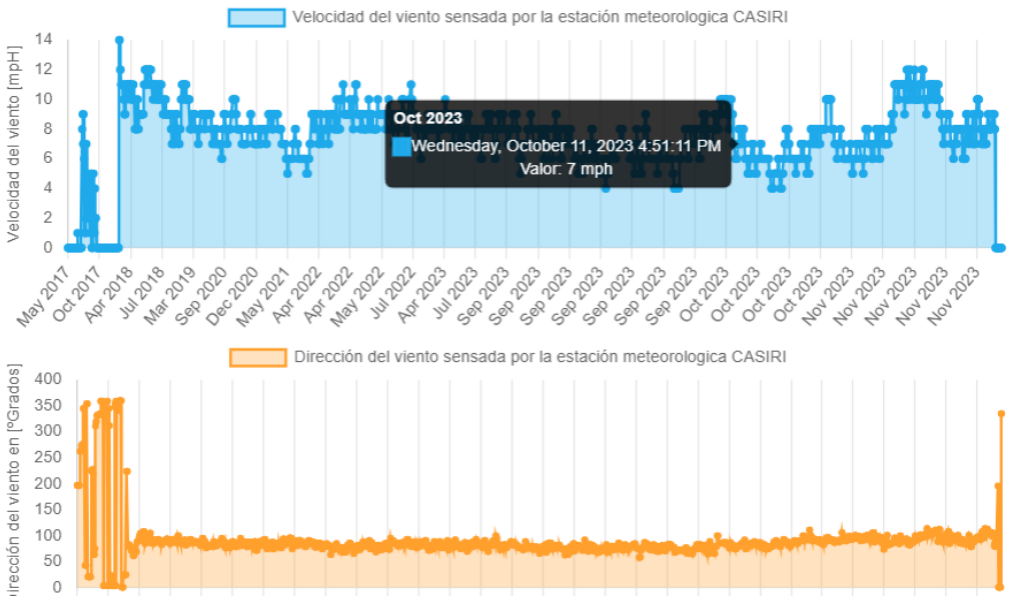


Ilustración 8. Graficas filtradas de acuerdo con los datos seleccionados en el selector de fechas.

3.3 Servicio de consulta en línea del último envió de datos

Para consultar el ultimo envío de datos que se registraron en la base de datos implementada con la API de Google Sheets, diríjase a cualquier sección de las variables medidas y observe en la parte derecha de la pantalla como hay un apartado especifico para esta opción, en la parte superior vera la hora y la fecha del ultimo envió de datos como se observa en la ilustración 9.

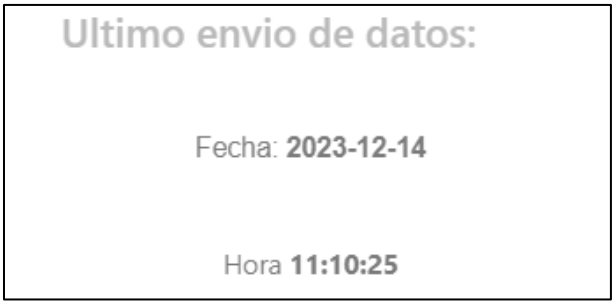


Ilustración 9. Fecha y hora del ultimo envío de datos

Para hacer uso de los manómetros haga clic en los selectores de unidades como se observa en la ilustración 10. Una vez seleccionadas las unidades debería ver los manómetros como se observan en la ilustración 11, en donde los colores de tonos azules pasteles indican una medición en un rango bajo, los tonos verdes pastel un rango optimo y los rojos en un rango alto.

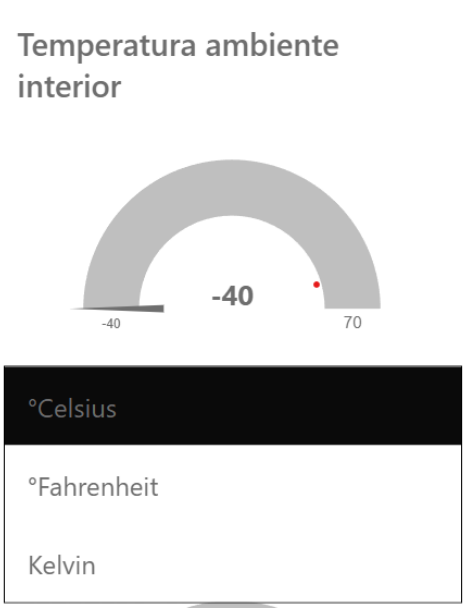


Ilustración 10. Selector de unidades.

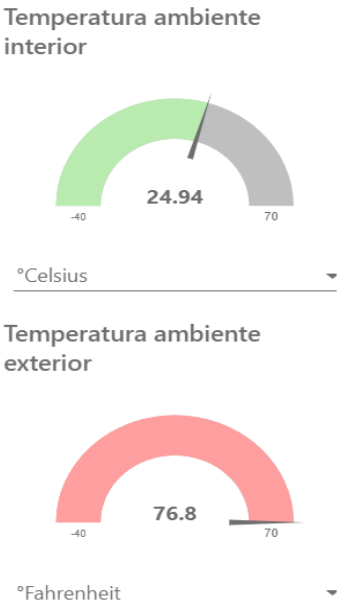


Ilustración 11. Manómetros en funcionamiento.

3.4 Apartado en Google Drive destinado a desarrolladores

Para acceder a el apartado que se creo en Google Drive para los desarrolladores, diríjase a la barra lateral izquierda y en el menú seleccione la opción “Google Drive”, allí vera una carpeta en Google Drive como se observa en la ilustración 12. En esta carpeta se encuentran archivos de interés para desarrolladores que estén interesados en implementar versiones futuras de la interfaz.

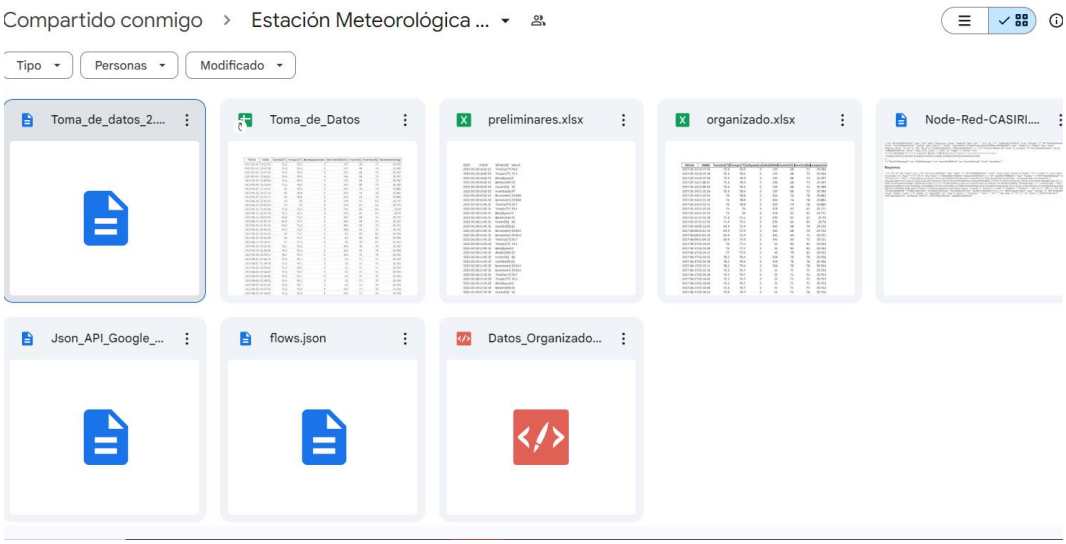


Ilustración 12. Apartado en Google Drive destinado a desarrolladores.

3.5 Servicio de programación de eventos con Google Calendar

Para hacer uso del servicio de programación de eventos con Google Calendar, diríjase a la barra lateral izquierda del panel de control del HMI y seleccione la opción “Google Calendar”, allí debería encontrar el Calendar destinado a la programación de eventos como se observa en la ilustración 13.

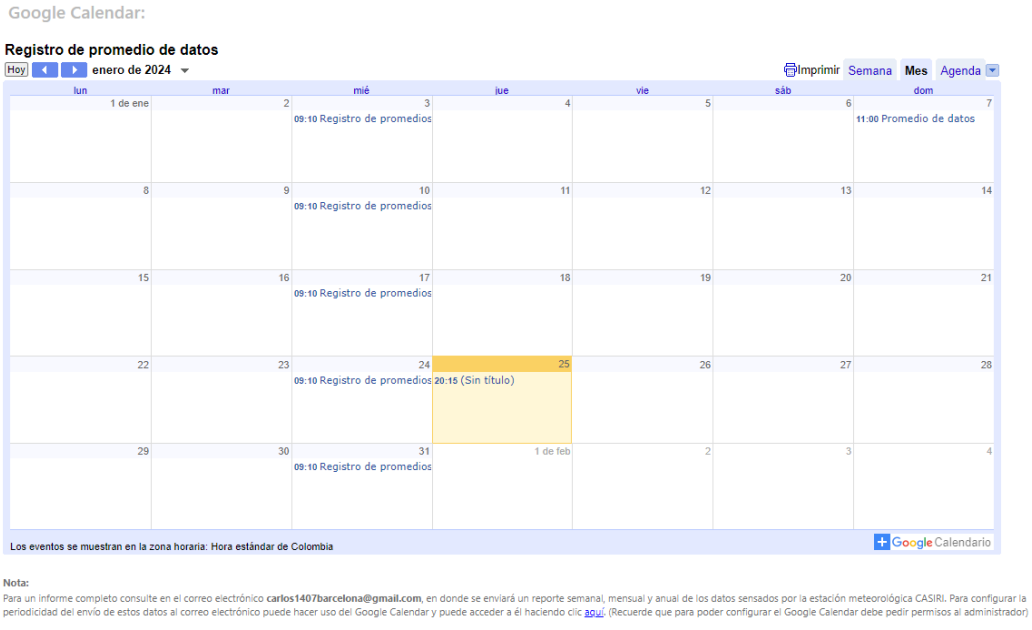


Ilustración 13. Google Calendar destinado a la programación de eventos.

Para hacer uso de este servicio diríjase al hipervínculo que se encuentra en la nota inferior del Google Calendar. Allí podrá programar eventos no inferiores a 5 minutos como se observa en la ilustración 14. Una vez programado el evento, espere a que el evento ocurra y revise la bandeja de entrada del correo electrónico registrado, allí debería haber un correo con los valores promedios calculados por la HMI en base a las muestras registradas por la estación de radioastronomía CASIRI.

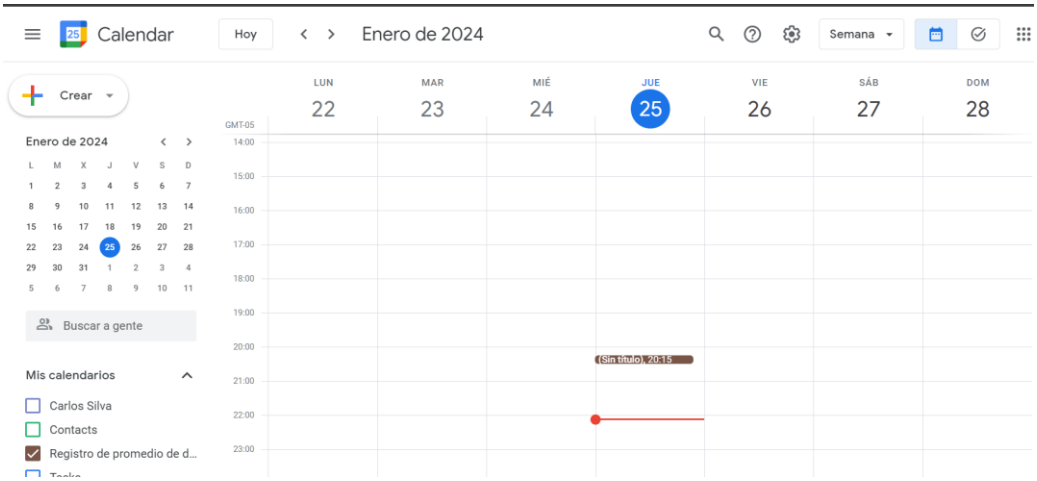


Ilustración 14. Programación de eventos en el Google Calendar.

REFERENCIAS

N. U. Khuzairi, S. Saon, A. K. Mahamad, M. S. M. Zainordin, S. Yamaguchi and M. A. Bin Ahmadon. (s/f). Weather Station Monitoring System with Node-RED,.

nodered.org. (s/f). *https://nodered.org/docs/user-guide/*.