UACM

Universidad Autónoma
de la Ciudad de México

NADA HUMANO ME ES AJENO

COLEGIO DE CIENCIA Y TECNOLOGÍA

LICENCIATURA EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS
Y DE TELECOMUNICACIONES

Sistemas de recolección de datos para IoT

T E S I S

QUE PARA OPTAR POR EL TÍTULO DE

LICENCIADOS EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS Y DE TELECOMUNICACIONES

P R E S E N T A N

CRISTHIAN GOMEZ CHAVEZ

JESICA PAOLA REGALADO ROBLERO

CARLOS FERNANDO SÁNCHEZ HERRERA

D I R E C T O R

M. EN I. OSCAR RENÉ VALDEZ CASILLAS

Ciudad de México, agosto de 2025.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

Agradecimientos

Hoy que concluimos esta etapa tan importante, no podemos dejar de mirar atrás y reconocer a todas las personas que caminaron junto a nosotros. A nuestras familias, gracias por ser nuestra fuerza cuando sentimos que ya no podíamos más, por sus palabras de aliento, su paciencia infinita y su amor que nos sostuvo en cada paso.

A nuestro asesor de tesis, Oscar René Valdez Casillas gracias por su tiempo, por compartir sus conocimientos y por guiarnos con paciencia y compromiso. Agradecemos profundamente a los miembros del comité: Dra. Kaur Tejinder, Mtro. Luis René Sagredo Hernández y Lic. Jorge Mendoza Zavala, cuyas observaciones críticas y enriquecedoras han elevado la calidad de este proyecto.

Nuestra gratitud también al Mtro. Víctor Manuel Macías Medrano, quien nos ofreció su apoyo y motivación para concluir este trabajo, además de brindarnos la oportunidad de utilizar el laboratorio de electrónica libre, recurso esencial para avanzar en nuestra investigación

De igual manera a el laboratorio de telecomunicaciones por apoyarnos con materiales y equipo que nos ayudaron a enriquecer nuestro aprendizaje con mayor facilidad y terminar la carrera. Gracias a nuestros compañeros y amigos por su entusiasmo, apoyo emocional y por esas conversaciones inspiradoras que moldearon parte de nuestras ideas sabiendo que no estamos solos en este camino

Planteamiento del problema.

En la actualidad, el Internet de las Cosas (IoT) se ha convertido en un soporte fundamental para la transformación digital en sectores como la agricultura, salud, transporte, industria, hogares y ciudades inteligentes. Esta tecnología permite la interconexión de dispositivos y la recolección de datos en tiempo real, lo que facilita la toma de decisiones y mejora el funcionamiento operativo de múltiples sistemas.

En el plantel educativo donde se desarrolla esta investigación, se ha identificado que varios espacios físicos presentan condiciones poco favorables, como áreas reducidas y escasa ventilación, lo que podría afectar la calidad del ambiente y, en consecuencia, el bienestar de los estudiantes y docentes. A pesar de esta situación, actualmente no se cuenta con un sistema que permita monitorear en tiempo real las condiciones ambientales, como temperatura, humedad o calidad del aire, lo que impide detectar situaciones de riesgo o implementar medidas correctivas de manera oportuna.

La ausencia de un sistema eficiente de recolección de datos ambientales mediante IoT representa un problema concreto, ya que limita la capacidad del plantel para gestionar de manera proactiva el entorno físico. Además, la instalación de sistemas IoT en este tipo de espacios enfrenta desafíos relacionados con la conectividad, la ubicación estratégica de sensores y el procesamiento de los datos recolectados.

Por lo tanto, se hace necesaria la implementación de un sistema de recolección de datos para IoT que permita monitorear continuamente las condiciones ambientales del plantel, brindando información útil para la toma de decisiones orientadas a mejorar la seguridad, el confort y el rendimiento académico en ambientes escolares.

Justificación.

Este trabajo se centra en la implementación de un sistema IoT, explorando dos enfoques distintos: el primero, a través de soluciones comerciales, que simplifican la configuración y gestión de datos, gracias a su interfaz intuitiva y herramientas integradas.

El segundo enfoque se basa en el uso de software libre, Este enfoque requiere una configuración manual y personalizada, pero ofrece mayores ventajas. A lo largo de este trabajo, se analizarán las ventajas y desventajas de cada enfoque. Así como su aplicación y adaptación, con la intención de ofrecer una visión general sobre la implementación de sistemas IoT que se adapten a diferentes necesidades y circunstancias. En este sentido, se busca no solo entender las capacidades de IoT, sino también explorar cómo estas soluciones pueden ser implementadas de una mejor manera a través de diversas aplicaciones.

Objetivos.

- Generales. Realizar la comparación de un sistema de recolección de datos comercial y un sistema de recolección de datos basado en software libre.

- Particulares.
    I. Conocer el funcionamiento de los sistemas de recolección de datos y las partes que lo conforman.
    II. Selección de sensores y dispositivos adecuados para la medición de variables físicas como Temperatura, Humedad relativa e Intensidad lumínica.
    III. Implementar una arquitectura de comunicación adecuada entre los sensores y la plataforma central de procesamiento, utilizando protocolos apropiados como MQTT.
    IV. Selección de la plataforma de almacenamiento de datos que permita organizar y visualizar la información recolectada.
    V. Programar e integrar microcontroladores o nodos IoT para la captura y envío de datos en tiempo real.
    VI. Validar el funcionamiento del sistema.

Metodología.

I. **D**iseño del sistema.

- Seleccionar las herramientas a utilizar. Definir el software que será utilizado para la implementación del sistema basado en software libre, así como la plataforma comercial que será utilizada.
- Selección de las variables físicas a monitorear. En base a las variables se definirán los sensores y su integración con los nodos que serán

utilizados para recabar la información que será almacenada y desplegada por medio del tablero de presentación de datos.

II. **I**nstalación y configuración.

- Preparación del entorno. Configurar los nodos y solicitar el acceso a la infraestructura necesaria, sea virtual o física.
- Instalador del broker. Seleccionar el software que hará las funciones de broker, realizar su instalación y configuración.
- Instalación de la base de datos. Seleccionar el software manejador de bases de datos temporales y realizar la instalación.
- Instalación del tablero de presentación de datos. Seleccionar, instalar y configurar el software de presentación de datos.
- Investigación y configuración de la plataforma comercial. Selección de la plataforma comercial a utilizar para realizar su configuración y obtención de credenciales para su uso.

III. **P**ruebas y evaluación.

- Prueba de funcionamiento de los nodos. Programar los nodos y verificar que los sensores den las lecturas adecuadas.
- Pruebas de conectividad. Configurar los nodos para su conexión al sistema de recolección de datos y verificar que se mantenga activa.
- Configuración y prueba del tablero de presentación de datos. Definir las partes de los tableros que presentaran los datos, tanto la solución basada en software libre como en la plataforma comercial.
- Análisis de resultados. Analizar los resultados de las pruebas para identificar los posibles errores que se presenten y realizar su corrección.

IV. **R**esultados.

Mostar los resultados, destacando el funcionamiento de ambas plataformas, sus pros y contras de cada una.

Contenido

Índice de tablas.

Índice de figuras

Capítulo 1. Introducción.

1. Antecedentes.

El avance que se tiene constante en las tecnologías digitales como el Internet de las Cosas, es un concepto que combina dispositivos físicos con capacidades de comunicación, procesamiento y almacenamiento, estos han cambiado de manera sorprendente. En este capítulo se presenta una visión general del Internet de las cosas, mencionando su evolución histórica, la arquitectura que utiliza, los elementos que conforman esta tecnología, los principales protocolos de comunicación que permiten la interoperabilidad entre dispositivos. Además, se analiza la base de datos que se utilizara para la información que almacene, así como otros aspectos clave que permiten comprender el funcionamiento integral del Internet de las Cosas.

1.1 Internet de las cosas IoT.

La tecnología Internet of Things (IoT) se caracteriza por la conexión entre sistemas físicos con el internet, esto con la finalidad de interpretación y análisis de datos que son enviados por sensores, softwares etc. Esto se logra con la ayuda de una base datos, donde los datos son guardados temporalmente.

Un aspecto importante a tener en cuenta sobre el Internet de las Cosas es que no todo dispositivo conectado a internet forma parte de este concepto. La simple conexión a la red no es suficiente; para que un dispositivo sea considerado parte del IoT, debe estar diseñado para interactuar con otros sistemas o dispositivos, recolectar datos y ofrecer un beneficio funcional, generalmente sin intervención humana directa. Por ejemplo, una computadora personal está conectada a internet, pero no se considera parte del IoT porque su propósito principal es la

interacción directa con el usuario. En cambio, un sistema de alarma doméstico conectado a una central policial sí forma parte del IoT, ya que opera de forma automatizada y con una finalidad específica orientada a la seguridad.

Por otra parte, se cuenta con una interpretación sobre lo que es esta tecnología se define como “Una red de objetos físicos conectados a través de internet, los cuales logran interactuar entre sí, redes de comunicación, mecanismos de computación de respaldo y aplicaciones típicamente en la nube.” [1]

En teoría, la IoT es una tecnología que se puede conectar entre sí con varios dispositivos y ofrece varios beneficios.

Esta tecnología es reciente por lo que nos menciona que el término “Internet de las Cosas” Fue definido en 2009 cuando Kevin Ashton, profesor del MIT en aquel entonces, usó la expresión Internet of Things (IoT) de forma pública por primera vez, y desde entonces el crecimiento y la expectación alrededor del término ha ido en aumento de forma exponencial. [2]

La idea de IoT de Aston estaba centrada en el uso de la identificación por radiofrecuencia (RFID) con el fin de interconectar dispositivos entre sí. Por supuesto, el concepto de Ashton de un IoT basado en RFID no era sorprendente en ese momento. En 1999, las redes inalámbricas tal como las conocemos hoy todavía estaban en su infancia y las redes celulares aún no habían cambiado a una configuración totalmente basada en una dirección IP.

Actualmente tiene un gran impacto en la sociedad, pero para lograr esta tecnología sucedieron acontecimientos muy relevantes los cuales ayudaron a

formar las bases para lograr el desarrollo de esta tecnología, menciona que “La revolución en el área tecnológica durante las últimas décadas continuara redefiniendo de forma sustancial la manera en la que se entiende la sociedad por lo que el día a día de las personas ha sido testigo de un cambio importante en usos y costumbres, altamente influenciado por nuevos dispositivos y formas de comunicación.” [2]

1.2. Elementos que conforman un sistema de Internet de las cosas.

Esta tecnología está conformada por dispositivos eléctricos y electrónicos los cuales son mencionados como “sensores, actuadores, placas de prototipo, etc. Además de tecnologías de red, protocolos de comunicación, plataformasIoT para el tratamiento inteligente de datos (BigData o Small Data) y aplicaciones de usuario.” [2]

Cada parte de esta tecnología lleva a cabo tareas en específico por lo que las partes que conforman la (IoT) son:

I. Los sensores y dispositivos.

II. Conectividad.

III. La plataforma IoT.

IV. Microprocesador.

V. Protocolos de comunicación. [2]

En la tabla 1 se describe cada una de estas partes.

Tabla 1 Partes que conforman la IoT, autoría propia basada en M.C. Vega [2]

Sensores Dispositivos	**Conectividad**	**Procesamiento de datos**	**Interfaz de usuario**
Estos elementos son los que recopilan los datos, un ejemplo puede ser las cámaras de vigilancia que perciben cualquier movimiento y envían información (datos) mediante un teléfono móvil o comunicación inalámbrica.	Los datos de los sensores o dispositivos se envían a la nube mediante un almacenamiento de datos ya sea por conexión Wi-fi, Bluetooth, satélite, entre otros.	Los datos enviados a la nube son procesados y de acuerdo a un algoritmo, se determina la necesidad de realizar una intervención, ya sea mediante una alarma o algún aviso.	Es la parte donde el usuario puede interactuar y visualizar la información que es recolectada.

1.2.1. Protocolos de comunicación de red.

Un protocolo de red es un estándar de comunicaciones, contiene las reglas necesarias y la información sobre cómo las computadoras intercambian datos entre sí.

La sección de conectividad indicada en las partes que conforman la IoT del apartado anterior, involucra una red de dispositivos interconectados. La comunicación se puede modelar a través del modelo de referencia Open Systems Interconnection por sus siglas en inglés (OSI) y a su vez los protocolos utilizados

se pueden ubicar en la suite de protocolos Transmission Control Protocol/Internet Protocol (TCP/IP).

Para definir la comunicación a través de la red se ha desarrollado un modelo de referencia que permite identificar cada una de las actividades necesarias para realizar la comunicación de un punto a otro conectado a la red. Este modelo es conocido como OSI.

El modelo OSI se divide en 7 capas, cada una de ellas dependen una de la otra y realizan tareas en específico. La información se va transmitiéndose de capa en capa añadiéndole información específica la cual, una vez que llega a la interfaz del receptor, es eliminada. [3]

El modelo OSI provee un conjunto detallado de estándares para desarrollar protocolos que describen a una red. Estas capas se describen a continuación para entender el proceso que realizan.

- **Física**: se encarga de establecer e interrumpir la conexión y supervisa su funcionamiento durante la transmisión de información.
- **Enlace**: Transforma un canal de comunicaciones en un canal libre de errores entre los dos límites de en enlace físico. Provee inicialización del flujo, control de flujo terminación del vínculo y control de errores.
- **Red**: Controla la operación sobre la subred de comunicaciones, provee servicios de ruteo, transferencia sobre la red y administración de la red.

- **Transporte**: Provee un canal para enviar mensajes entre dos procesos que se comunican.
- **Sesión**: Organiza, sincroniza el intercambio de mensajes y controla el proceso de la comunicación.
- **Presentación**: Estructura los mensajes, provee semántica y sintaxis. Define los formatos de transmisión de datos.
- **Aplicación**: Provee servicios y procedimientos para las aplicaciones del usuario.

La diferencia entre el modelo OSI y el modelo de TCP/IP es que el modelo OSI se encarga de dar la descripción de cada una de las capas, así como la teoría de cómo funciona la red, pero el que se implementa es el modelo de TCP/IP.

Se debe incluir una comparación entre el modelo de referencia OSI y la suite de protocolos TCP/IP, señalando qué capas son equivalentes en ambos modelos y cuáles son agrupadas en la arquitectura TCP/IP. Específicamente, se debe destacar que las capas de Aplicación, Presentación y Sesión del modelo OSI se integran en una sola capa de Aplicación en TCP/IP. Asimismo, las capas de Enlace de Datos y Física del modelo OSI se representan conjuntamente como la capa de Acceso a la red en TCP/IP. Las capas de Red y Transporte, por su parte, tienen correspondencias directas en ambos modelos.

Los dispositivos de IoT se comunican mediante protocolos que garantizan que la información de un dispositivo sea leída y comprendida por otro. Dada la diversidad de dispositivos de IoT disponibles, es importante utilizar el protocolo adecuado en el contexto adecuado.

Cada una de las partes que conforman esta tecnología están ubicadas dentro del modelo de TCP/IP.

1.2.2 Arquitectura de un sistema de Internet de las cosas.

Un punto importante para que la implementación de la IoT sea exitosa, es su arquitectura, ver Figura 1. Se puede encontrar diferentes modos dependiendo de los diferentes usos que se les den a estos dispositivos. Es importante mencionar que no importa la arquitectura que se tome, esta debe ser capaz de soportar la incorporación de distintas funcionalidades.

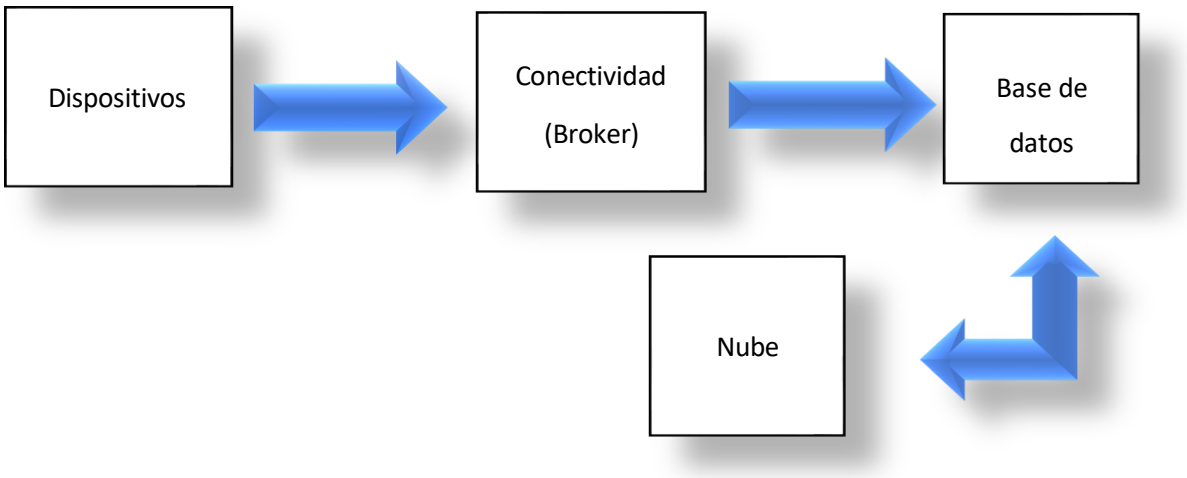

Figura 1 Arquitectura para un sistema IoT, autoría propia.

Esta arquitectura se puede realizar con ciertos elementos los cuales son:

- **Dispositivos**: Son las placas, sensores entre otros.
- **Broker**: Es el encargado de transmitir los mensajes, para que sean almacenados en una base de datos.
- **Base de datos**: Útil para la recopilación de datos que se obtengan de los dispositivos.

- **Nube**: Donde se guardan todos los datos obtenidos. En específico son servidores de almacenamiento de datos.

1.2.2.1. Descripción de la arquitectura general.

De manera general, se menciona una arquitectura de tres capas, y de igual manera funciona para este trabajo.

- **Capa de Percepción**: Esta capa representa a los dispositivos físicos de la IoT (sensores y actuadores): “Esta capa es la responsable de la identificación de los dispositivos, la recopilación de datos del entorno por parte de los sensores y la transformación de datos en señales digitales. Esta capa también permite ejecutar acciones a través de los actuadores”. [4]
- **Capa de Red**: Esta capa también se utiliza para transmitir y procesar los datos de los sensores, “La capa de red es responsable de conectase a otros dispositivos físicos, dispositivos de red y servidores.” [4]
- **Capa de aplicación**: “En esta capa se encuentran las aplicaciones y servicios de usuario, que hacen un uso efectivo de la información procesada de los sensores. Adicionalmente, esta capa permite compartir datos con otras aplicaciones, servicios, sistemas y plataformas.” Algunos ejemplos donde observas son; transporte móvil, ciudad automatizada, casas inteligentes, etc. [4]

1.2.2.2 Protocolos de red.

Los protocolos de red son un conjunto de reglas y estándares los cuales establecen la forma en que los dispositivos se comunican y comparten la información entre ellos.

Siguiendo la estructura de la suite de protocolos TCP/IP se encuentran con los siguientes protocolos involucrados.

- **Tecnología de capa Física**: Comunicación a través de redes cableadas (Ethernet, IEEE 802.3) o inalámbricas (Wifi, Red Celular, ZigBee).
- **Protocolo de Red**: IP por sus siglas en inglés, (Internet Protocol). Permite la organización lógica de la red. Se tienen dos versiones, IPv4 e IPv6.
- **Protocolo de Transporte**: TCP por sus siglas en inglés, (Transmission Control Protocol) utilizado para establecer una comunicación confiable entre ambos extremos, orientado a conexión.
- **Protocolo de Aplicación**: MQTT por sus siglas en inglés, (MQ Telemetry Transport) que permite la transmisión de información ligero que puede llevar la información recabada por la Capa de Percepción del sistema IoT.

En la Tabla 2 se muestra un resumen de los protocolos involucrados. Ubicación en el modelo OSI y TCP/IP

Tabla 2 Ubicación en el modelo OSI y TCP/IP.

Capa(s) TCP/IP	**Capa(s) OSI**	**Protocolo utilizado**
aplicación	aplicación presentación sesión	MQTT, AMQP, CoAP
Transporte	Transporte	TCP, UDP
Red.	Red.	IPv6, IPv4

Física	Enlace Física	LPWAN, Zigbee, Ethernet, Bluetooth low energy (BLE), Wi-Fi

basada en: https://www.internetsociety.org/wp-content/uploads/2017/09/report-InternetOfThings20160817-es-1.pdf

1.2.2.3. Base de datos de series temporales.

Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Normalmente, una base de datos está controlada por un sistema de gestión de bases de datos (DBMS). [5]

Por este motivo una base de datos debe presentar los datos en una forma que el usuario pueda interpretarlos y modificarlos.

Una Base de datos temporal es un sistema de gestión de base de datos (DBMS) el cual implementa y trata con especial énfasis aspectos temporales, teniendo un modelo de datos y una versión temporal del lenguaje de consulta estructurado. [5]

Las bases de datos temporales son bases de datos históricas, además de transaccionales.

I. No hay datos sino datos temporales.

II. Los datos tienen marcas de tiempo.

III. Almacenan datos actuales y datos históricos.

Las bases de datos temporales se caracterizan por el manejo del tiempo de manera total, es decir; pasado, presente y futuro, o parcial (pasado y presente), en pocas palabras se puede decir que una base de datos temporales contiene datos históricos y datos actuales.

También incluyen tiempo de validez y tiempo de transacción, las combinaciones de estas dos formas generan un dato bitemporal. El tiempo de validez indica un intervalo de tiempo en el cual un hecho es verdad en el mundo real. El tiempo de transacción indica el periodo de tiempo en el cual un hecho está guardado en la base de datos. [6]

1.2.2.4 Tablero de presentación de datos.

Un tablero de presentación de datos o dashboards es una interfaz de usuario que puede presentar algo de semejanza con el panel de control de un coche, donde se organiza y se presenta la información de una manera más fácil de leer.

Un tablero de presentación de datos para IoT es la interfaz de usuario dentro de una plataforma de IoT que permite a los usuarios monitorear e interactuar con dispositivos conectados a través de gráficos, cuadros y otras herramientas al igual que los elementos de la interfaz de usuario.

Los paneles, permiten administrar todos los aspectos de los dispositivos conectados, así como obtener una perspectiva de su entorno a través de la visualización de los datos de los dispositivos. [7]

En pocas palabras, un tablero de datos es una interface gráfica de usuario por sus siglas en inglés, Guide User Interface (GUI) se puede asemejar a un tablero de instrumentos.

Algunas de las características que se pueden encontrar de un tablero de datos son:

I. Todas las visualizaciones entran en una sola pantalla.

II. Muestra los indicadores de desempeño o medidas de rendimiento más importante a controlar.

III. La interactividad, como el filtrado, debe ser fácil de emplear en un dashboards.

IV. Deben ser fáciles de entender y utilizar por todos.

V. Los datos que se muestran deben actualizarse automáticamente sin ninguna ayuda por parte del usuario.

VI. Los paneles de control más eficaces tienen datos actualizados, al menos sobre una base diaria.

1.3. Requerimientos de un nodo de Internet de las cosas.

Los sistemas electrónicos que utiliza la IoT se encargan de la recopilación, interpretación y realización de los procesos, de un sistema electrónico es aquel componente que se utiliza para controlar el transporte y la distribución de energía, este sistema se interconecta entre sí para realizar una función en específico. [8] Los sistemas electrónicos son conjuntos de circuitos que operan con señales eléctricas y se procesan para ejecutar una determinada función. Constan de una etapa de entrada, en la que se recogen datos del exterior, como: (luz, humedad, movimiento, pulsación en un teclado, temperatura, entre otros).

1.3.1. Consumo de energía.

La energía consumida en un circuito eléctrico es una métrica que determina cuánta energía se utiliza en un período determinado.

Es importante tener en cuenta factores como la eficiencia de los dispositivos en el circuito, posibles pérdidas de energía y la naturaleza del suministro de energía; por ejemplo, si es constante o variable. Estos elementos pueden afectar la precisión del cálculo de la energía consumida y requieren métodos más avanzados de análisis y medición. [9]

Los dispositivos IoT deben cumplir estrictos requisitos de consumo energético para poder alcanzar la duración esperada de la batería de varios años sin necesidad de conectarlos a una fuente de alimentación. Para verificar el consumo y la duración prevista de la batería es necesario medir con absoluta precisión el voltaje y la corriente en diferentes estados operativos del dispositivo bajo prueba.

Consumo por envío y recepción de datos, representa la energía consumida por una acción de envío y recepción a la plataforma. Se debe tener en cuenta:

- El consumo del procesado de los datos.
- Ineficiencias intrínsecas de la electrónica.
- El envío de los datos.
- La recepción de datos asociada a un envío.

Además, se debe considerar el consumo de lectura del sensor y tareas adicionales.

- **Lectura del sensor**: Como el equipo se encuentra en modo de reposo la mayor parte del tiempo, es necesario activar un conjunto de elementos para proceder a obtener la medida del sensor.

- **Tareas adicionales**: Aquí se agrupan el conjunto de tareas de mantenimiento del sistema que se realizan periódicamente, en ellas se incluyen supervisión de errores y gestión de temporización del sistema.

1.3.2 Adquisición de datos.

Los sensores en la IoT son los principales elementos, encargados de recibir la información. En [10] los sensores se definen como: Dispositivo que detecta una determinada acción externa, temperatura, presión, etc., y la transmite adecuadamente.

Como ya se había mencionado anteriormente, los sensores son los encargados de obtener los datos solicitados y transformarlos en señales de tal manera que el sistema pueda interpretarlos. En este sentido Bell [8], define a los sensores como: Dispositivos que generan voltaje con base a una propiedad mecánica o química, es decir que muestra alguna variable física y entrega una señal de voltaje proporcional.

1.3.3 Procesamiento de datos.

Los datos se refieren a hechos en bruto y desorganizados, y generalmente tienen un valor limitado hasta que se procesan, una vez procesados los datos, se llaman información (Figura 2).

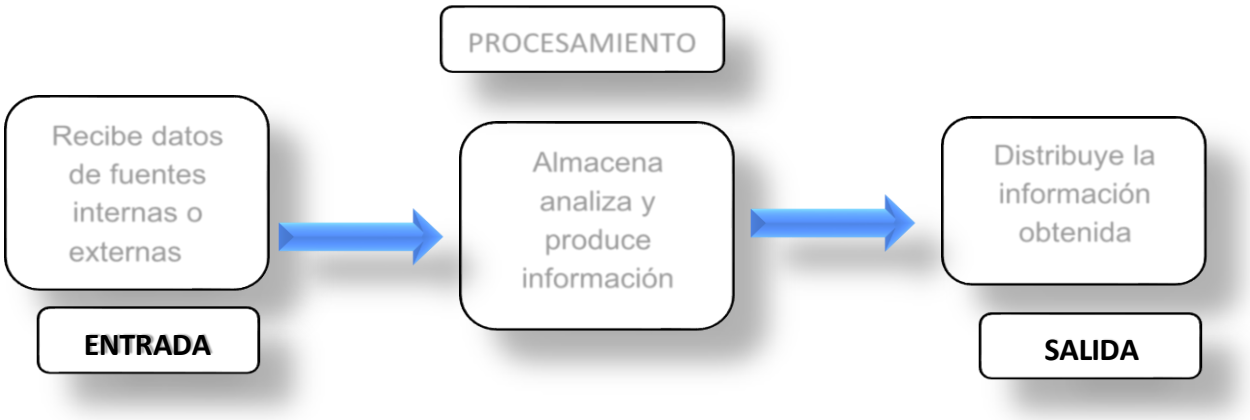

Figura 2, Bloques de procesos de la información, Autoría propia

El procesamiento de información consiste principalmente en tres etapas.

- **Entrada**: Los datos recopilados se convierten en una forma legible para la computadora, mediante una etapa de acondicionamiento. Si no son reconocibles, no se pueden procesar.
- **Procesamiento**: La computadora transforma los datos en bruto en información. La transformación se lleva a cabo utilizando diferentes técnicas de manipulación de datos, como la clasificación en diferentes grupos, la ordenación en función de diversos criterios o el cálculo, operaciones aritméticas y lógicas que se pueden aplicar a datos numéricos.
- **Salida**: Los datos procesados se convierten en una forma legible para los humanos y se presentan al usuario final como información útil [11].

Un nodo de red representa un punto de conexión dentro de una red de comunicaciones. Este punto, comúnmente denominado punto final, tiene la capacidad de conectarse a la red para enviar y recibir información. Además, puede

generar y reenviar datos a través de diferentes rutas dentro de la red. Algunos ejemplos de nodos finales incluyen dispositivos como computadoras, teléfonos inteligentes y impresoras. [38]

No obstante, no todos los nodos cumplen la función de punto final. Existen nodos intermedios cuya función principal no es originar ni recibir datos, sino dirigirlos hacia su destino correspondiente. Entre estos se encuentran los enrutadores y los conmutadores. Asimismo, hay nodos centralizados, denominados servidores, que se encargan de almacenar información y brindar servicios o recursos a otros dispositivos conectados en la red.

Para el procesamiento de las señales recibidas por los sensores y su posterior transmisión vía la red, cada nodo debe incluir ya sea un microcontrolador o un microprocesador.

El microcontrolador se define como “Un dispositivo que actúa bajo el control del programa almacenado de la memoria. La CPU se ocupa básicamente de traer las instrucciones del programa desde la memoria, interpretarlas y hacer que se ejecuten.”

Comparando la definición anterior se puede observar que se tiene un concepto similar por lo que se define como. “El controlador es el “cerebro” del sistema, el cual se encarga de recolectar la información que proveen los sensores y enviar una respuesta en forma de órdenes para los actuadores y estos pueden ser programables” [12]

Por su parte, los microprocesadores tienen características diferentes. Un microprocesador es un componente electrónico que incorpora las funciones típicas de una computadora. [13]

También se menciona que “el microprocesador es el cerebro y actúa bajo el control del programa almacenado en la memoria, este hace la función de interpretarlas y hacer que se ejecuten.” [14]

1.3.4 Comunicación a internet.

Aunque los sistemas de IoT suelen tener arquitecturas diferentes, la mayoría incluyen los componentes, que se muestran en la tabla 3:

Tabla 3 Cuadro de comunicación a internet, Autoría propia.

Dispositivo IoT	Todo tipo de sensor e interface de conexión con el medio físico o tecnología de conexión de red.
Comunicaciones locales	El método que utiliza el dispositivo para comunicarse con los dispositivos vecinos.
Protocolo de aplicación	El marco que define cómo se transporta el contenido de la información.
Pasarelas	Traducen y retransmiten la información, normalmente enlazando las redes de dispositivos locales con Internet.
Servidores de red	Sistemas que gestionan la aceptación y transmisión de los datos de IoT , normalmente ubicados dentro de los centros de datos en la nube.

Aplicaciones en la nube	Procesan los datos de IoT para convertirlos en información útil y presentarlos a los usuarios.
Interfaz de usuario	Donde los usuarios ven la información de IoT , la manipulan y emiten órdenes a los dispositivos de IoT.

De acuerdo con la tabla anterior, se requiere de una conexión constante hacía Internet. Algunas de las maneras de mantener los nodos en lugares distantes sin tener que llevar hasta ellos un cableado es por medio de las señales de radio frecuencia, por lo que se suelen utilizar diferentes tecnologías. Entre ellas, la conexión a la red celular y la transmisión vía red inalámbrica local o WiFi.

La comunicación a internet es elemental en esta tecnología ya que esto hace que funcionen de manera eficaz para poder interactuar con otros dispositivos como por ejemplo, enviar y recibir datos, así como aprovechar los servicios y recursos que se encuentran en la nube.

1.3.4.1 Uso de la red celular.

Una red celular o red móvil es una red de comunicación en la que el último enlace es inalámbrico. Las redes celulares ofrecen una serie de características deseables, como:

- Más capacidad capacidad de manejar más usuarios o conexiones simultáneamente que un solo transmisor de gran tamaño, ya que la misma frecuencia puede utilizarse para múltiples enlaces siempre y cuando estén en celdas diferentes.
- Los dispositivos móviles utilizan menos energía que con un solo transmisor o un satélite, ya que las torres de telefonía móvil están más cerca.

- Zona de cobertura más amplia que la de un solo transmisor terrestre, ya que pueden añadirse torres de comunicaciones adicionales indefinidamente y no están limitadas por el horizonte.

Los principales proveedores de telecomunicaciones han desplegado redes celulares de voz y datos en la mayor parte del mundo. Esto permite que los teléfonos móviles y los dispositivos informáticos móviles se conecten a la red telefónica pública conmutada y a Internet pública [15].

Si bien la tecnología celular es capaz de enviar grandes cantidades de datos, el gasto y consumo de energía pueden ser demasiado altos para muchas aplicaciones. Aunque puede ser ideal para proyectos de datos de bajo ancho de banda basados en sensores que enviarán muy pocas cantidades de datos a través de Internet. En los últimos años, las operadoras de redes celulares están desplegando soluciones orientadas a IoT, que se basan en utilizar bandas de frecuencia de guarda, con poco ancho de banda y un bajo consumo de energía [16]. Las redes celulares pueden ahora acomodar fácilmente los dispositivos de IoT.

Otros protocolos como el 4G LTE y el 5G requieren mucha más potencia, pero también pueden manejar datos más pesados como el vídeo digital.

Actualmente, se utiliza la tecnología SIM y eSIM para realizar la conexión a la red celular. Una SIM card (o tarjeta SIM, por sus siglas en inglés de Subscriber Identity Module) es un pequeño chip que se inserta en un teléfono móvil o en otros dispositivos de comunicación. Su función principal es almacenar información importante como:

- **Datos del usuario**: Incluye el número de identificación de la SIM.
- **Autenticación**: Permite a la red móvil identificar al usuario y autenticar el dispositivo.
- **Configuración de red**: Facilita la conexión del dispositivo con la red del proveedor de servicios móviles.

La eSIM MFF2 es una versión de la eSIM, que se integra directamente en el dispositivo, eliminando la necesidad de una tarjeta SIM física.

- **eSIM (embedded SIM)**: Es un chip integrado en el dispositivo que reemplaza a la tarjeta SIM física. Permite a los usuarios cambiar de operador o gestionar múltiples planes de datos sin tener que cambiar una tarjeta física.
- **MFF2 (Machine-to-Machine Form Factor 2)**: Es una especificación para el tamaño y la forma del chip eSIM. El MFF2 es una versión más pequeña y delgada del eSIM, diseñada para ser utilizada en dispositivos que requieren un factor de forma compacto, como teléfonos inteligentes, relojes inteligentes, dispositivos de Internet de las Cosas (IoT), y otros gadgets donde el espacio es limitado [16].

1.3.4.2 Uso de la red inalámbrica.

Una red inalámbrica permite que los dispositivos permanezcan conectados a la red, pero sin usar cables. Los puntos de acceso amplifican las señales de Wi-Fi, de manera que un dispositivo puede estar lejos de un router, pero permanecer conectado a la red [17].

Las ventajas de una red inalámbrica son:

- **Comodidad**: Acceder a los recursos de red desde cualquier ubicación del área de cobertura de la red inalámbrica o desde cualquier zona Wi-Fi.
- **Movilidad**: De cualquier parte puede conectarse a la red. Fácil configuración: No hace falta pasar cables, por lo que la instalación puede ser rápida y rentable.
- **Seguridad**: Los avances en redes inalámbricas proporcionan sólidas protecciones de seguridad Costo: Como las redes inalámbricas eliminan o reducen los gastos de cableado, pueden costar menos que las redes cableadas para su operación [17].

1.4 Descripción de un sistema de recolección de datos.

A continuación, se presenta un diagrama de bloques Figura 3, que describe de forma general la arquitectura del sistema de recolección de datos implementado.

El diagrama muestra los componentes principales del sistema: el sensor, encargado de capturar los datos del entorno, el broker, que actúa como intermediario en la transmisión de mensajes y los nodos suscriptores y publicadores, que se encargan de enviar y recibir la información recolectada. Esta estructura permite una transmisión de datos, facilitando la integración de múltiples dispositivos y el procesamiento distribuido de la información.

Figura 3 Conexión con el Broker Autoría propia basada en https://solectroshop.com/es/blog/que-es-mqtt- el- protocolo-de-comunicacion-para-iot-n117

1.4.1 Protocolo Message Queuing Telemetry Transport (MQTT).

Message Queuing Telemetry Transport (MQTT) es un protocolo de comunicación abierto que se distingue por su ligereza y sencillez, gracias a que se puede utilizar con éxito en microcontroladores pequeños, también cuando se trata de un ancho de banda de enlace bajo, por lo que es perfectamente con el concepto IoT.

MQTT en un principio se creó para conectar dispositivos y enviar la información de un sensor a servidores remotos relacionados con el sector de la industria petrolera. [18] Este protocolo se puede utilizar en varias industrias, como:

- Automotriz.
- Manufactura.
- Las telecomunicaciones.
- Internet de las cosas (IoT).

El protocolo MQTT se basa en un modelo de publicación/suscripción (pub-sub) que permite la comunicación asincrónica entre el editor y los clientes suscriptores.

La arquitectura de MQTT sigue una topología de estrella, con un nodo central que hace de servidor o "broker" normalmente con una capacidad teórica de hasta 10000 clientes [18].

Los sensores son los que proporcionaran la información como temperatura, humedad, proximidad etc, mientras que el bróker es el medio donde se pasa la información mediante un suscriptor o el publicador.

Está basado en la pila TCP/IP como base para la comunicación. En el caso de MQTT, cada conexión se mantiene abierta y se “reutiliza” en cada comunicación.

Para filtrar los mensajes que son enviados a cada cliente, los mensajes se disponen en topics (tópicos) organizados jerárquicamente. Un cliente puede publicar un mensaje en un determinado topic. Otros clientes pueden suscribirse a este topic, y el broker le hará llegar los mensajes suscritos.

Por defecto, MQTT emplea el puerto 1883 y el 8883 cuando funciona sobre TLS [19].

De acuerdo con [20] la definición de un TLS es: Seguridad de capa de sockets seguros y capa de transporte. Es un protocolo o regla de comunicación que permite a los sistemas informáticos comunicarse entre sí en Internet de forma segura.

Estructura de un mensaje MQTT

Uno de los componentes más importantes del protocolo MQTT es la definición y tipología de los mensajes, ya que son una de las bases de la agilidad en la que radica su fortaleza. Cada mensaje consta de 3 partes:

- **Cabecera fija**. Ocupa 2 a 5 bytes, obligatorio. Consta de un código de control que identifica el tipo de mensaje enviado, y de la longitud del mensaje. La longitud se codifica en 1 a 4 bytes, de los cuales se emplean los 7 primeros bits, y el último es un bit de continuidad.

- **Cabecera variable**. Contiene información adicional que es necesaria en ciertos mensajes o situaciones.
- **Contenido (Payload)**. Es el contenido del mensaje, puede tener un máximo de 256 Mb, aunque en implementaciones reales el máximo es de 2 a 4 Kb [20].

Tabla 4 Estructura de un mensaje MQTT

<table>
<tr><td colspan="2">Cabecera Fija</td><td rowspan="2">Cabecera variable.</td><td rowspan="2">Contenido (Payload)</td></tr>
<tr><td>Cabecera de control.</td><td>Longitud de mensaje.</td></tr>
<tr><td colspan="2">1 a 5 Bytes</td><td></td><td>256 Mb</td></tr>
</table>

1.4.2 Broker.

El broker es el encargado de gestionar la red y de transmitir los mensajes, para mantener activo el canal, los clientes mandan periódicamente un paquete de datos y según el caso pueden esperar una confirmación del broker [21].

También se puede encontrar que el broker es el servidor que acepta mensajes publicados por clientes y los difunde entre los clientes suscritos, como se puede ver en la Figura 4 [22].

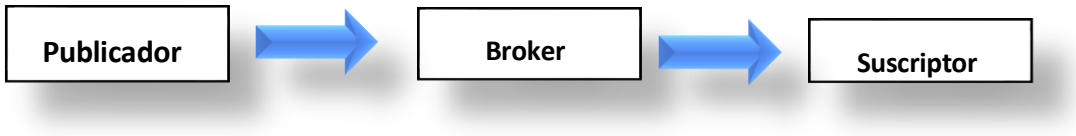

Figura 4 Funcionamiento del bróker, autoría propia basada en Goto IoT | Introducción a MQTT. (s. f.). https://www.gotoiot.com/pages/articles/mqtt_intro/index.html

Cada bloque de la Figura 4 realiza la siguiente función:

- **Publicador**: Es el encargado de enviar los datos de los dispositivos a utilizar.
- **Broker**: Transmite y recibe los datos del publicador.
- **Suscriptor**: Recibe los datos del publicador.

Cada punto ayuda a que se desarrolle una buena comunicación entre los elementos de este proyecto, ejemplo;

- **Conectividad**: Es la capacidad de los dispositivos para comunicarse entre sí y con otros sistemas a través de redes. Esta comunicación permite que los dispositivos recojan, compartan y procesen datos.
- **Sensores**: Estos son los encargados de percibir la información del medio ambiente.
- **Tópicos**: Estos están dentro del protocolo de MQTT, Broker su función es que cada cliente pueda suscribirse o crear un topico donde estará guardada toda la información.

1.4.3 Base de datos de series temporales aplicados a un sistema IoT.

Una base de datos temporales se caracteriza por la recopilación organizada de información o datos estructurados que normalmente se almacena en un sistema informático la cual está controlada por un sistema de gestión de bases de datos, basada solo en un periodo de tiempo [5].

De acuerdo con lo definido anteriormente, las bases de datos de este tipo no tienen una estructura similar a las bases de datos que suelen ser usadas para el almacenamiento de la información en general.

El que se almacenen datos con marca de tiempo hace que la información de cada tabla que se crea necesite ser reducida y que pueda tener una temporalidad en el almacenamiento de los datos, pudiendo ser eliminados datos con una antigüedad predefinida y solo manteniendo los que son recientes.

Hay algunos ejemplos que se pueden encontrar en la literatura para poder realizar una base de datos, los cuales solo se nombraran los que se muestran en la Tabla 5.

Tabla 5 Ejemplos de una base de datos

Prometheus.	**TimescaleDB.**	**QuestDB.**	**AWS Timestream**	**OpenTSDB.**
Se almacenan los datos como series temporales. Su programación es en go.	Es una base de datos de código abierto. Es utilizada para sistemas de: • Monitoreo. • Plataformas de negociación. • Sistemas para recopilar métricas. • Estados de sensores.	Es una base de datos de alto rendimiento. Utiliza un modelo relacional para datos de series de tiempo. La ingesta de datos es muy escalable.	Datos de serie temporal mediante SQL. Almacenamiento para los datos recientes y otro para datos más antiguos. Puede visualizar los datos casi en tiempo real.	Los datos se almacenan como son proporcionados por el usuario Se ejecutan en Hadoop y HBsase. Se genera gráficos desde GUI que es una interfaz de código abierto.

En la Tabla 6 se muestra un cuadro comparativo de las bases de datos mencionadas en la Tabla 5. Los conceptos clave de comparación son cuatro:

Tabla 6 Ejemplos de una base de datos

Base de datos	**¿Se integra a Grafana?**	**¿Visualiza los datos en tiempo real?**	**Almacenamiento (retención de datos) gratuito**	**Almacenamiento (retencipon de datos) pagado**
Influx db	Si	Si	30 días	Ilimitado
TimescaleDB	No	Si	30 días	30 USD por 30 días
QuestDB	Si	Si	30 días	Por 30 días $248
AWS Timestream	No	Si	50 GB para la ingesta de datos, 100 GB de almacenamiento en nivel magnético, 750 GB por hora de almacenamiento en nivel de memoria, y 750 GB para el uso de consultas durante un mes.	Costo de computación: 697,88 US. Costo de almacenamiento: 40,00 USD. Costo mensual total: 737,88 USD.

La tabla anterior se construyó en base a InfluxData. (2021, 10 diciembre). InfluxDB: Open Source Time Series Database | InfluxData. https://www.influxdata.com/ PostgreSQL ++ for time series and events. (s. f.). Timescale. https://www.timescale.com/ , QuestDB. (s. f.). QuestDB | High performance time series database. https://questdb.io/ , Introduction to Amazon Timestream (Introducción a Amazon Timestream) (1:34). (s. f.). [Vídeo]. Amazon Web Services, Inc. https://aws.amazon.com/es/timestream

1.4.4 Tablero de datos

Un tablero de datos ayuda tener un orden en la información que se maneja, esto se realiza mediante herramientas que contienen gráficos, cuadros, dashboards, entre otros [7].

La presentación de la información de manera gráfica se deja al tablero de datos. Existen varias opciones tanto libres como de pago de regalías, algunas de ellas se muestran en la Tabla 7.

Tabla 7 Comparación de los softwares que nos ofrecen tableros

Software	**¿Tiene algún costo realizar la cuenta?**	**¿Su tiempo de utilidad es lo suficiente bueno para trabajar?**	**Almacenamiento Gratuito.**	**Almacenamiento pagado**	**¿Tiene distintos tipos de tableros para trabajar?**
Grafana.	No	Si	50Gb Registros. 50Gb seguimiento. 50Gb de perfiles.	Ilimitado	Si
Arduino IoT cloud.	No	No	0.0122 Gb para almacenar bocetos.		No
Ubidots.	No	Si		Profesional 15 Mb Industrial 50 Mb Empresa 100 Mb	No
ThingsBoard.	No	No	10 millones de puntos de datos por mes.	Prototipo 100 millones de datos puesta en marcha	No

				500 millones de puntos de datos de acceso.	

Basada en:

Dashboards (s. f.). Grafana Labs. https://grafana.com/grafana/dashboards/?plcmt=footer,

Arduino Cloud | Build, Control, monitor your IoT Projects. (s. f.). https://cloud.arduino.cc/ ,

Ubidots - Sencillo pero potente IoT industrial. (s. f.). https://es.ubidots.com/ ,

Thingsboard. (s. f.). ThingsBoard — Opensource IoT (Internet of Things) Platform. ThingsBoard. https://thingsboard.io/

1.4.5 Plataforma basada en software libre.

Un software de código abierto se desarrolla y se mantiene a través de una colaboración abierta. Suele estar disponible sin costo para que cualquier persona que lo utilice, lo analice, lo modifique y lo comparta libremente. A diferencia de aplicaciones de software de código cerrado, el cual se debe de realizar un pago para adquirir derecho de poder modificarlo y compartirlo, sin poder editar [23].

Dado lo anterior, cabe indicar que se tienen diferentes opciones disponibles para la implementación del sistema de recolección de datos. Una de ellas está orientada al uso de software libre, lo que implica realizar la instalación y configuración de cada uno de los elementos de la arquitectura para un sistema IoT.

Otro de los requerimientos que conlleva la elección de esta opción es tener la infraestructura local de red (equipo de servidores, configuración de la red cableada e inalámbrica, conectividad a Internet), además del conocimiento de la administración del sistema operativo.

1.4.6 Plataforma comercial.

Las plataformas comerciales ofrecen no tener que construir una infraestructura propia para el usuario, indicando que se requiere solamente de que los nodos tengan conexión a Internet y que se incluyan las credenciales de acceso a la plataforma. Esto permite centrarse en el desarrollo de los nodos que serán conectados, dejando que la empresa que ofrece el servicio, se encargue de la administración.

Además, se deben pagar por los servicios solicitados y el almacenamiento de los datos. Su gran ventaja es facilitar la realización de actividades comerciales o transacciones entre empresas, consumidores o entre usuarios. Estas plataformas pueden ser de diferentes tipos, pero su objetivo principal es apoyar en la venta de productos y servicios.

Capítulo 2. Sistemas de recolección de datos

2.1. Sistema comercial Ubidots.

La plataforma comercial Ubidots permite configurar de manera sencilla el tablero de datos, la recolección de los datos y da las pautas para poder agregar los nodos que se tienen en el sistema que se va a desplegar. Sus características principales son:

- **Interfaz intuitiva**: Ofrece una interfaz gráfica que facilita la creación de tableros, llamados dashboards, visualización de datos en tiempo real y el diseño de interfaces de usuario personalizadas.
- **Integración de dispositivos**: Soporta una amplia gama de dispositivos y protocolos, lo que permite la integración de hardware variado con la plataforma.
- **Visualización de datos**: Proporciona herramientas para crear gráficos, mapas y otros tipos de visualizaciones que ayudan a interpretar los datos obtenidos.
- **Alertas y notificaciones**: Permite configurar alertas y notificaciones basadas en umbrales de datos, lo que es útil para monitoreo y respuesta proactiva a eventos.

- **API y Webhooks**: Ofrece APIs y webhooks que permiten la integración con otras aplicaciones y servicios, facilitando la automatización y la extensión de funcionalidades.
- **Escalabilidad y seguridad**: Está diseñada para manejar grandes volúmenes de datos y proporciona características de seguridad para proteger la información sensible.

Ubidots es utilizado en una variedad de industrias, como la agricultura, la manufactura, la salud y el transporte, facilitando la gestión de datos de manera eficiente y accesible [24].

2.1.1. Configuración de la plataforma Ubidots en la nube.

Se accede a la página oficial de Ubidots: https://ubidots.com/

Figura 5 Página oficial UBIDOTS. Fuente: captura de pantalla de la página web: https://stem.ubidots.com/accounts/signin/

Se crea una cuenta, si ya se cuenta con una se accede a ella. Enseguida, se crea un dispositivo en blanco y se selecciona en el tipo (Figura 6 y 7).

Figura 6 Creacion de dispositivos. Fuente: captura de pantalla de la página web: https://stem.ubidots.com/accounts/signin/

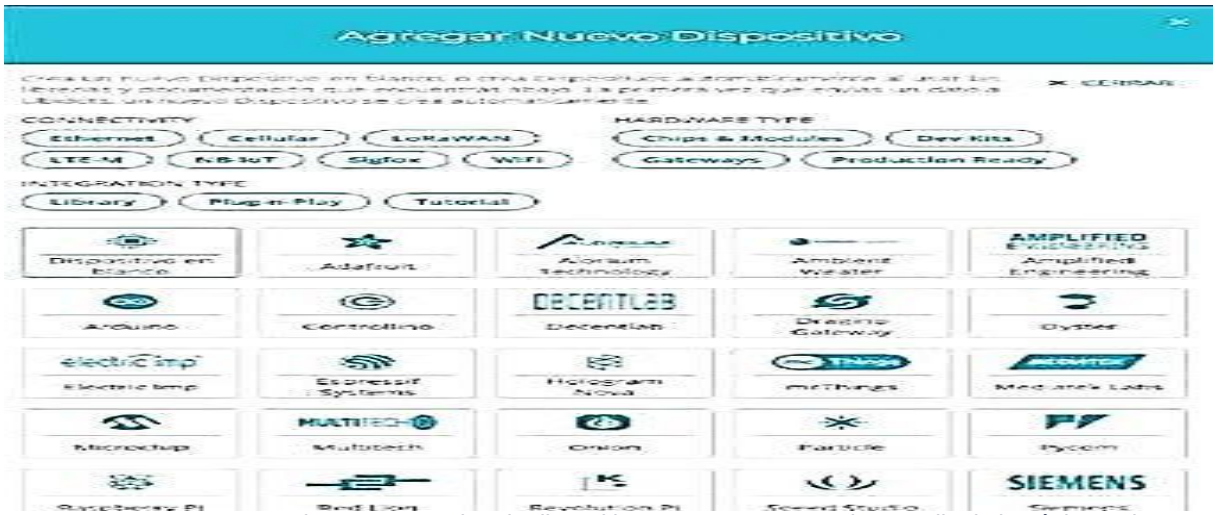

Figura 7 , Creacion de dispositivos. Fuente: captura de pantalla de la página web: https://stem.ubidots.com/accounts/signin/

Posteriormente, se coloca un nombre para el dispositivo y se agregará automáticamente (Figura 8).

Figura 8 Nombre del dispositivo Fuente: captura de pantalla de la página web: https://stem.ubidots.com/accounts/signin/

Al ingresar a la configuración del dispositivo se visualiza “API Label”; información necesaria para el código y su funcionamiento.

Se crean las variables de Humedad, Temperatura, Luminosidad y Proximidad para el dispositivo (Figura 9).

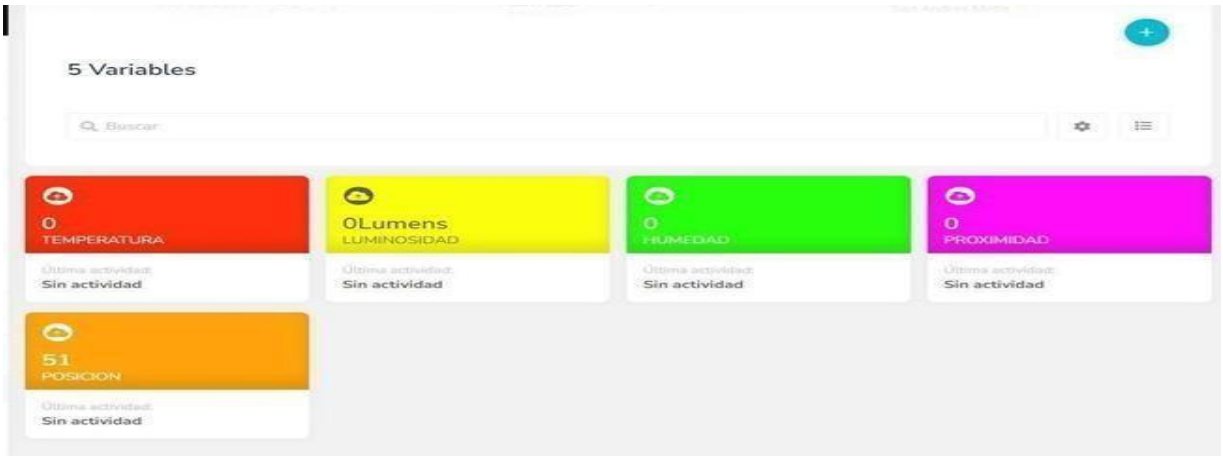

Figura 9 Variables creadas, Fuente: captura de pantalla de la página web: https://stem.ubidots.com/accounts/signin/

Se asigna el número de variable. Así, Temperatura es la variable 1, Luminosidad la variable 2, Humedad la variable 3 y Proximidad la variable 4.

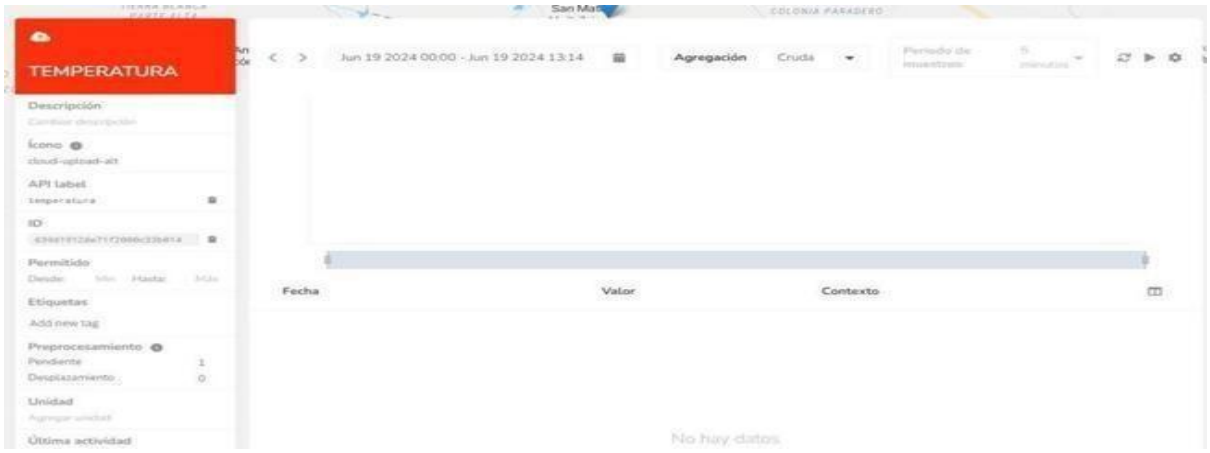

Figura 10 Variable Temperatura Fuente: captura de pantalla de la página web: https://stem.ubidots.com/accounts/signin/

Al seleccionar una de ellas se mostrará una gráfica donde se puede ver los valores que se almacenan en tiempo real, (Figura 10).

Para crear un nuevo Dashboards, seleccionar el apartado “Datos” > “Tableros” > “Crear nuevo tablero, (Figura 11).

Figura 11 Creación de tablero Fuente: captura de pantalla de la página web: https://stem.ubidots.com/accounts/signin/

Una vez creado el Dashboard, se pueden agregar nuevos widgets para las variables creadas, (Figura 12).

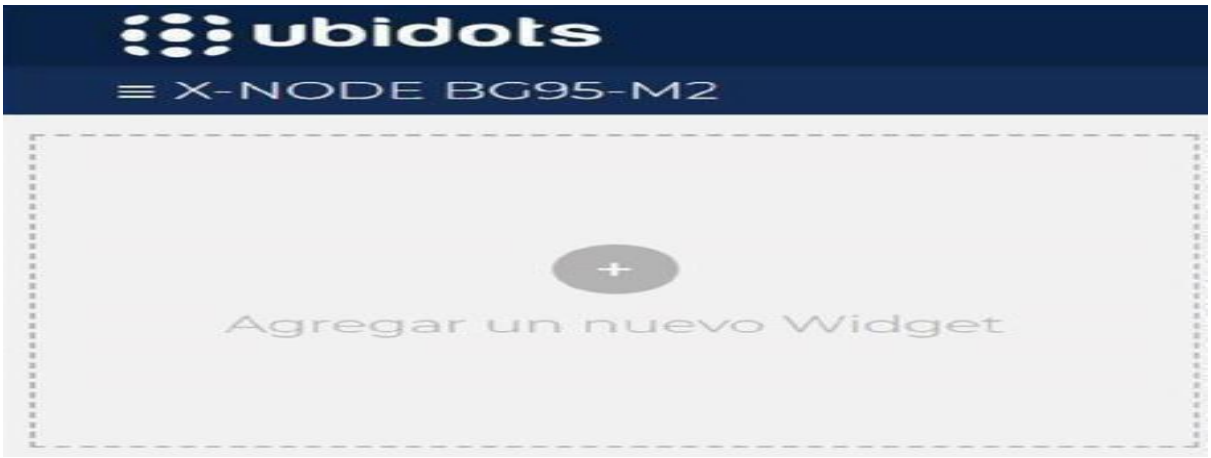

Figura 12 Widget Fuente: captura de pantalla de la página web: https://stem.ubidots.com/accounts/signin/

Se agrega un widget “Tanque” y se asocia a la variable “Humedad” con los datos que vayan obteniendo, después se selecciona el icono de guardar.

Figura 13 Widget tanque Fuente: captura de pantalla de la página web: https://stem.ubidots.com/accounts/signin/

De esta manera, se crean los Widgets para cada una de las variables: temperatura, proximidad y luminosidad.

Una vez guardados todos los Widgets, el Tablero toma la forma como se muestra en la Figura14.

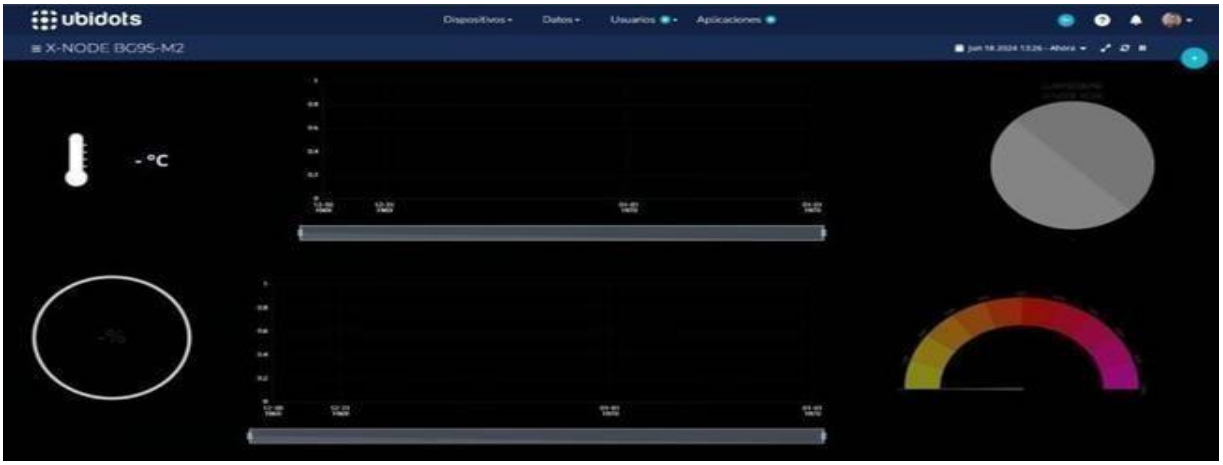

Figura 14 Tablero en Ubidots Fuente: captura de pantalla de la página web: https://stem.ubidots.com/accounts/signin/

2.2. Sistema basado en software libre.

El sistema basado en software libre requiere de mayor intervención al momento de realizar la instalación. Sin embargo, el espacio en donde se instala y la

infraestructura de red pueden ser propias. Para este trabajo, la parte de la infraestructura se considera ya establecida, lo que resta es describir la instalación y configuración de cada uno de los elementos que conforman la arquitectura del sistema de recolección de datos, los cuales son:

- Elección del sistema operativo.
- Instalación del software.
- Programación del software.
- Configuración del software.
- Monitoreo de los datos.
- Recolección de los datos.

2.2.1. Eclipse Mosquitto.

Eclipse Mosquitto es un bróker, intermediario de mensajes de código abierto (con licencia EPL/EDL) que implementa las versiones 5.0, 3.1.1 y 3.1 del protocolo MQTT. Mosquitto es un software muy ligero, adecuado para su uso en todos los dispositivos.

El método de mensajería que usa MQTT (suscripción y publicación) permite su uso con facilidad en aplicaciones como el internet de las cosas, sensores, móviles, microcontroladores, entre otros. Por ejemplo, PLC con Arduino Controllino MAXI Automation (el cual cuenta con su Librería MQTT), datalogger Z-LTE-WW con compatibilidad para este protocolo. Un punto a resaltar es que Mosquitto es compatible con la interfaz de usuario HMI P5070NA (que será empleada en este trabajo).

El software Mosquitto también proporciona una biblioteca C para implementar clientes MQTT y los muy populares clientes MQTT de línea de comandos mosquitto_pub y mosquitto_sub. [25]

Existen más brokers aparte de mosquitto, como: EMQ X, RabbitMQ, HiveMQ, VerneMQ, CloudMQT.

Mosquitto es una buena opción para proyectos IoT ya que este cuenta con muchas características ideales para el usuario, ya que su instalación es muy fácil de realizar, su configuración e instalación es simple al igual que su uso. Por estas características, se ha seleccionado este broker.

Una vez instalado correctamente el servicio y cliente de mosquitto se verifica el estatus para ver que no exista ningún error (Figura 15).

```
root@IoTV2: "# systemctl status mosquitto
mosquitto.service - Mosquitto MQTT Broker
Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor preset: enabled)
Active: active (running) since Mon 2025-06-02 19:48:09 CST: 3min 41s ago
Docs:   man: mosquitto.conf (5)
        man: mosquitto (8)
Process: 329 ExecStart Pre=/bin/mkdir -m 740 -p /var/log/mosquitto (code=exited, status=0/SUCCES
Process: 347 ExecStart Pre=/bin/chown mosquitto /var/log/mosquitto (code=exited, status=0/SUCCES
Process: 349 ExecStart Pre=/bin/mkdir -m 740 -p /run/mosquitto (code=exited, status=0/SUCCESS)
Process: 360 ExecStart Pre=/bin/chown mosquitto /run/mosquitto (code=exited, status=0/SUCCESS)
Main PID: 407 (mosquitto)
Tasks: 1 (limit: 2323)
Memory: 2.6M
CPU:162ms
CGroup: /system.slice/mosquitto.service 407 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
```

Figura 15 Estado del servidor mosquitto.

Para verificar la comunicación, se realiza una prueba con MOSQUITTO_SUB Y MOSQUITTO_PUB.

Mosquitto_sub es un cliente MQTT versión 3.1.1 simple que se suscribirá a temas e imprimirá los mensajes que reciba.

Mosquitto_pub es un cliente MQTT versión 5 simple que publicará un solo mensaje sobre un tema.

De manera sencilla el mosquitto_sub es la herramienta para recibir mensajes (suscriptor) del mosquitto_pub (editor).

Las versiones 3.1.1 y 5 funcionan para esquemas IoT, las diferencias que se encuentran una con otra es que la versión 3.1.1 es más básica que la versión 5, y la versión 5 es más compleja ofreciendo mayor complejidad.

Los mensajes publicados a través de mosquitto_pub se verán publicados en el receptor mosquitto_sub lo que significara que el servicio mosquitto se realizó correctamente.

El comportamiento del Broker Mosquitto, se puede observar desde un archivo único llamado mosquitto.conf, que debe estar situado en el mismo directorio de instalación de Mosquitto.

```
root@IoTV2: # cd /etc/mosquitto
root@IoTV2: /etc/mosquitto# ls
aclfile.example certs mosquitto.conf old.auth pskfile.example
ca_certificates conf.d mosquitto.config oldpassud pwfile.example
root@IoTV2: /etc/mosquitto#
```

Figura 16 Comando ls muestra todo el listado de archivos y directorios.

El archivo Config, que tiene acceso mediante un usuario/contraseña, se pueden definir algunos archivos instrumentales, que se requieren para establecer una comunicación con otros servidores.

Antes de hacer las modificaciones al archivo Config, se recomienda relaizar una copia.

```
GNU nano 5.4
listener 1883 0.0.0.0
allow_anonymous true
passwd file /etc/mosquitto/.passwd
```

Figura 17 Archivo de mosquitto.conf.

Para terminar de configurar el archivo Config, se agregarán las líneas de:

- **listener 1883 0.0.0.0:** Este comando, fija el puerto en el que el servidor MQTT va a recibir las peticiones.
- **allow anonymous false**: Niega las conexiones anónimas.
- **password file /etc/mosquitto/. passwd**: Realiza el registro de los nombres de usuario y contraseñas asignadas para los usuarios registrados que se desee tener en el servidor.

Cada vez que se realice una modificación en el archivo de configuración, se tiene que detener el servidor y volverlo a iniciarlo para que los cambios tengan efecto.

Con esta configuración el sistema permitirá que los usuarios registrados se puedan conectar.

Se puede añadir un usuario y una contraseña, (Figura 18).

```
root@IoTV2: /etc/mosquitto# cd
root@IoTV2: "# cd /etc/mosquitto
root@IoTV2: /etc/mosquitto# mosquitto passwd -c passwd usuario
```


Figura 18 Archivo de mosquitto.conf.

Ya que se creó un usuario y contraseña, se puede verificar que se guardó correctamente, pero la contraseña aparecerá de manera encriptada.

```
root@IoTV2: /etc/mosquitto# less .passwd
usuario: $7%dnjncfjgndksnjfcjhchjhgkdhckhglhslc
```

Figura 19 Usuario registrado con contraseña.

Se realizan algunas pruebas para comprobar su funcionamiento y se observa que en la ventana en la que se suscribió el tópico “Sensor/Co2” –m aparece en la otra ventana el valor de 100, esto indica que se está suscribiendo y publicando correctamente.

Una vez que se ha configurado usuario/contraseña y los archivos, se pueden comprobar los cambios realizados.

```
root@IoTV2: "# mosquitto_sub -d -h localhost -t "Sensor/Temperatura" -u" usuario" -P
"xxxx" -W
client (nul) sending CONNECT
client (nul) received CONNACK (0)
client (nul) sending
SUBSCRIBE (Mid: 1, Topic: Sensor/Temperatura, QoS: 0,Option:0x00)
client (nul) received SUBACK
Subscribed (Mid: 1):0
Client (nul) received PUBLISH (do, qo, ro, no, "Sensor/Temperatura"... (3 bytes))
Sensor/Temperatura 100
```

Figura 20 Ventana para usar el comando mosquitto_sub que permite suscribir en el servidor MQTT.

```
root@IoTV2: # mosquitto_sub -d -h localhost -t "Sensor/Temperatura" -m "100" -u "usuario"
-P "xxxx" -v
client (nul) sending CONNECT
client (nul) received CONNACK (0)
client (nul) sending PUBLISH (do, ro, qo, n1, "Sensor/Temperatura"...(3 bytes))
client (nul) sending DISCONNECT_
```

Figura 21 Ventana para usar el comando mosquitto_pub que permite suscribir en el servidor MQTT

Para el password se utilizó la "-P" en mayúscula, porque la "-p" en minúscula es para indicar el puerto "-p 1883”.

Para poder observar los datos se utiliza MQTT Explorer, este programa se utiliza para proporcionar una descripción general estructurada de los temas y subtemas.

Para habilitar un servidor nuevo, se selecciona en el símbolo "+" en rojo y se ingresa al menú para agregar nuevas conexiones. Posteriormente, se llenan los datos del Bróker MQTT. La dirección IP del broker es de la red local con el puerto 1883, colocando el usuario y contraseña, finalmente se guarda y se conecta.

Figura 22 MQTT Explorer.

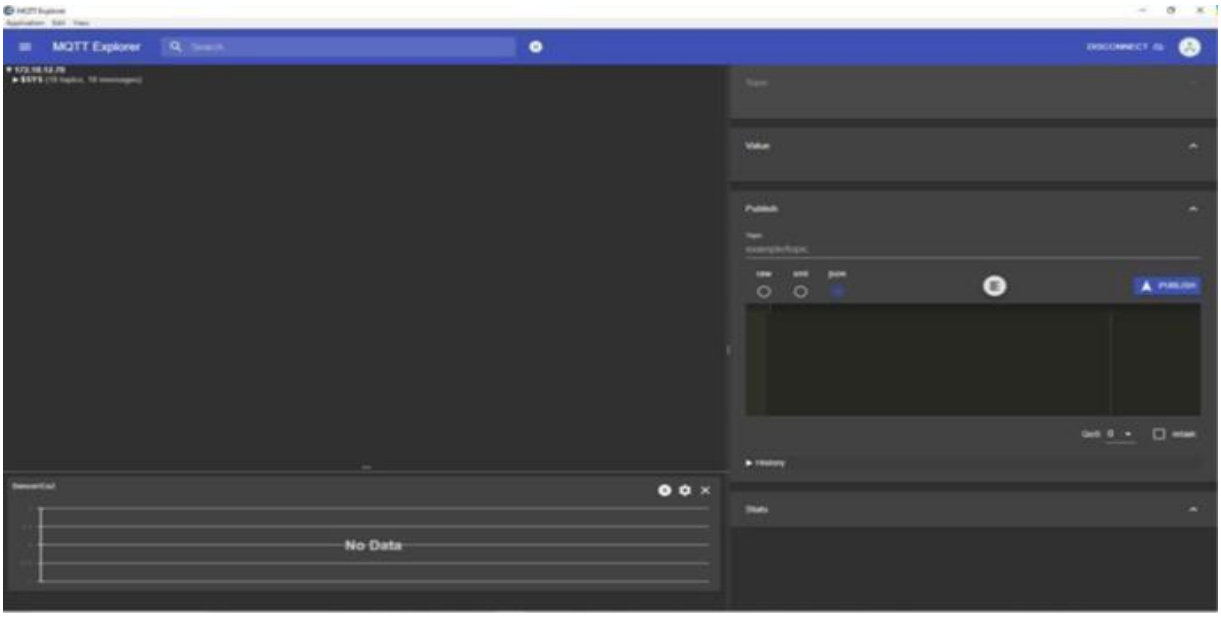

Figura 23 Pantalla de MQTT Explorer.

Al conectar el MQTT, se abrirá la pantalla principal donde se podrá suscribir tópicos o publicarlos en el servidor. Los tópicos están formados por uno o más “niveles” separados entre sí por una barra inclinada’/ ‘(Figura 23). Cada nivel debe está formado por uno o más caracteres [26].

Por un lado, el Broker acepta todos los tópicos. No es necesario crearlo antes de publicar o suscribirse al Broker. Por su parte, los clientes pueden suscribirse a uno o varios Topicos.

Finalmente, los clientes publican mensajes indicando un único tópico. El Broker recibe el mensaje y, si encuentra alguna suscripción que cumpla con el tópico, transmite el mensaje a los clientes suscritos.

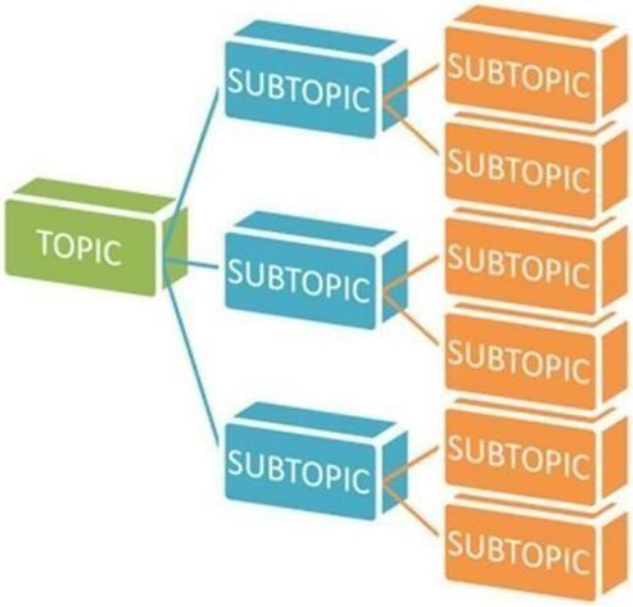

Figura 24 Arquitectura de tópicos

Para comprobar que la comunicación con dispositivos opera adecuadamente, se pone en marcha el microcontrolador ESP32, para que publique varios valores aleatorios.

Cuando se establece una conexión con el bróker, el MQTT Explorer muestra el sensor y su valor, como se muestra en la Figura 26.

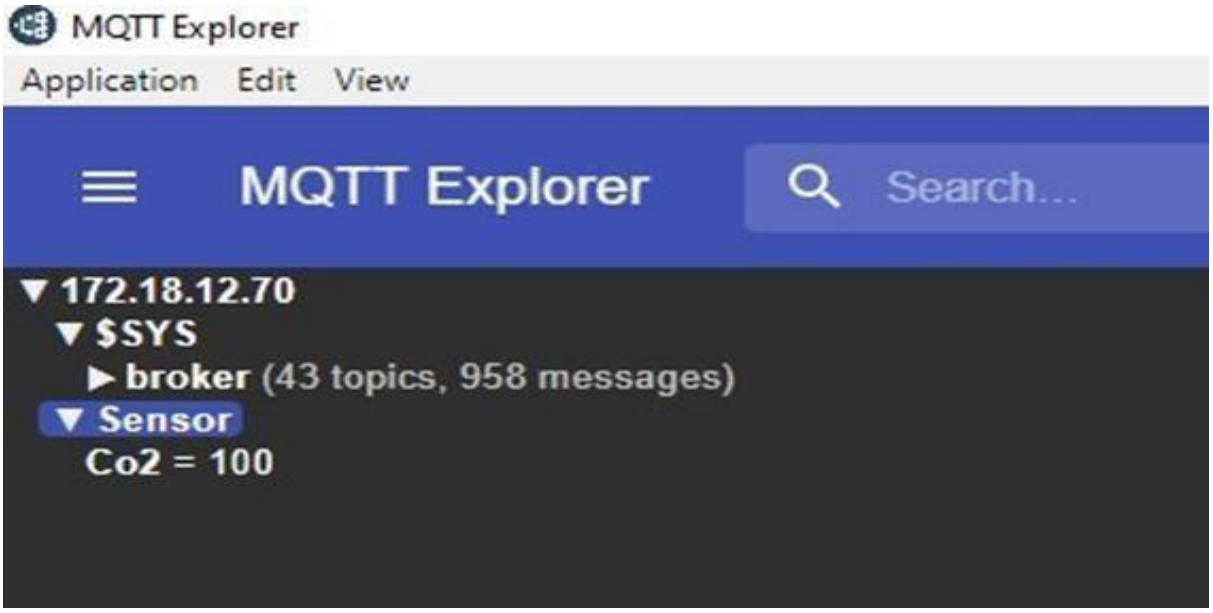

Figura 25 Pantalla de MQTT Explorer.

En la Figura 25 se observa una lista desplegable debajo del servidor donde aparece el tópico del sistema y el tópico al que suscribió, en este caso es "Sensor/Co2". Al desplegar el sensor se muestra el valor publicado. Así, que los valores que recibe el dispositivo ESP32 son actualizados instantáneamente.

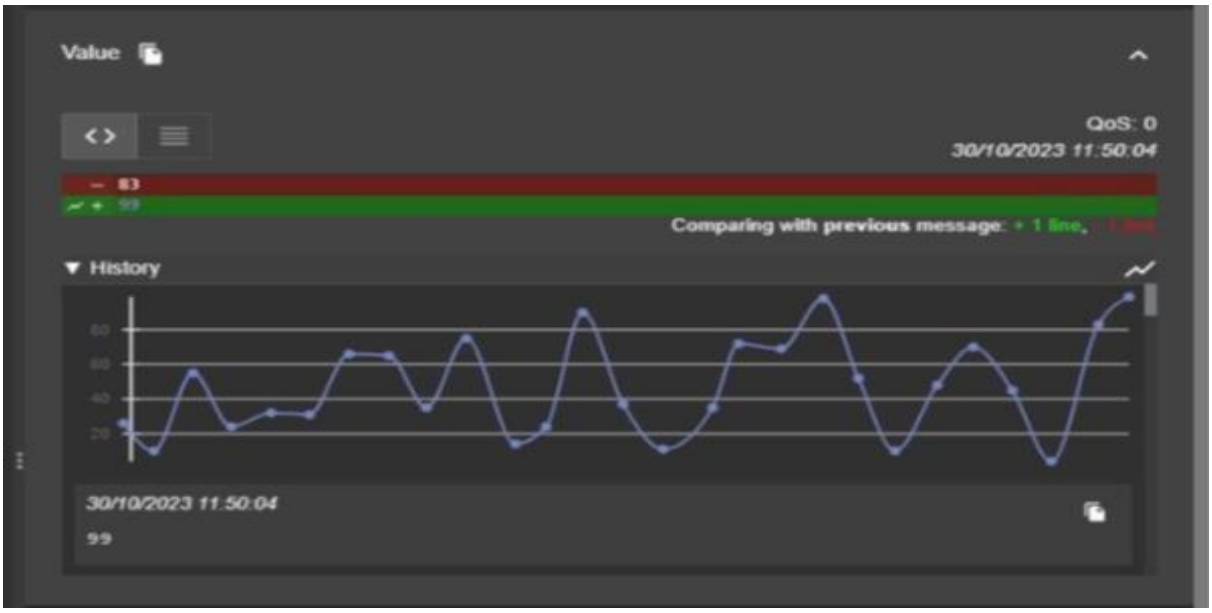

Figura 26 Grafico por MQTT Explorer

En el MQTT Explorer, también se puede observar un despegable llamado valor (value), que muestra los valores actuales de la última publicación y una línea roja que indica el número de mensajes publicados hasta la fecha. Si se pulsa el símbolo de una línea, justo al principio de la línea verde se obtendrá un gráfico de los últimos valores publicados.

De este modo se puede ver en la consola, una serie de valores que se van publicando (Figura 27 y 28).

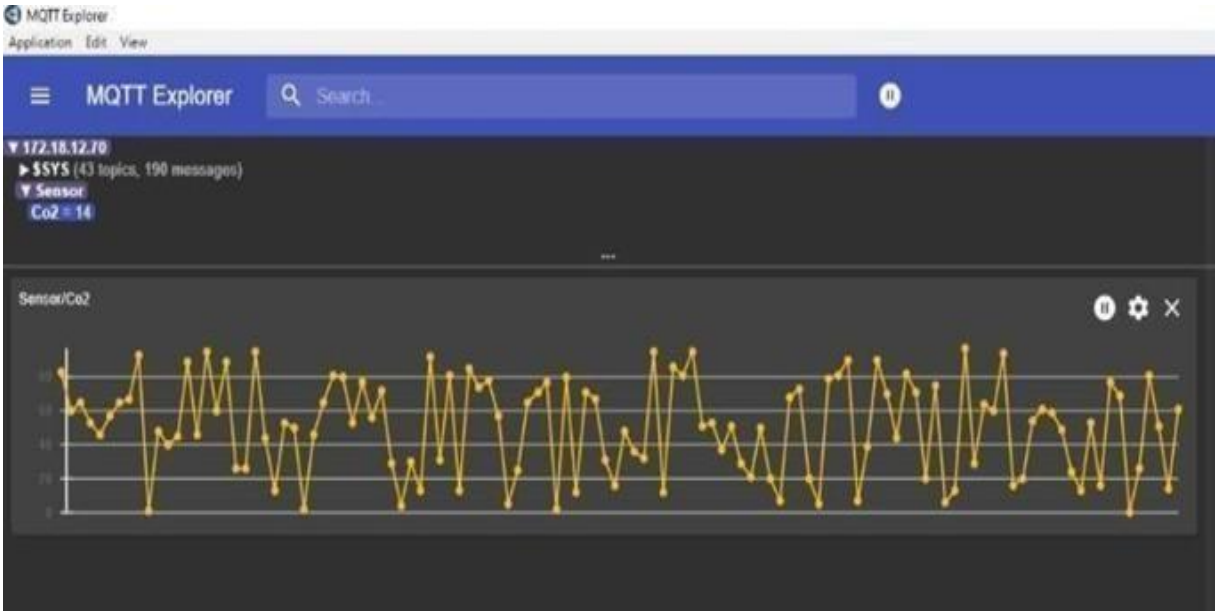

Figura 27 Grafico.

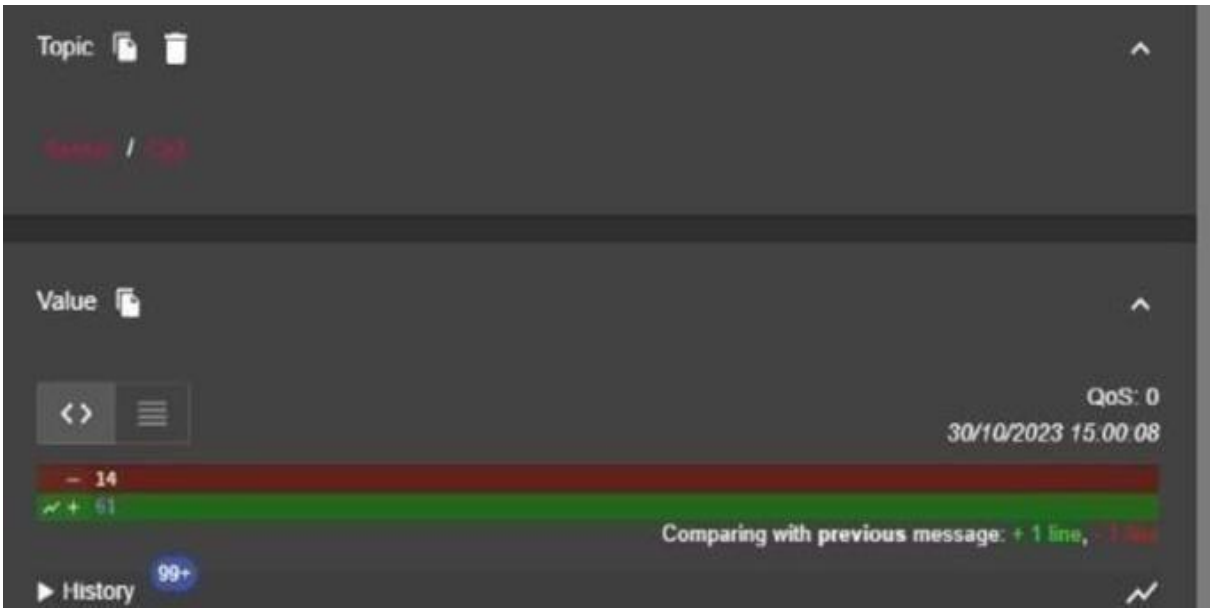

Figura 28 Valores.

2.2.2. Base de datos Influxdb.

Cuando ya se tiene definido el broker, en este caso MQTT, se centraliza para la gestión de los sensores que enviara información a la placa ESP32.

Para ello se debe tener en cuenta que Mosquitto no dispone de una base interna que almacene las lecturas de los sensores. Si se requiere guardarlas a largo plazo para hacer graficas o estadística, es necesario almacenarlos en una base de datos externa.

En [27] se menciona todas las ventajas que ofrece Influxdb como:

- Ofrece análisis de los datos en tiempo real.
- Puede conservar muchos valores de datos.
- Es compatible con Grafana.
- Permite crear dashboards para visualizar los datos en tiempo real.

Influxdb tiene muchas más características, es por eso que se eligió como base de datos, ya que es fácil de instalar, es gratuito, ofrece un amplio espacio de almacenamiento para el valor de las lecturas de los sensores. Otro aspecto que interesa es que Influxdb es compatible con MQTT y con otros softwares que se emplean en este trabajo.

Para iniciar la instalación y configuración de Influxdb se realiza el siguiente procedimiento.

Se añaden los repositorios de Influxdb en la siguiente ruta: repos.influxdata.com para este caso, será un repositorio DEB (empleado en plataformas basadas en Ubuntu, Debian, entre otras).

Cuando se tengan descargados los repositorios de Influxdb se procede a su instalación. En la Figura 29 se muestra el estado del sistema.

```
root@IoTV2: "# ls
influxdata-archive_compat.key     influxdata-archive_compat.key.3  telegraf.conf.save
influxdata-archive_compat.key.1        influxdb-1.8.10_linux_amd64.tar.gz.asc     influxdata-
archive_compat.key.2                   influxdb2-2.7.0-amd64.deb
root@IoTV2:~# systemctl status influx
Unit influx.service could not be found.
root@IoTV2:~# systemctl status influxdb
influxdb.service -  InfluxDB is an open-source, distributed, time series database
                                                    ,
Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset: enabled)
Active: active (running) since Mon 2025-06-02 19:48:11 CST; 1h 47min ago
Docs: https://docs.influxdata.com/influxdb/
Process:  327  ExecStart=/usr/lib/influxdb/scripts/influxd-systemd-start.sh  (code=exited,
status= >
 PID: 351 (influxd)
Tasks: 11 (limit: 2323)
Memory: 115.OM
CPU: 17.599s
CGroup: /system.slice/influxdb.service
                        351 /usr/bin/influxd -config /etc/influxdb/influxdb.conf
```

Figura 29, Estatus de sistema influxdb (running).

Ya que la instalación sea correcta, se procede a la creación de una base de datos, para ello se puede usar el intérprete de comando Influx que permite usar comandos básicos desde la terminal, (Figura 30).

```
root@IoTV2:"# influx
Connected to http://localhost:8086 version 1.8.10
InfluxDB shell version: 1.8.10
```

Figura 30 influx (running)

En la Figura 31 se puede observar la versión en curso y el puerto en el que por

```
root@IoTV2:#  influx
Connected to http://localhost:8086 version 1.8.10
InfluxDB shell version: 1.8.10
>create database dbSensor
>show databases
name:  databases
name
internal
Sensor
XN04
dbSensor
>use dbSensor
Using database dbSensor
```

Figura 31 Influxdb (creación de base).

Como se muestra en la Figura 32, la base de datos ha sido nombrada dbSensor, lo que implica que, a partir de este punto, todas las instrucciones y operaciones estarán dirigidas a esta base de datos. A continuación, es necesario crear un usuario específico para la base de datos con el fin de implementar un control básico de acceso mediante nombre de usuario y contraseña. Para ello, se deben ejecutar los siguientes comandos y, posteriormente, asignarle todos los privilegios necesarios.

```
> use dbSensor
Using database dbSensor
>create user usuario with password "xxxx"
>grant all on dbSensor to usuario
```

Figura 32 influxdb.

Con esto, se ha creado la Base de Datos y un usuario al que se le ha otorgado todos los derechos. Para tener una mayor seguridad también se le puede añadir una contraseña, Figura (33).

```
root@IoTV2: # influx -username usuario -password XXXX
Connected to http://localhost:8086 version 1.8.10
Int luxDB shell version: 1.8.10
> create database dbSensor
> show databases
name: databases
name
----------
_internal
Sensor
XN04
dbSensor
> use dbSensor
Using database dbsensor
```

Figura 33 Influxdb con usuario y password.

Finalmente, se pueden pasar los valores que lleguen desde MQTT a la Base de Datos influxdb.

2.2.3 Telegraf.

Para poder registrar las suscripciones que llegan desde MQTT a la base de datos Influx, se necesita un programa intermedio que lea en el primero y escriba en el segundo. Telegraf es un agente de recopilación de datos creado por Influxdb.

Los desarrolladores de influx crearon un módulo externo que se puede configurar para recibir datos de entrada de una fuente o varias, por muy distinta procedencia o formato que pueden ser, y a su vez especificar la salida a influx o a otros destinos finales [28]. Por estas razones y con la ventaja de que el envio de datos es en tiempo real, Telegraf es el programa seleccionado.

Para realizar la instalación y configuración de telegraf se realiza lo siguiente:

```
root@IoTV2: # systemctl status telegraf
telegraf.service - Telegraf
Loaded: loaded (/lib/systemd/system/telegraf.service; enabled; vendor preset: enabled)
Active: active (running) since Mon 2025-06-02 19:48:12 CST; 2h 29min ago
Docs: https://github.com/influxdata/telegraf
Main PID: 335 (telegraf)
Tasks: 8 (limit: 2323)
Memory: 172.3M
CPU: 30.304s
CGroup: /system.slice/telegraf.service
        335 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -config-directory /e
```

Figura 34 Estatus de telegraf. service (creación de base).

En seguida se realiza la configuración de telegraf, desde su archivo de configuración.

Estas configuraciones controlan varios aspectos del comportamiento de Telegraf en términos de recolección, envío y procesamiento de métricas, y ofrecen opciones para ajustar la precisión y el registro de datos.

El telegraf.conf se gestiona a base de un plugin de entrada, y pluggins de salida.

Se configura el output de Telegraf, en este caso Influxdb. Estas configuraciones definen cómo se enviarán las métricas recopiladas hacia la base de datos, incluyendo la URL del servidor, el nombre de la base de datos de destino y las credenciales de autenticación HTTP necesarias para acceder a Influxdb.

```
## Read metrics from MQTT topic(s)
[[inputs.mqtt_consumer]]
##  ## Broker URLS for the MQTT server or cluster. To connect to multiple
##  ## clusters or standalone servers, use a separate plugin instance.
#    ## example: servers = ['tcp://localhost: 1883"]
# ##    servers = ["ssl://localhost: 1883"]
#  ##   servers = ['ws://localhost: 1883"]
#  ##  servers = ['tcp://192.168.1.250:1883"]
#   ## Topics that will be subscribed to.
topics = [
"pasillo/Sensor/Temperatura",
"pasillo/Sensor/Luminosidad",
"pasillo/Sensor/Humedad",]

# "telegraf/host01/cpu",
# "telegraf/+/mem",
#  "sensors/#",
]
              # ## The message topic will be stored in a tag specified by this value.
     ## to the empty string no topic tag will be created.
#topic_tag= "topic"
```

Figura 35 Configuración de métricas de entrada

```
# # Configuration for sending metrics to InfluxDB
[[outputs. influxdb]]
##  # The full HTTP or UDP URL for your InfluxDB instance.
##
## Multiple URLs can be specified for a single cluster, only ONE of the
# ## urls will be written to each interval.
# ## urls = ["unix:///var/run/influxdb.sock"]
# ## urls = ["udp://127.0.0.1:8089"]
urls = ['http://192.168.1.250:8086"]
# ## The target database for metrics; will be created as needed.
# ## For UDP url endpoint database needs to be configured on server side.
database = "dbSensor"
# ## The value of this tag will be used to determine the database. If this
## # tag is not set the 'database' option is used as the default.
# database_tag= “”

## If true, the 'database_tag' will not be included in the written metric.
# exclude_database_tag= false

## If true, no CREATE DATABASE queries will be sent. Set to true when using
## Telegraf with a user without permissions to create databases or when the
## database already exists.
skip_database_creation = false
## # Name of existing retention policy to write to. Empty string writes to
## # the default retention policy. Only takes effect when using HTTP.
# retention_policy = """"
        ## The value of this tag will be used to determine the retention policy. If this
```

Figura 36 Configuración de métricas de salida.

Se guardan los cambios y se inicia el servicio de telegraf. En la Figura 36 se puede observar el arranque de telegraf.

```
root@IoTV2: /etc/telegraf# telegraf
2025-06-03T04:45:46Z I! Loading config: /etc/telegraf/telegraf.conf
2025-06-03T04:45:46Z I! Starting Telegraf 1.30.1 brought to you by InfluxData the makers of InfluxDB
2025-06-03T04:45:462 I! Available plugins: 233 inputs, 9 aggregators, 31 processors, 24 parsers, 60 outputs, 6 secret-stores
2025-06-03T04:45:46Z I! Loaded inputs: cpu disk diskio kernel mem mqtt_consumer processes swap system
2025-06-03T04:45:46Z I! Loaded aggregators:
2025-06-03T04:45:462 I! Loaded processors:
2025-06-03T04:45:46Z I! Loaded secretstores:
2025-06-03T04:45:462 I! Loaded outputs: influxdb 2025-06-03T04:45:462 I! Tags enabled: host=IoTV2
2025-06-03T04:45:46Z I! [agent] Config: Interval: 10s, Quiet:false, Hostname: "IoTV2", Flush Interval:
10s
2025-06-03T04:45:46Z I! [inputs.mqtt_consumer] Connected [tcp://192.168.1.250:1883]
```

Figura 37 , Arranque de telegraf (conexión).

2.2.4 Tablero de datos Grafana.

Grafana es un programa que extrae datos de múltiples fuentes y genera gráficos dinámicos en un servidor Web al que se puede acceder para estudiarlos.

Para su instalación y configuración se realizará lo siguiente:

Se instala el repositorio de Grafana (versión más reciente) para obtener un clave APT y validar los paquetes que se descarguen, obteniendo una clave de seguridad.

Los repositorios de Grafana Labs se obtiene de: https://packages.grafana.com/

En seguida, se procede con la instalación de Grafana, si esta correcto se verifica el estado de Grafana, como se muestra en la Figura 37.

```
root@IoTV2:/etc/grafana# systemctl status grafana-server
grafana-server.service Grafana instance
Loaded: loaded (/lib/systemd/system/grafana-server.service; enabled; vendor preset:
enabled) Active: active (running) since Mon 2025-06-02 19:48:11 CST; 3h 4min ago
Docs: http://docs.grafana.org
Main PID: 480 (grafana)
Tasks: 12 (limit: 2323)
Memory: 184.6M
CPU: 17.436s
CGroup: /system.slice/grafana-server.service
        480 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfil
```

Figura 38 Arranque de Grafana (status).

Se comprueba que Grafana es accesible desde el navegador web, localhost:3000, Figura (38).

Figura 39 Inicio de sesión.

Cuando se inicia Grafana por primera vez, se debe ingresar el usuario y la contraseña admin. En seguida, se debe cambiar la contraseña de administrador y con eso se abrirá la pantalla principal de Grafana, (Figura 39).

Para que Grafana pueda generar gráficos a partir de una base de datos, se necesita definir una fuente externa que apunte a esa fuente, Influx en este caso (Figura 40 y 41).

Figura 40 Home Grafana configuración

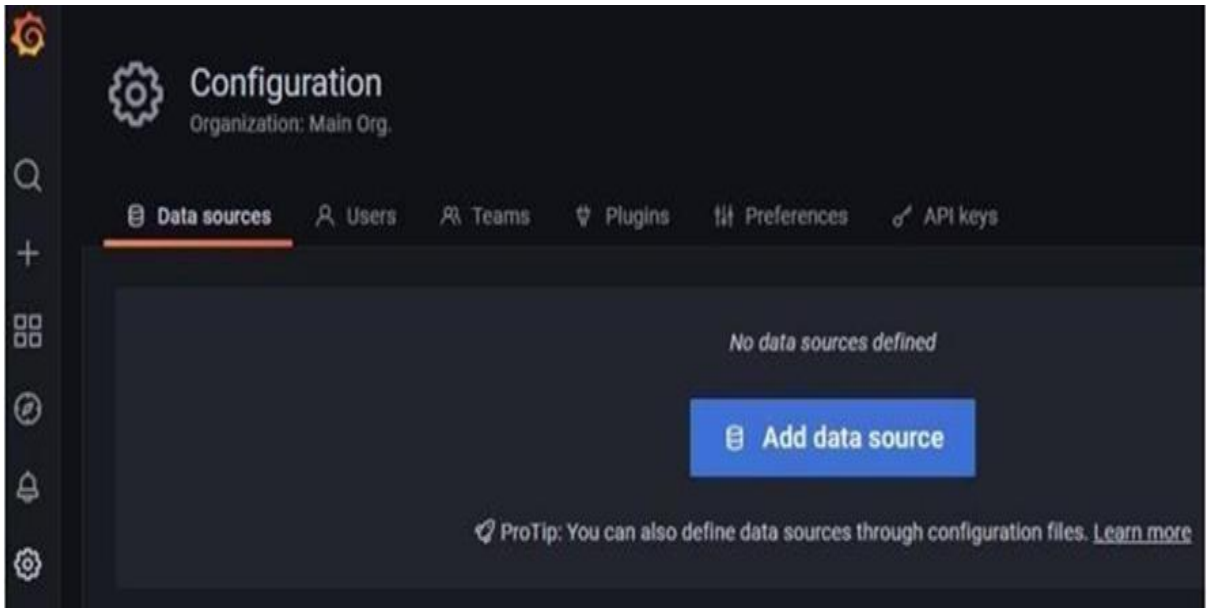

Figura 41 Data Source.

Posteriormente, se añade el data Source Influxdb (Figura 42).

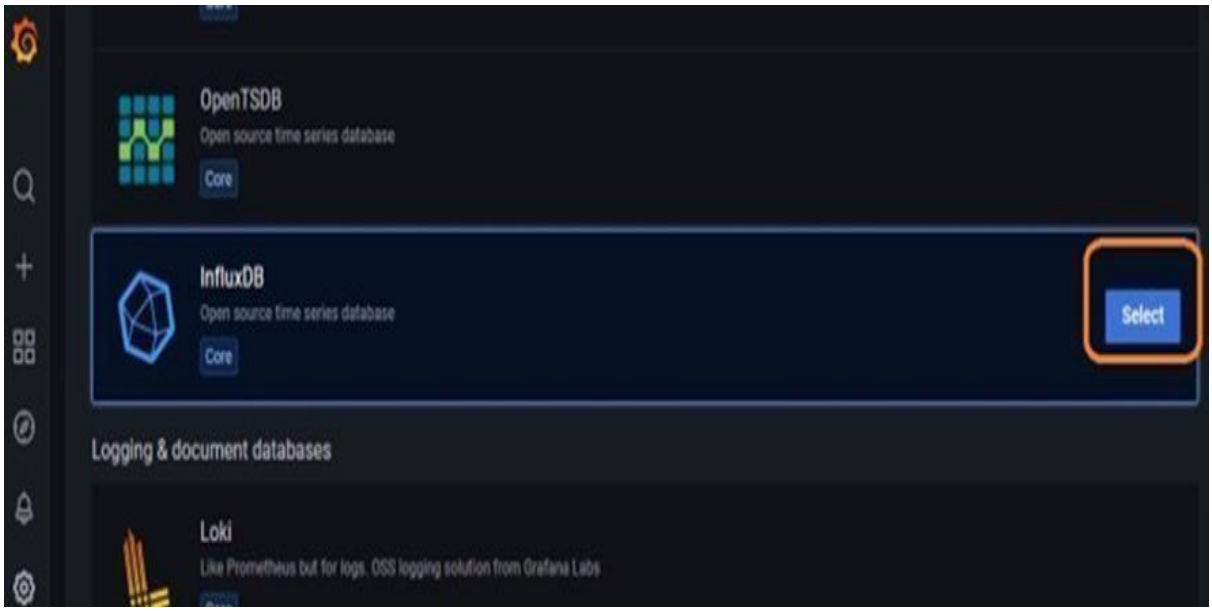

Figura 42 Base de Datos influxdb.

Una vez selecciona la información de Influxdb, se establece la dirección y puerto el tipo de BD, (Figura 43).

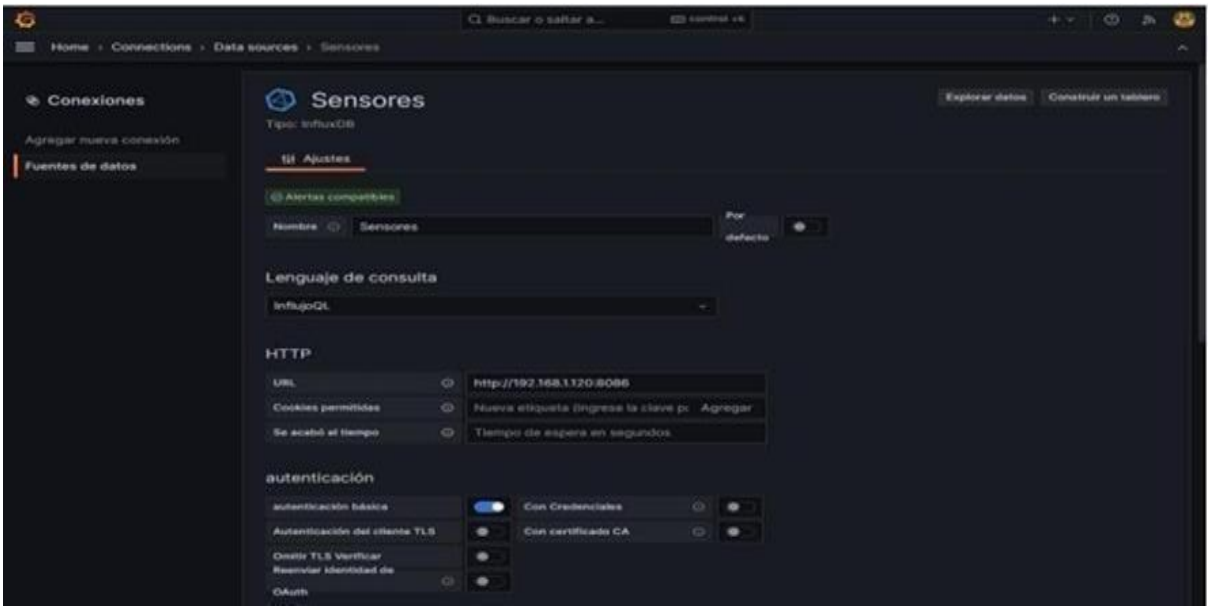

Figura 43 Configuración http.

En la URL está la dirección IP del server Grafana, y la autentificación elegida es Basic “Auth, with credentials”, esto debido a que se fijó un nombre de usuario y contraseña para acceder.

Para acceder a la base de datos, es necesario proporcionar la información de conexión que se detalla a continuación. En esta sección se especifica el nombre de la base de datos a la que se desea acceder, así como las credenciales (usuario y contraseña) con permisos de lectura, las cuales pueden diferir de las utilizadas para acceder al servidor.

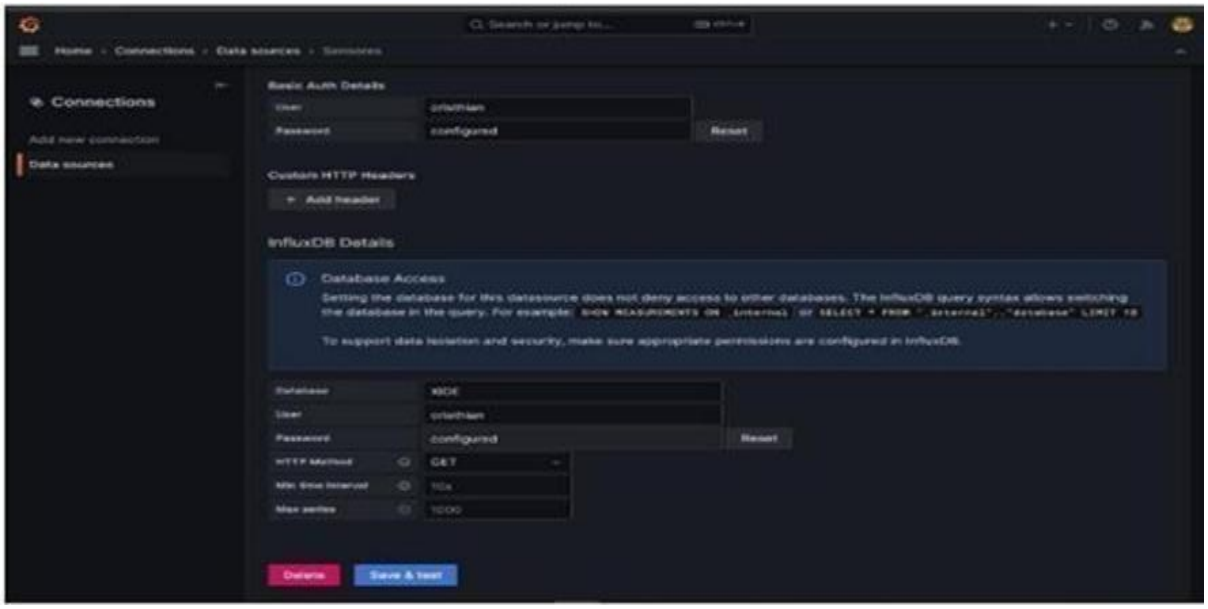

Figura 44 Configuración base.

Terminando esto, solo queda guardar la configuración, donde muestra un mensaje como el de la Figura 45.

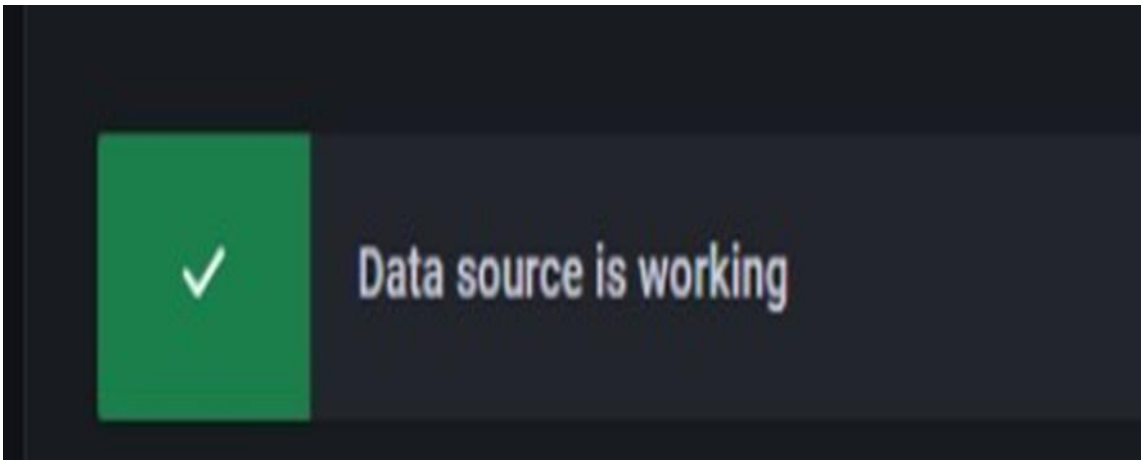

Figura 45 Configuración exitosa.

Tras confirmar el acceso a InfluxDB, se emplea la función "Explore" de Grafana para validar que los datos almacenados en la base de datos dbSensor pueden ser consultados correctamente.

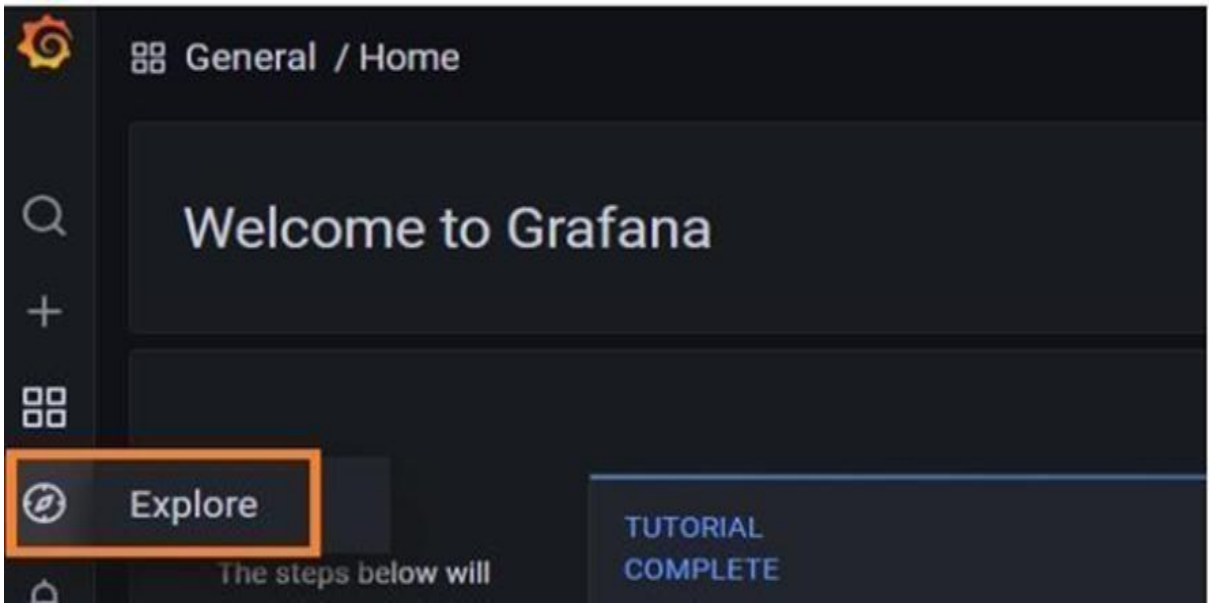

Figura 46 Explore.

Una vez configurada la consulta con los datos correspondientes, se mostrará una interfaz gráfica que permite acceder manualmente a la información almacenada en la base de datos. El primer paso consiste en seleccionar la versión de InfluxDB desde el primer menú desplegable; en este caso, la base de datos fue registrada con el nombre Sensores.

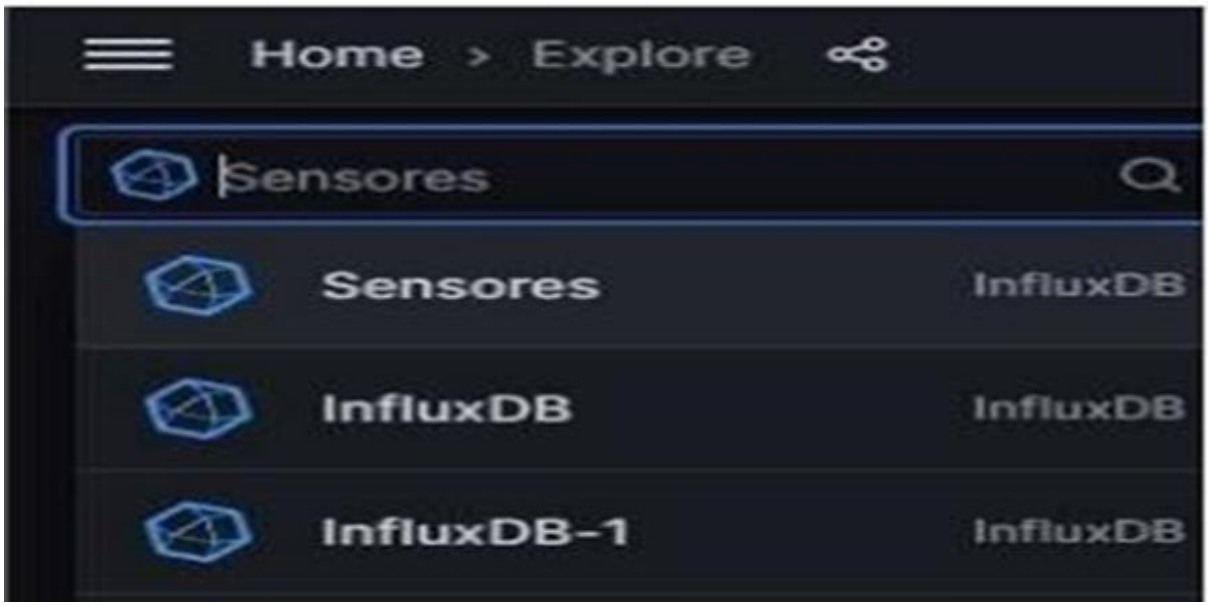

Figura 47 Versión de la base

En el apartado de select measurements se encuentran las medidas disponibles, donde se pueden mostrar los valores guardados de (los sensores los cuales se encuentran en mqtt_consumer, ver Figura 48).

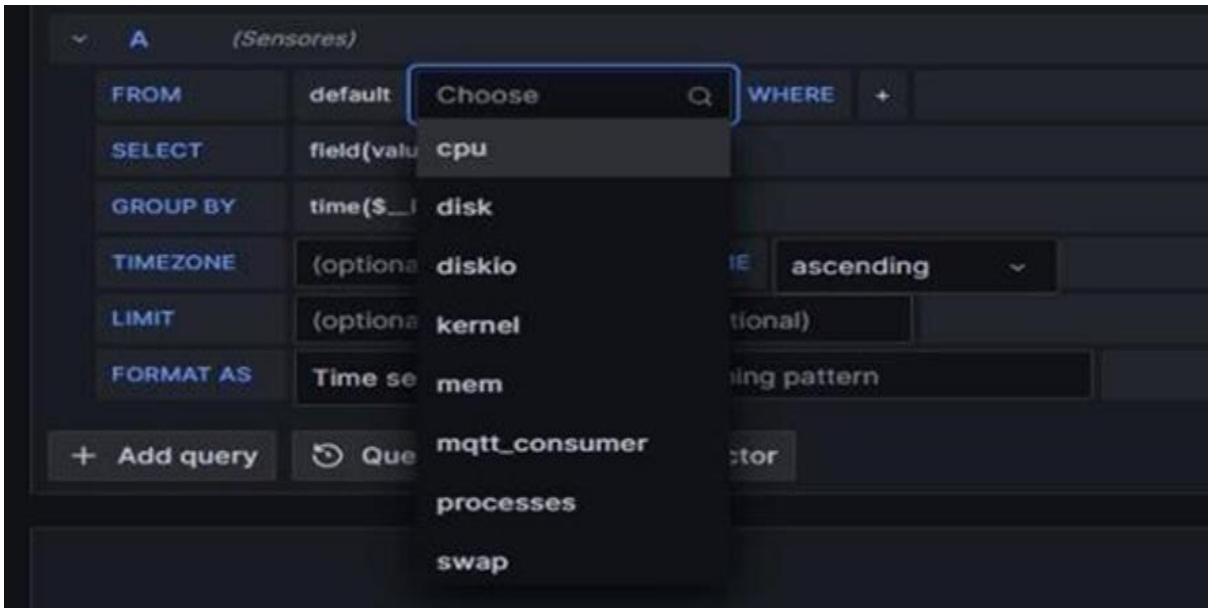

Figura 48 C.

En este punto, es posible realizar la selección utilizando un topic tag, previamente configurado para identificar la ubicación del sensor. En este caso, existen dos ubicaciones disponibles: Pasillo y LEL (Laboratorio de Electrónica Libre).

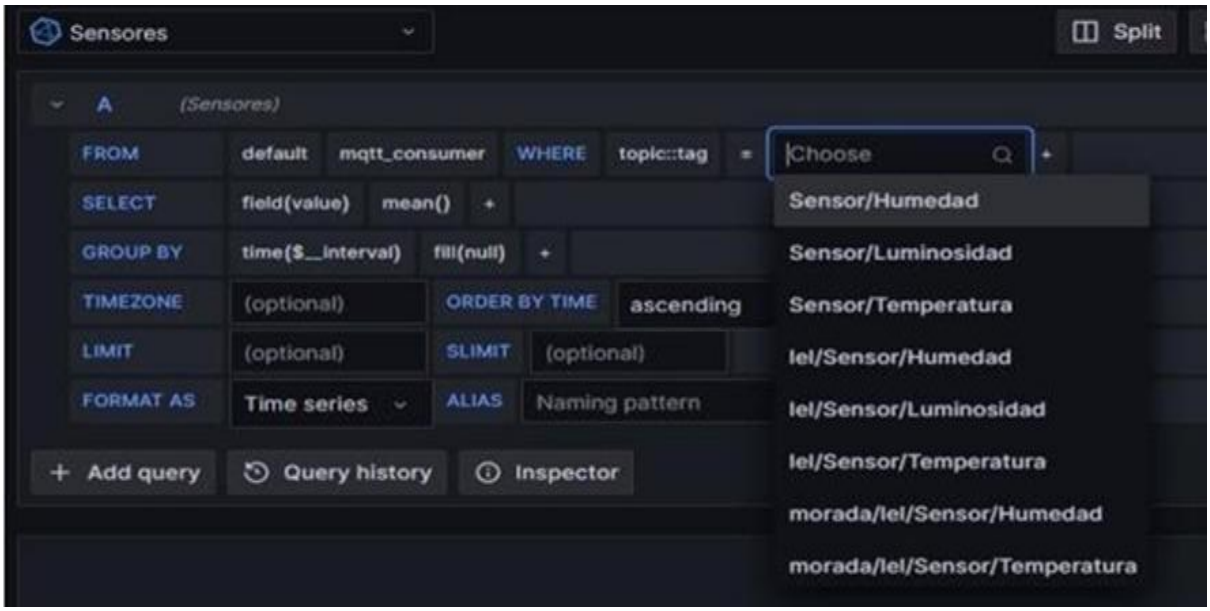

Figura 49 identificador de topic: tag.

Finalmente se pueden ver gráficos de los valores almacenados en la base de datos, (Figura 50).

Figura 50 Grafico de datos almacenados

Una vez instalado Grafana, lo que sigue es configurar alguna salida grafica que ayude a visualizar y supervisar las medidas de los sensores. Para eso se crean gráficos que presentan la concentración de temperatura, humedad y luminosidad (obtenidos con el dispositivo ESP32).

El software Grafana permite crear Dashboards (cuadros de instrumentos) y paneles dentro del mismo Dasboard con diferentes secciones.

A cada uno de los instrumentos que componen un panel, Grafana le otorga el nombre de panel y permite combinar todo un conjunto de ellos (auque sean de diferentes tipos).

Con estas ventajas se puede crear un primer Dashboards e ir añadiéndole paneles, (Figura 51).

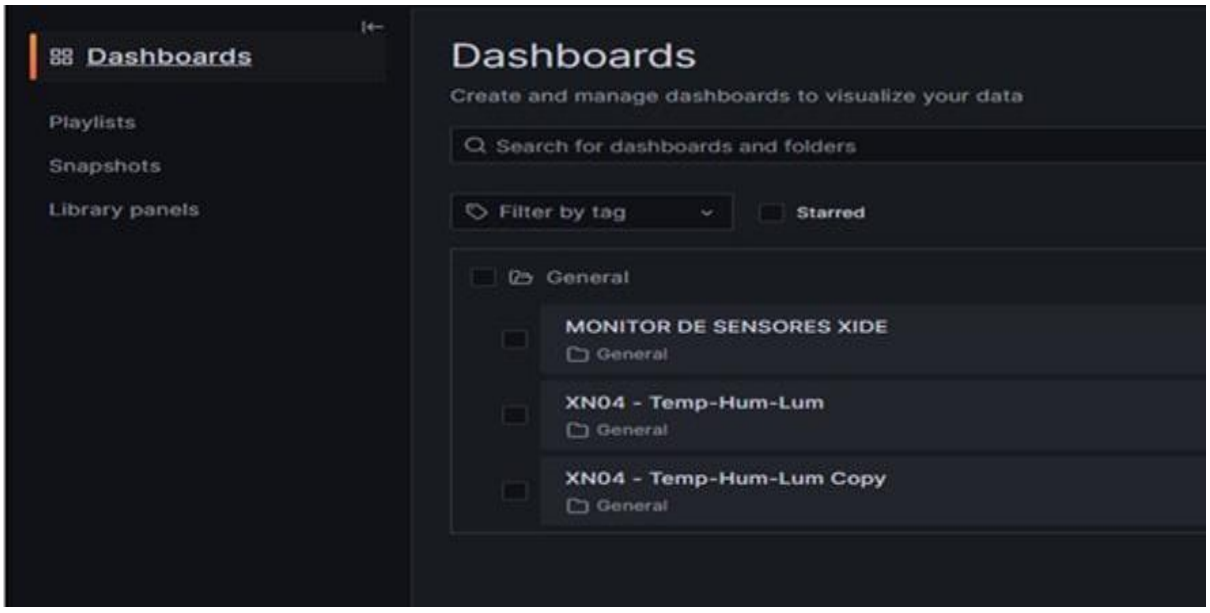

Figura 51 Creación de panel.

Al acceder a esta sección, se mostrará una pantalla con los dashboards previamente creados. Si se selecciona el botón azul New Dashboard, se puede crear uno

personalizado. Una vez creado, se accede a una pantalla en blanco desde la cual es posible agregar paneles correspondientes a cualquier sensor disponible.

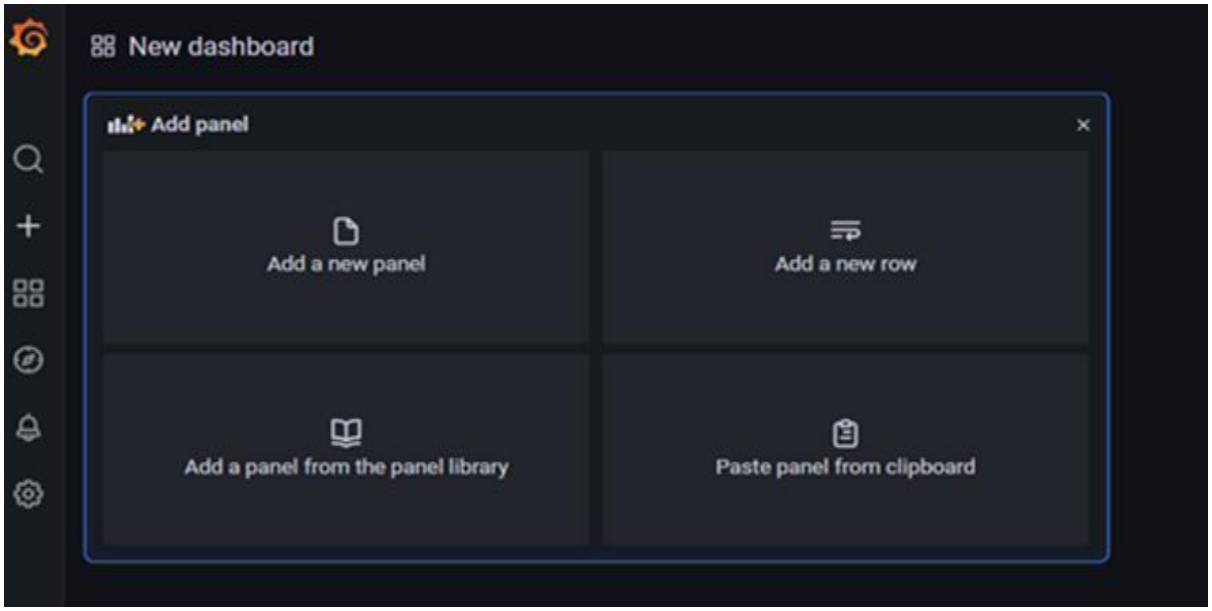

Figura 52 Nuevo panel.

Finalmente, al organizar los paneles creados se tiene un tablero como el de la Figura 53.

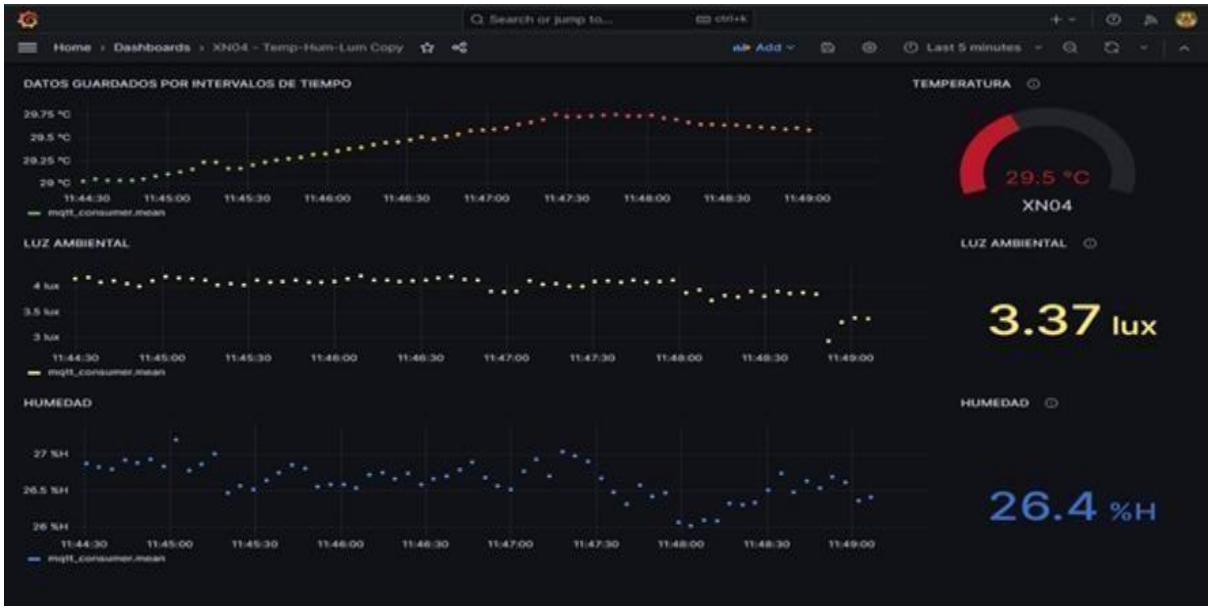

Figura 53 , Panel de control y monitoreo de sensores.

Capítulo 3. Selección del nodo a utilizar.

3.1. Plataforma de desarrollo XIDE.

XIDE es un kit de hardware integrado por módulos X-NODE y tarjetas de expansión X- BOARD (Figura 54).

El X-NODE es un módulo integrado por un sensor, actuador o dispositivo de comunicación y un controlador en hardware, cuenta con el estándar mikroBUS, que lo hace compatible con un gran ecosistema de kits para desarrollo de hardware.

Las X-BOARD son tarjetas de expansión para la evaluación rápida y sencilla de proyectos orientados al IoT, cuentan con diferentes conectores y puertos estándar mikroBUS para compatibilidad con módulos X-NODE y Click Boards. También integran conectores JST compatibles con el estándar de conexión Qwiic de SparkFun, que brinda una comunicación entre diversos módulos y tarjetas de desarrollo por medio del protocolo I2C.

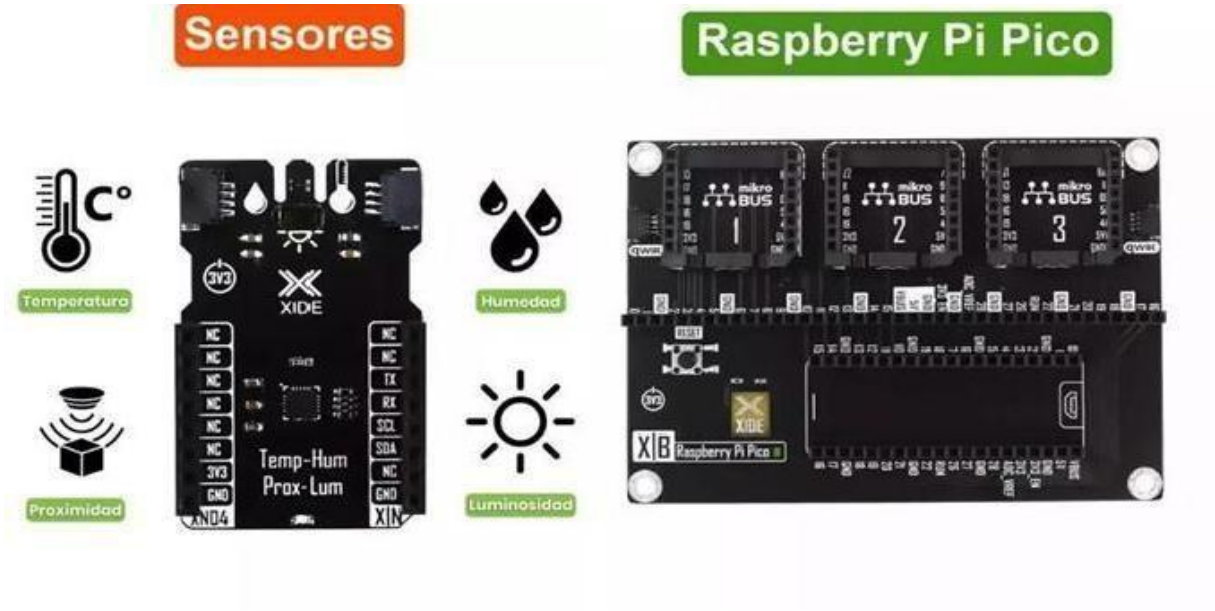

Figura 54 X-NODE y X-BOARD

3.1.1. Consumo de energía.

El consumo de la placa de X-Node (basada en ESP32) es variable, dependiendo de si está en modo activo o reposo. El, consumo, en promedio, del ESP32 es:

- **Modo de reposo profundo (Deep Sleep)**: Menos de 10 µA.
- **Modo activo (Transmitiendo Wi-Fi/Bluetooth)**: Alrededor de 160 mA (esto puede variar dependiendo de la actividad).

En este caso se eligió la alimentación pasiva ya que solo proporcionara energía cuando se usa el sistema, así se logra conservar carga de la batería por más tiempo. Para este trabajo, no fue necesario seleccionar la alimentación activa, debido a que el sistema incluye reposo.

Para este trabajo, el sistema se alimenta mediante una batería, por lo cual primero se necesita calcular el consumo total de todos los componentes, incluyendo los sensores y la placa X-Node. De la hoja de especificaciones, se obtiene que el consumo máximo de todos los sensores es de aproximadamente, 1.09 mA y el consumo máximo de la placa ESP32 es de 160 mA.

Por lo que el sistema en general requiere 162 mA aproximadamente. Respecto a la batería se considera lo siguiente:

Batería (modelo 18650) de 3.7 volts con una capacidad de 2000 mAh.

Para calcular el tiempo de duración de la batería, se realiza lo siguiente:

$$\frac{2000\ [mAh]}{162\ [mA]} = 12.34\ Hrs$$

Para este caso que se usa un sistema en modo pasivo, la duración podría ser mucho mayor, dependiendo del tiempo del reposo entre lecturas.

3.1.2 Procesamiento.

El procesamiento de datos para el sistema IoT se realiza en tres etapas, como se muestra en la Figura 55.

Figura 55 , Diagrama de Bloques para el procesamiento de datos.

- **Sensores**: Estos dispositivos IoT obtienen la información del entorno y generan una señal adecuada para cada variable de interés.
- **Transporte**: Una vez que los datos son obtenidos, son enviados de manera segura desde el dispositivo IoT hacia una plataforma o servidor donde serán procesados. Esto se realiza utilizando protocolos de comunicación que facilitan la transferencia de los datos de manera rápida, segura y confiable.
- **Visualización**: Este proceso representa la información final de las variables mostradas en un tablero de manera gráfica.

3.1.3 Raspberry Pi Pico H.

La Raspberry Pi Pico H es una placa de microcontrolador compacta y de bajo costo, basada en el chip RP2040 (Figura 56). Es el mismo microprocesador que se encuentra en la Raspberry Pi Pico original.

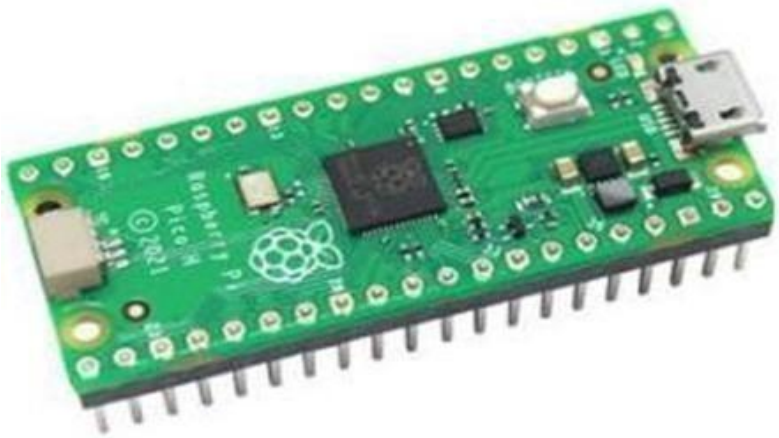

Figura 56 Raspberry Pi Pico H

Características de la Raspberry Pi Pico H:

- **Microcontrolador RP2040**: Procesador ARM Cortex-M0+ de doble núcleo con una velocidad de reloj de hasta 133 MHz.
- **Memoria**: 264 KB de SRAM y 2 MB de almacenamiento flash.
- **Puertos de I/O**: 26 pines GPIO multifuncionales, que se pueden usar para diversas tareas como I/O digital, PWM, ADC, entre otras. Estos pines se pueden utilizar como entradas o salidas digitales.
- **Conectividad**: Soporte USB 1.1 como host/dispositivo, programación a través de micro-USB.
- **Suministro de energía**: Puede ser alimentada a través de USB o fuentes de energía externas con un rango de 1.8V a 5.5V.
- Los pines de encabezado pre-soldados facilitan la conexión directa de la Pico H a una placa de pruebas o a otro hardware de prototipo, permitiendo conexiones más rápidas y fiables [29].

3.1.4 Tarjeta electrónica ESP32.

El dispositivo ESP32 es un microcontrolador versátil y potente de la marca Espressif Systems, tiene un procesador de doble núcleo, capacidades integradas de Wi-Fi y Bluetooth, además de una amplia gama de aplicaciones. .

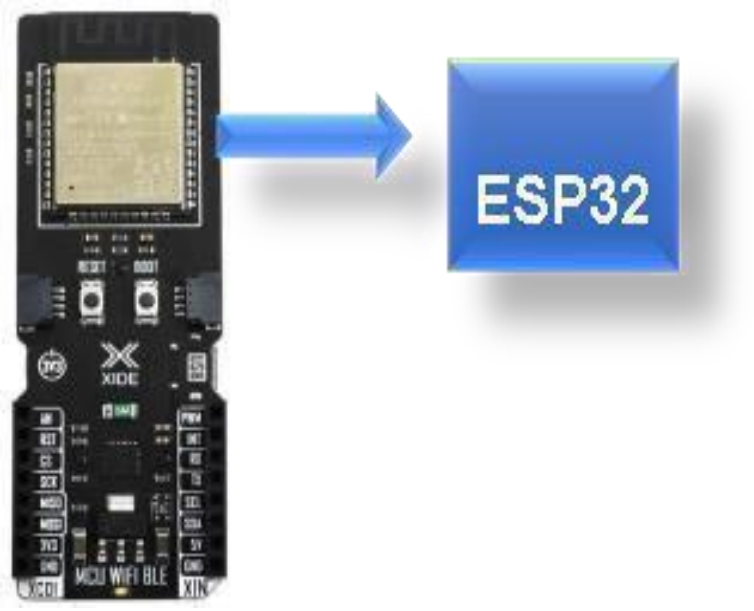

Figura 57 Dispositivo de comunicación ESP32

Características:

- **Procesador de Doble Núcleo**: Normalmente cuenta con un CPU Xtensa LX6 de 32 bits, que puede funcionar a hasta 240 MHz.
- **Wi-Fi y Bluetooth**: Soporta Wi-Fi de 2.4 GHz (802.11 b/g/n) y Bluetooth (Classic y BLE).
- **Puertos de Entrada/Salida (I/O)**: Incluye una variedad de pines GPIO, ADCs (convertidores analógico-digital), DACs (convertidores digital-analógico), temporizadores y más.
- **Memoria**: Generalmente tiene 520 KB de SRAM y hasta 16 MB de memoria flash externo.

- **Gestión de Energía**: Ofrece varios modos de suspensión para optimizar el consumo de energía en dispositivos que funcionan con batería.
- El dispositivo ESP32 se puede programar usando el IDE de Arduino, el IDF (IoT Development Framework) de Espressif, o MicroPython, entre otras herramientas [30].

Para la realización de este trabajo se usa un dispositivo ESP32, que ya viene incluido en la placa X-NODE MCU WIFI BLE (ESP32-WROOM-32). Incluye un módulo de conectividad inalámbrica ESP32-WROOM-32 de Espressif Systems, que combina Bluetooth/BLE y Wi-Fi de 2.4 GHz. Este módulo tiene dos modos de operación, permitiendo su uso como un X-NODE (Esclavo) o como una X-BOARD (Maestro), y es compatible con los protocolos I2C, UART y SPl.

También dispone de varios modos de potencia y ajuste dinámico del consumo energético, lo que ayuda a lograr un equilibrio entre el alcance de comunicación, la velocidad de transmisión de datos y el uso de energía. Es adecuado para aplicaciones de baja potencia, dispositivos móviles, electrónica portátil e Internet de las cosas (IoT), permitiendo la interacción con sensores, actuadores o sistemas de domótica, así como el monitoreo y control desde una computadora o un servicio en la nube.

3.2 Lenguaje de programación.

3.2.1 MicroPython.

MicroPython es una versión reducida del lenguaje de programación Python. Diseñada específicamente para funcionar en microcontroladores y entornos con recursos limitados. A diferencia de Python estándar, que se usa en PCs y

servidores, MicroPython está optimizado para ejecutarse en dispositivos con capacidades de procesamiento y memoria mucho más limitadas.

Características Principales de MicroPython.

- **Lenguaje Python.**
- **Optimización para Microcontroladores**: Reduce el uso de memoria y recursos del sistema, permitiendo que Python se ejecute en hardware con limitaciones.
- **Bibliotecas Integradas**: Incluye bibliotecas específicas para acceder a hardware y periféricos como GPIO, I2C, SPI, y UART, facilitando la programación de dispositivos electrónicos.
- **Interactividad**: Ofrece un REPL (Read-Eval-Print Loop) interactivo, permitiendo probar código en tiempo real directamente en el dispositivo [31].

Para este trabajo, se emplea MicroPython para la programación de la Raspberry Pi Pico H, en conjunto con el entorno de desarrollo (Thonny), que es de código abierto el cual cuenta ya con las herramientas, bibliotecas y dependencias que se necesitan para este proyecto.

3.2.2 IDE de programación para ESP32.

Para el desarrollo de aplicaciones de Internet de las Cosas IoT basadas en el microcontrolador ESP32, se utiliza el entorno de desarrollo Arduino IDE, una plataforma de código abierto este puede ser muy versatilidad y compatible con una gran variedad de dispositivos electrónicos. Este entorno permite escribir, compilar y cargar programas en el ESP32 utilizando los lenguajes C y C++, facilitando la integración de sensores, actuadores y módulos de comunicación inalámbrica.

Características principales del Arduino IDE para IoT:

- **Lenguaje C/C++ Simplificado**: Permite desarrollar código eficiente y directo para microcontroladores, adecuado para proyectos de IoT que requieren control preciso de hardware.
- **Compatibilidad con ESP32**: Soporta completamente el chip ESP32 mediante el núcleo proporcionado por Espressif, incluyendo Wi-Fi, Bluetooth, y periféricos como ADC, PWM, I2C y SPI.
- **Bibliotecas para Sensores y Comunicación**: Incluye soporte para una amplia gama de sensores (temperatura, humedad, movimiento, entre otros.) y módulos de comunicación como Wi-Fi, MQTT, HTTP y Bluetooth, esenciales para sistemas IoT.
- **Monitor Serial Integrado**: Ofrece herramientas de depuración como el monitor serial y el plotter de datos, útiles para visualizar el comportamiento de los sensores en tiempo real.
- **Entorno Multiplataforma**: Compatible con sistemas operativos como Windows, Linux y macOS, y con posibilidad de integrar otras herramientas como PlatformIO. [32]

En este trabajo, el Arduino IDE se utiliza para programar el ESP32, permitiendo la adquisición de datos desde distintos sensores conectados al dispositivo y su posterior transmisión mediante protocolos como MQTT a una red local o a la nube.

3.3 Sensores.

3.3.1 Sensor de luz.

Un sensor de luz es un transductor que convierte la energía lumínica en una señal eléctrica o serial. Tiene la capacidad de detectar cambios en la iluminación de un lugar específico. Esta iluminación es la que el ojo humano puede captar [33].

Algunas de las características de los sensores de luz son:

- Son muy resistentes así que tienen larga vida de utilidad.
- No requieren de mantenimiento.

El sensor que se utiliza para este trabajo es el X-NODE XN04, el cual tiene las siguientes características:

- Sensor de luz ambiental con detección de 0.0022 a 73000 lx, supresión de ruido a 50 Hz/60 Hz y consumo típico de 90µA.
- Puertos de comunicación: UART, I2C
- Compatible con el estándar mikroBUS y estándar Qwiic.
- Voltaje: 3.3 volts
- Dimensiones del módulo: 41 mm x 25.4 mm x 20.5 mm [35].

Este tipo de sensor generalmente es activo, esto quiere decir que mide la cantidad de luz detectada en luxes. Otra de sus ventajas es que entrega valores digitales a través de un protocolo de comunicación serial I2C.

Lo que significa que el microcontrolador puede leer directamente el valor de luminosidad sin necesidad de conversión adicional.

3.3.2 Sensor de humedad.

Un sensor de humedad es un transductor electrónico creado para medir y cuantificar la cantidad de humedad relativa en el entorno. Su función principal es detectar la presencia de vapor de agua en el aire o en un material específico, proporcionando información sobre las condiciones ambientales.

Cuando la humedad ambiental incrementa, el material dieléctrico absorbe agua y su constante dieléctrica se modifica, lo que afecta la capacitancia del sistema. Este cambio en la capacitancia genera una señal eléctrica que puede ser convertida en una medición de humedad [34].

Algunas de sus características son:

- Tienen la capacidad de cambiar su capacitancia eléctrica.
- Son rápidos debido a que tienen una respuesta inmediata.

Utiliza un sensor capacitivo para medir la humedad y un termistor para medir la temperatura. La forma de recibir los valores detectados es de forma binaria y a la hora de entregar los valores recolectados, es de forma digital, en forma de un valor binario que representa la humedad y la temperatura. La información de humedad se mide en porcentaje (%) relativo de un 0% a 100%.

3.3.3 Sensor de temperatura.

Los sensores de temperatura son transductores que permiten medir la temperatura a través de un cambio en sus propiedades eléctricas. En específico, mediante un cambio su resistencia. También se les conoce como sensores de calor o termo sensores. Estos dispositivos se utilizan, entre otras cosas, para el control de dispositivos cotidianos como: planchas, tostadoras, sensores de movimiento,

herramientas como soldadores con regulación automática de temperatura. También tienen aplicaciones industriales donde se supervisan circuitos que dependen de la temperatura, como cámaras de radiación térmica y sistemas de calefacción automatizados. Una aplicación común es la protección contra sobrecalentamiento de microprocesadores.

Algunas de sus características de sensores son:

- Tienen rangos grandes de medición.
- Su precisión es muy buena.
- Su respuesta de tiempo es rápida.
- Son sensores que tienen un tiempo de vida útil amplio.

La X-NODE incluye un sensor de temperatura con las siguientes características:

- Sensor de temperatura con detección de 0 °C a 85 °C con una precisión típica de +/- 0.2 °C y +/- 1.8% RH, calibración automática, consumo de 0.5 mA y alta relación señal/ruido.
- Puertos de comunicación: UART, I2C
- Compatible con el estándar mikroBUS y estándar Qwiic.
- Voltaje: 3.3 volts
- Dimensiones del módulo: 41 mm x 25.4 mm x 20.5 mm [35].

Este sensor también suele ser un sensor activo, la salida es completamente digital, el microcontrolador recibe un valor digital que representa la temperatura, que debe ser procesado para convertirlo en grados Celsius.

3.4 Comunicación con la red local e Internet.

3.4.1 Comunicación inalámbrica SIM y eSIM.

El módulo X-NODE LTE GNSS (SIM7080G) integra un módulo de conectividad celular SIM7080G de SIMCom. que brinda la comunicación cuenta con las siguientes características:

- Posicionamiento GNSS: Incluye soporte para GPS, GLONASS, BeiDou y Galileo, permitiendo una geolocalización precisa.
- También cuenta con una eSIM - MFF2 precargada con 50 MB para un uso inmediato, además de una bandeja para microSIM - 3FF (solo es posible utilizar una SIM a la vez)
- Una antena LTE YMLR001 y un adaptador de SMA a U.FL. Es ideal para la integración en dispositivos móviles portátiles alimentados por baterías, proyectos de IoT, Industria 4.0 y aplicaciones M2M que requieran baja latencia como medición, seguimiento de activos, monitoreo remoto, E-Health, terminales móviles [36].

Este módulo ofrece una comunicación por medio de una red móvil SIM o eSIM a través de sus datos, por lo que una de sus desventajas es que, terminado el paquete que ofrece, habría que estar realizando recargas ya que cuenta con un límite de datos.

3.4.2 Wi-Fi.

El Wi-Fi permite que los dispositivos inteligentes se conecten a la red y se comuniquen entre sí, para el intercambio de datos o bien conectarse a un punto de

acceso de red inalámbrica, pudiendo tener así conexión a Internet. Esto permite establecer una conexión inalámbrica entre dispositivos como:

- Computadoras.
- Consolas de videojuego.
- Smart Tv.
- Teléfonos celulares.
- Entre otros.

Para este trabajo se utiliza el módulo El X-NODE MCU WIFI BLE (ESP32-WROOM-32) es un módulo de comunicación inalámbrica que integra las siguientes características de Wi-Fi:

- **Compatibilidad con estándares Wi-Fi**: Soporta los protocolos 802.11 b/g/n/e/i en la banda de 2.4 GHz, permitiendo una amplia compatibilidad con redes Wi-Fi existentes.
- **Velocidades de transmisión**: Ofrece velocidades de hasta 150 Mbps en el estándar 802.11n, facilitando la transmisión de datos de manera eficiente [37].

3.5 Implementación de la plataforma y nodos.

3.5.1 Configuración de los nodos.

Los nodos requieren ser programados para establecer los protocolos a utilizar, las direcciones de los servidores a los que se conectan y de ser necesario las credenciales de acceso.

3.5.2 Programación del nodo.

Para verificar una comunicación con el nodo, se envía un comando al dispositivo externo (módulo XIDE) a través de UART.

A continuación, se muestran fragmentos de código que permiten realizar tareas asignadas a cada nodo.

Posteriormente se envía un comando para obtener el valor de la temperatura, luminosidad y humedad de los sensores, se procesa la respuesta y se publica en el MQTT.

Estos mismos pasos se realizan para el sensor de luminosidad y de humedad solo cambia el término “GL y GH”, esto indica que cada uno tiene su código de identificación, para detectar cada sensor. Esta programación de cada sensor es mediante C.

En la Tabla 8 se muestran algunos de los comandos comúnmente empleados en la comunicación con el nodo.

Tabla 8 Comandos para la comunicación con el nodo.

XN04A?<CR+LF>	Verifica si se estableció una comunicación con éxito. **Respuesta:** OK<CR+LF>
XN04A+V<CR+LF>	Obtiene la versión del firmware actual que integra el X-NODE. **Respuesta:** XN04A=Versión<CR+LF> **Ejemplo:** XN04A=0.1<CR+LF>
XN04A+ID= **(A-Z)<CR+LF>**	Cambia el index del ID por una letra diferente del abecedario de la A a la Z, la nueva letra debe ser en mayúscula. Una vez modificado, para volver a cambiarlo es necesario colocar el ID con el nuevo index. **Respuesta:** OK<CR+LF> **Ejemplo de envío:** XN04C+ID=H<CR+LF>
XN04A+GT<CR+LF>	Obtiene el valor de Temperatura del sensor. Retorna con el valor final en °C con dos decimales. **Respuesta:** XN04A=VAL<CR+LF> **Ejemplo:** XN04A=27.81<CR+LF>
XN04A+GH<CR+LF>	Obtiene el valor de Humedad del sensor. Retorna con un valor relativo (%) de humedad. **Respuesta:** XN04A=VAL<CR+LF> **Ejemplo:** XN04A=80<CR+LF>
XN04A+GL<CR+LF>	Obtiene el valor de Luminosidad ambiental del sensor. Retorna con un valor final en Luxes. **Respuesta:** XN04A=VAL<CR+LF> **Ejemplo:** XN04A=316<CR+LF>
XN04A+GP<CR+LF>	Obtiene el valor de Proximidad del sensor. Retorna con un valor relativo de proximidad. **Respuesta:** XN04A=VAL<CR+LF> **Ejemplo:** XN04A=198<CR+LF>

3.5.3 Nodos en el sistema.

Para gregar otros sensores u otros nodos se tienen que desarrollar las relaciones entre tópico y lugar, así como tópico y nodo. Para ello se requiere de la modificación en la programación del nodo a agregar, además de indicar las relaciones entre tópicos en los archivos de configuración.

Para que el usuario pueda observar los datos obtenidos debe realizar una suscripción al Tópico principal, dbSensor. Este cuenta con dos subtopicos de trabajo LEL y Pasillo, donde se encuentra la información de los sensores (Figura 58).

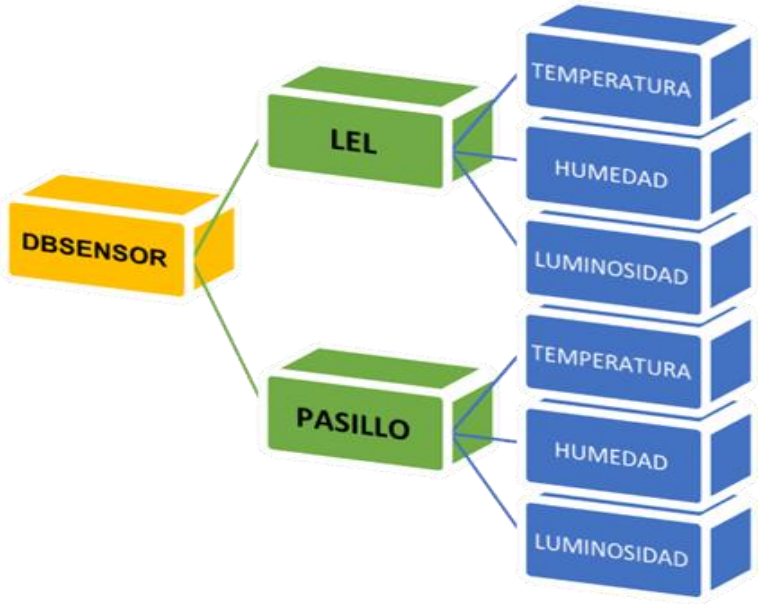

Figura 58 Relaciones entre tópicos.

La arquitectura de este trabajo cuenta con un nodo, que es el que se encarga de monitorear el laboratorio de electrónica (LEL), para añadir más nodos es necesario modificar el código de la tarjeta ESP32.

Este proceso se lleva a cabo mediante el uso de una segunda tarjeta, la cual realiza una nueva suscripción a los tópicos del sistema MQTT.

- client.publish("lel/Sensor/Temperatura", str1.c_str());

En esta línea de código, si se desea incorporar otro nodo al sistema, se debe modificar el sub-tópico (por ejemplo, reemplazar "LEL" por "Pasillo") de acuerdo con el lugar desde el cual se desea obtener la información.

3.5.4 Sensores en un nodo.

Para agregar otros sensores a los nodos se requiere que cumplan con las características de los buses utilizados para la comunicación; de forma serial o con el protocolo I2C. Si se requieren agregar sensores al nodo se necesita seleccionar la tarjeta con la que se está trabajando, por ejemplo:

Tabla 9 Puertos disponibles.

Raspberry Pi Pico	X-Node
Puertos Disponibles	Puertos disponibles
I2C.	I2C
UART	UART
SPI	SPI
	GPIO
	ADC

Para agregar sensores al nodo que se elige es necesario que el sensor sea compatible con alguna de estas tecnologías, por ejemplo, en la Raspberry Pi Pico contiene puertos de entrada y salida como I2C, UART, SPI, por otra parte, si se quisiera agregar al nodo la placa X-Node, se tiene que tener en cuenta que esta tarjeta trabaja mediante su

puerto Qwiic, por lo que los sensores deben ser compatibles y también el lenguaje de programación.

Capítulo 4

Resultados.

4 Resultados.

A partir del desarrollo de este proyecto, enfocado en la comparación entre una plataforma comercial y una solución basada en software libre, para la recolección y visualización de datos IoT, obtiene una solución basada en herramientas de código abierto demostrando ser eficiente frente a la plataforma comercial, siendo esta una buena alternativa con recursos limitados y la capacidad de personalización y control sobre el flujo de datos.

Si bien la solución basada en software libre implica una inversión inicial mayor en términos de tiempo de configuración y formación técnica, se comprobó que a mediano y largo plazo representa una reducción significativa en costos operativos, eliminando pagos por licenciamiento y dependencia de almacenamiento en la nube.

4.1 Datos recolectados.

Los paneles se organizan de la siguiente manera.

- Datos del software libre.

Figura 59 se observa la recolección de datos referente a la Humedad relativa.

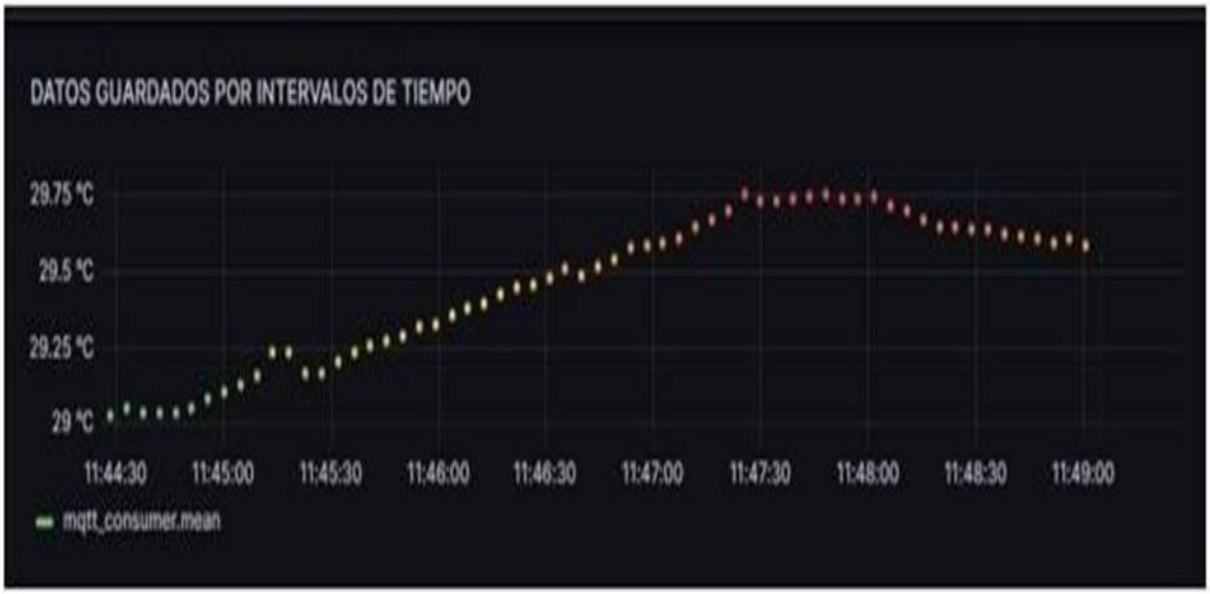

Figura 60 se observa la recolección de datos refente a la temperatura.

Figura 61 se observa la recolección de datos referente a la luz ambiental.

En las figuras 63-65 se puede observar en la parte horizontal que los intervalos de cada monitoreo se realizan cada 30 segundos. El intervalo de monitoreo se puede ajustar en el software al valor que se desee, mientras que en la parte vertical muestra el rango de datos según el sensor, esto puede ser temperatura, humedad o grado de intensidad lumínica según sea el sensor que se requiera analizar.

- Datos del software comercial.

  En la Figura 62 se observará la recolección de datos referente a la Temperatura.

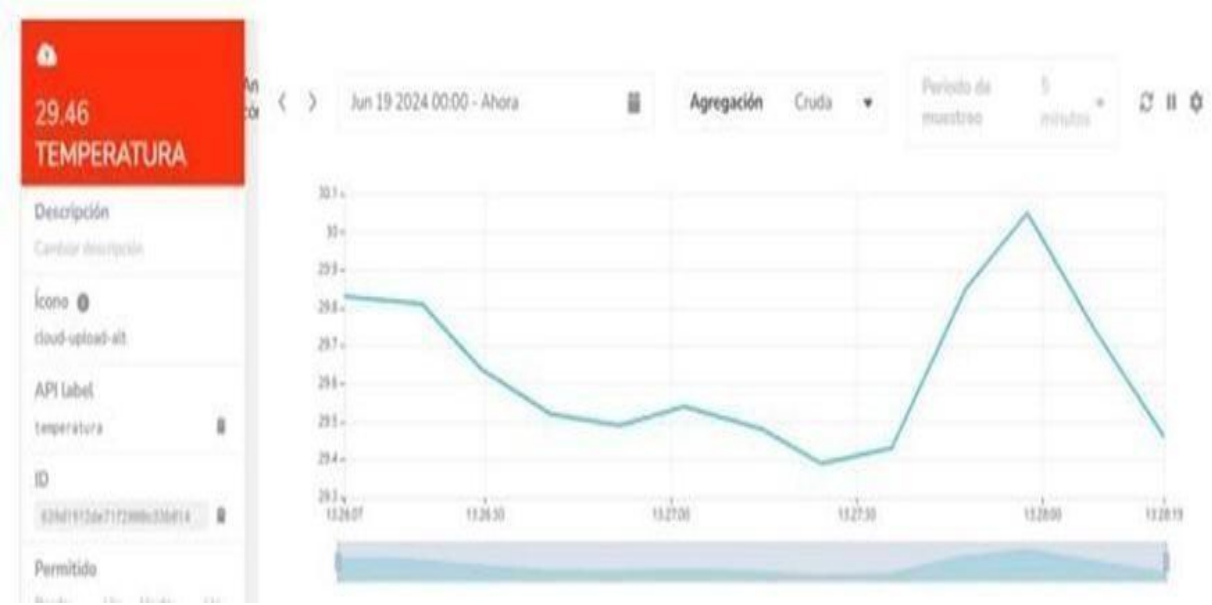

Figura 62 Datos recolectados de Temperatura

En la Figura 63, se observará la recolección de datos referente a la Humedad relativa.

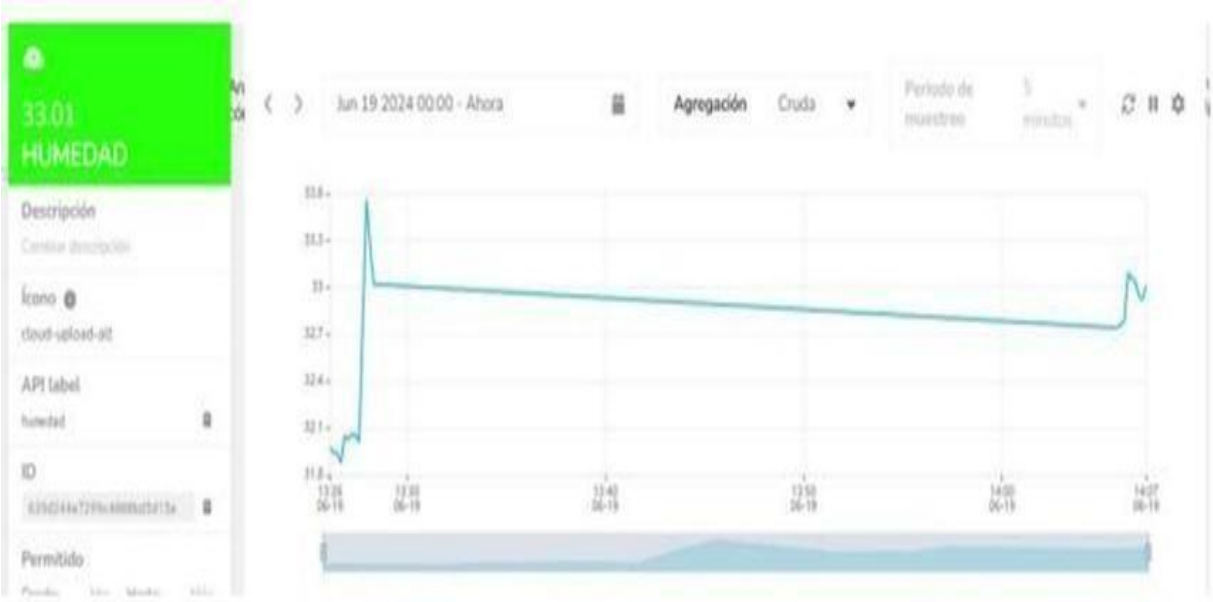

Figura 63 Datos recolectados de Humedad

En la Figura 64, se observa la recolección de datos referente a la Intensidad lumínica.

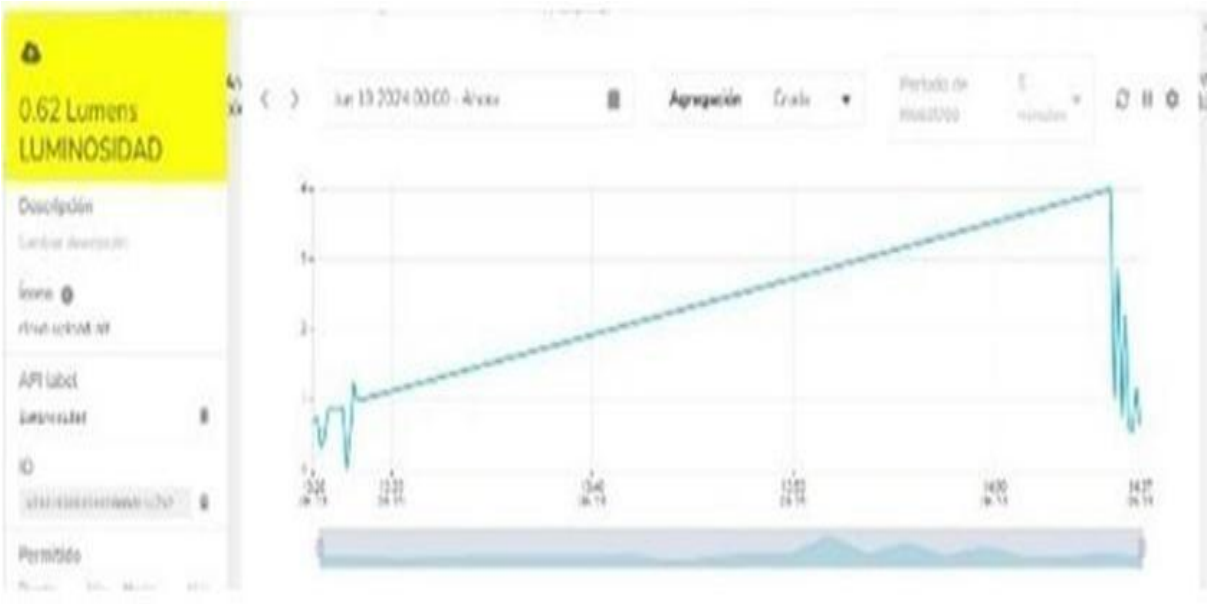

Figura 64 Datos recolectados de Luminosidad

En las figuras 66-68 se muestran los datos recolectados con el software de ubidots, en la parte horizontal se puede observar que el intervalo de monitoreo de los sensores es cada 30 segundos, en los que el sensor recolecta información, al valor que desee o en tiempo real, mientras que en la parte vertical se muestra el rango de datos según el sensor; las variables son temperatura, humedad o nivel de luminosidad, según sea el sensor que se requiera analizar.

4.2 Comparativa entre la implementación con software libre y la implementación en la nube.

En la Tabla 10 se presenta un cuadro comparativo entre la implementación del software Ubidots y el software Grafana.

Tabla 10 Cuadro comparativo

Aspecto	UBIDOTS	GRAFANA
Tipo	Plataforma IoT como servicio	Plataforma de visualización de datos
Uso principal	Recolección, almacenamiento, análisis y visualización de datos de sensores	Visualización y análisis de datos desde múltiples fuentes
Interfaz	Intuitiva y fácil de usar	Potente y flexible, pero con una curva de aprendizaje
Dashboards	Personalizable con widgets predefinidos	Altamente personalizables con una variedad de visualizaciones
Almacenamiento de datos	Almacenamiento en la nube para datos de sensores	No almacena datos; se conecta a diversas bases de datos (InfluxDB, Prometheus, etc.)
Alertas	Configuración de alertas basada en reglas	Alertas basadas en consultas de datos
APIs	RESTful y MQTT	Amplias opciones de APIs y plugins
Seguridad	Autenticación y cifrado de datos	Autenticación y permisos avanzados
Escalabilidad	Escalable para dispositivos y datos	Altamente escalable, adecuado para grandes volúmenes de datos
Facilidad de uso	Alta; configuración rápida y fácil	Requiere conocimiento técnico para configuración y conexión de datos
Soporte	Documentación completa y soporte en planes de pagos	Documentación extensa y comunidad activa; soporte en profesional y versiones empresariales
Costos	Plan gratuito limitado; planes pagos según características	Gratuito para la versión de código abierto; opciones de pagas (Grafana Enterprise) para características adicionales
Casos de uso	Ideal para proyectos de IoT de pequeña a mediana escala	Ideal para visualización avanzada y análisis en entornos grandes y técnicos

Tomando en cuenta las características, de la Tabla 10, se realizó una comparación más detallada sobre costos, los cuales son:

- Plataforma de software libre.

Tabla 11 Material para uso de plataforma de software libre. Precio establecido en pesos mexicanos.

Material	**Costo unitario**	**Cantidad**	**Unidad**	**Total**
• KIT XIDE IoT (2 modulos,tarjetas integradas)	3980	2	Paquete	$7960
• ModuloX-NODE MCU WIFI BLE (ESP32-WROOM	1280	2	Pza	$2560
• Panel solar mini (107x61 mm,200mA, 5V, 1W)	280	2	Pza	$560
• Bateria 3.3- 5 V	45	2	Pza	$90
• Modulo solar CN3065	240	4	Pza	$960
• Jumpers macho hembra	20	6	Pza	$120
• cables 22 awg	4	2	Metro	$8
				$12528

Teniendo un costo total de $12,528 en el material.

Para este trabajo los costos relacionados a recursos humanos como el programador, el ingeniero, el vendedor entre otros, no se toma en cuenta ya que es un trabajo que se está estructurado y realizado con fines educativos.

Sin embargo, es importante mencionar que el equipo de trabajo requerido para desarrollar un trabajo de estas características se requiere del personal que se menciona en la Tabla 12.

Tabla 12 Personal con conocimiento en diferentes áreas del IoT requeridos para el desarrollo de la plataforma de monitoreo de variables.

Personal
• Ingeniero en Electrónica y de Telecomunicaciones
• Programador (Tarjeta)
• Técnico en informática

Haciendo la suma de todos los gastos contemplando los gastos del personal se obtienen un total de $42,528 con una plataforma de software libre. En la siguiente sección se realiza el análisis de costos para una plataforma de uso comercial.

- Plataforma de uso comercial.

Tabla 13 Material de una plataforma de uso comercial. Precio establecido en pesos mexicanos

Material	**Costo unitario**	**Cantidad**	**Unidad**	**Total**
• Modulo: X-NODE LTE GNSS BG95 M2	1860	2	Pza	$3720
• Modulo: X-NODE SENSORES TEMP/HUM/LUM/PROX	393	1	Pza	$393
• Placa: Tarjeta X-BOARD RASPBERRY PI PICO	737	2	Pza	1474
• Raspberry pi pico H	180	2	Pza	$360
• Cable USB	20	2	Pza	$40
• Software UBIDOTS	500	1	Licencia	$500
• SIM Telcel 50 MB	295	2	Pza	$590
				$7077

Se tiene un costo total de $7,077 en material, para la parte de recursos humanos hay que tener en cuenta que no es necesario el personal, ya que en el uso comercial no se lo requiere.

Para realizar la comparación se utilizó la plataforma Ubidots, esta maneja dos planes de contratación; el plan profesional con un costo de $2,032 mensuales y el plan industrial con un costo de $10,242 mensuales.

Haciendo una proyección, si se utiliza el plan de uso profesional durante un lapso de 6 meses, se genera un costo total de $12,192. Si se utiliza el plan industrial por un lapso de 6 meses sería un total de $73,152 (como se puede observar en la Tabla 14).

En relación para el equipo de cómputo donde se realiza programación e integración, se requiere una computadora con las siguientes características: un equipo básico con un procesador Intel i3 (8ª gen) o AMD Ryzen 3, lo recomendado Intel i5/i7 (10ª gen o superior) o Ryzen 5/7 con el software Linux es lo más recomendable, pero de igual manera se puede usar Windows, memoria RAM minima de 8 GB, lo recomendado seria de 16 GB (ideal si usas varios servicios a la vez), entre otras.

Tabla 14 Comparación de costos entre un plan profesional y un plan industrial.

Uso profesional	**Uso Industrial**
Material= $7077	Material= $7077
Plan por 6 meses= $10242	Plan por 6 meses= 73142
Total= 17,319	Total= 80,219

Realizando la comparativa se tiene que en el uso de software libre tiene un costo de $42,528.

Para poder realizar una comparación similar entre estos dos softwares, se tiene que realizar con el paquete industrial, ya que, a diferencia del profesional, el paquete industrial es el que más se asemeja al de una plataforma de software libre, con un costo de $80,219 teniendo en cuenta que este costo solo abarca por 6 meses y si se extendiera el tiempo ira aumentando mes por mes.

Teniendo esta comparativa entre un software libre y una plataforma comercial, se llegó a la conclusión que es mejor usar una plataforma de software libre.

Grafana es más adecuado para quien requiere un control detallado sobre la visualización de datos y tienen experiencia en configuración técnica, especialmente en entornos con grandes volúmenes y múltiples fuentes de datos. Mientras que Ubidots es excelente para usuarios que buscan una solución integrada para IoT sin necesidad de mucha configuración técnica. Por lo anterior, se seleccionó el software de Grafana para la realización de este trabajo de tesis.

En la Figura 65 y 66 se muestran los Widgets tanto de Grafana como de Ubidots en el monitoreo de las variables de Temperatura, Humedad relativa e Intensidad lumínica.

Figura 65 Widgets de Grafana

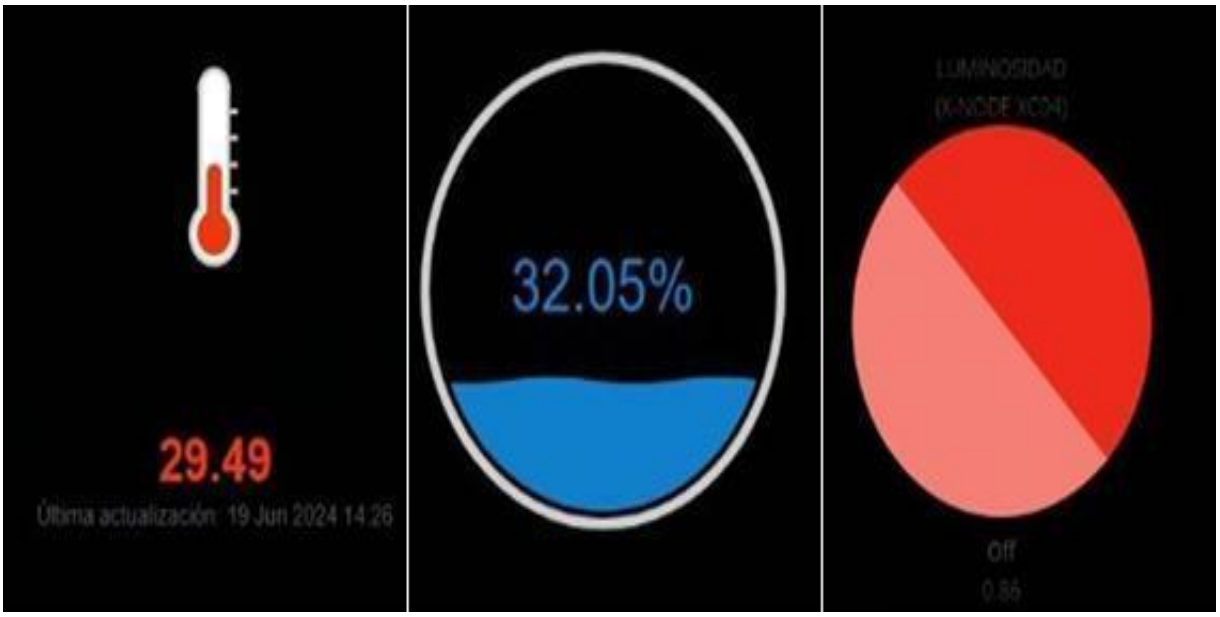

Figura 66 Widgets de Ubidots.

Por último, en las Figuras 67 y 68 se comparan los paneles de monitoreo de cada enfoque.

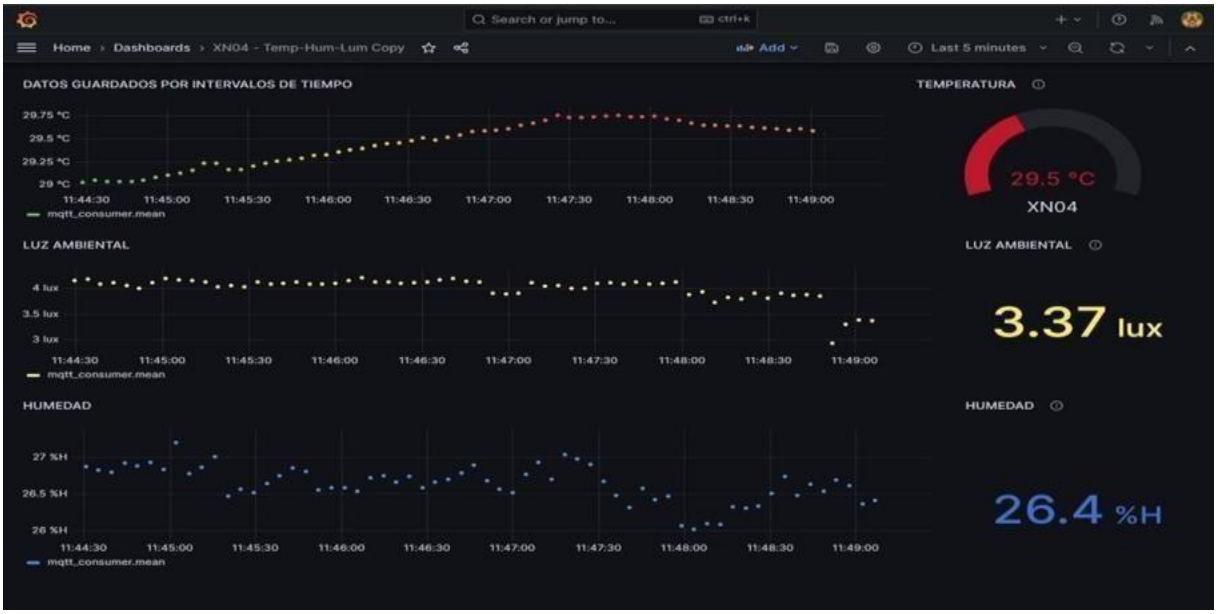

Figura 67 Dashboard de Grafana.

Figura 68 Dashboard de Ubidots.

Resultados.

- Mediante el modulo X-NODE LTE se estableció un nodo de la plataforma XIDE que consistió en la obtención de datos a través de los sensores de un área en especifica.
- Se realizó un análisis de energía para conocer el consumo total de la implementación, en promedio el sistema consume 162 mA aproximadamente por 12.34 hrs.
- Con la conexión y configuración de los sensores de Temperatura, Humedad relativa e intensidad lumínica, se obtuvo información del medio donde fueron colocado. El monitoreo se realizó a intervalos de 30 segundos.

- Se realizó una base de datos con un tamaño de terminado por la capacidad donde se encuentra instalado, es decir el tamaño del disco duro (o volumen) asignado a la instancia de InfluxDB, por ejemplo, si se utiliza un servidor con, 100GB de almacenamiento y el sistema operativo ocupa 10 GB, entonces InfluxDB puede usar hasta 90 GB, para almacenar datos en la nube y visualizarlos mediante el software de Grafana.
- Para la recolección de datos, se utilizó la plataforma Ubiots, en esta se configuró un tablero donde se agregarón los Widgets de Temperatura, Humedad relativa e intensidad lumínica. Tambien se utilizó el software Eclipse Mosquito como sistema intermediario recibiendo y distribuyendo mensajes entre diferentes dispositivos mediante su protocolo de publicador y suscriptor.
- Las herramientas empleadas fueron MQTT, Telegraf, InfluxDB y Grafana, cada una cumpliendo un rol específico en la arquitectura general del sistema.

    I. MQTT actuó como el protocolo de comunicación entre los sensores físicos y el servidor central. Su estructura basada en cliente-servidor (broker) permitió transmitir los datos de forma ligera y eficiente, publica y suscribe información a través de topics, comprendiendo de forma eficaz el modelo de comunicación pub/sub.

    II. Telegraf funcionó como el agente de recopilación de datos, encargado de recibir los mensajes MQTT, transformarlos y enviarlos a la base de datos. Se configuraron plugins específicos para MQTT y para la salida hacia InfluxDB, permitiendo así la comprensión de los datos en tiempo real.

III. InfluxDB fue la base de datos utilizada para almacenar las series temporales generadas por los sensores. Su capacidad para ejecutar consultas optimizadas sobre grandes volúmenes de datos permitió observar cómo se comportaban las variables monitoreadas (temperatura, humedad, etc.) en distintas condiciones.

IV. Finalmente, Grafana permitió la visualización clara y dinámica de los datos almacenados. Se crearon paneles interactivos que mostraban los datos en tiempo real, los cuales interpretaran gráficamente el comportamiento de los sensores.

- Para realizar este proyecto, se implementó una tarjeta ESP32 ya que contiene un módulo de conectividad inalámbrica y WiFi, así como compatibilidad con los protocolos I2C, UART Y SPI. Y para la captura, procesamiento y envió de datos se utilizó la tarjeta RaspBerry Pico H, ambas empleando el IDE de Arduino.

Conclusiones y trabajo a futuro.

Con la integración de tecnologías en la IoT se mejora la eficiencia en gran manera para la recolección y transmisión de datos de tiempo real. Esto ofrece una gran variedad de aplicaciones en áreas como la automatización del hogar, agricultura inteligente, monitoreo ambiental, la industria de manufactura, entre otros.

Con el análisis comparativo entre la plataforma comercial de Ubidots y Grafana, para la recolección de datos en IoT, analizando aspectos significativos como las ventajas y desventajas de la tabla de comparación se tomó la decisiónque el software más adecuado para este trabajo es Grafana.

Si bien Ubidots presenta ventajas al tener una interfaz intuitiva y un soporte técnico accesible, los costos asociados y la dependencia de un proveedor pueden ser desventajas para algunos proyectos.

Por otro lado, la decisión de elegir Grafana para este trabajo se basa en la flexibilidad y la personalización que ofrece ya que da la opción de diseñar tableros personalizados, y permite a los desarrolladores adaptar el sistema a necesidades específicas, así como dar el control total sobre el entorno del trabajo, lo cual lo hace ser más compatible con otros softwares.

El uso de una base de datos de software libre demostró ser una solución eficiente para este proyecto ya que se requirió de una mayor personalización, además de que es un software económico y flexible en comparación con las bases de datos comerciales.

Además, tiene la capacidad de manejar grandes cantidades de datos, su uso sencillo y el costo asociado a largo plazo es menor.

Para finalizar, se logró dar de alta un sistema de recolección de datos IoT a través de un software libre, ya que no solo fue adecuada para los objetivos de este proyecto, sino que también fomenta un aprendizaje sobre la implementación y gestión de sistemas IoT.

Trabajo a futuro.

Algunas de las áreas que se puedan desarrollar como una extensión a este trabajo son:

- Diseño y fabricación de tarjetas de desarrollo enfocadas al IoT.
- Desarrollo de una guía donde se describan los pasos a seguir para el diseño de proyectos

  IoT.
- La elaboración de un protocolo para proyectos IoT.
- La implementación de una comunicación bidireccional.
- Selección de modulos fotovoltaicos y sistemas de almacenamiento de energía adecuados para proveer de energía a sistemas IoT.

BIBLIOGRAFÍA.

- [1] O. Q. Muñoz, Internet de Las Cosas (IoT). Ibukku, LLC, 2019.
- [2] M.C. Vega Las tecnologías IoT dentro de la Industria Conectada4.0. Madrid: pwc, 2015.
- [3] N. Olifer y V. Olifer, Computer Networks: Principles, Technologies and Protocols for Network Design. Wiley Sons, Inc., John, 2014.
- [4] V. Yacchirema y C. Diana, Arquitectura de Interoperabilidad de dispositivos físicos para el Internet de las Cosas (IoT), 2019a ed. España: D.C., 2019.
- [5] “¿Qué es una base de datos?” 24 noviembre 2020 Oracle | Cloud Applications and Cloud Platform. Accedido el 16 de junio de 2024. [En línea]. Disponible: https://www.oracle.com/mx/database/what-is-database/
- [6] A. y. C. Higa y Zenteno, “Auditoría de Base de Datos Temporal”, Licenciatura, Univ. Palermo, Argentina, 2012.
- [7] “IoT Dashboards | IoT Solution Templates | Kaa IoT Cloud”. Kaa IoT platform. Accedido el 10 de julio de 2024. [En línea]. Disponible: https://www.kaaiot.com/iotdashboards
- [8] C. Bell Introducción a las redes de sensores. En: Introducción a las redes de sensores con XBee, Raspberry Pi y Arduino. Apress, Berkeley, CA. (2020)
- [9] Electricity Magnetism. “¿Cómo se calcula la energía consumida en un circuito eléctrico?” Electricity - Magnetism. Accedido el 18 de diciembre de 2024. [En línea].

- Disponible: https://www.electricity-magnetism.org/es/como-se-calcula-la-energiaconsumida-en-un-circuito-electrico/

[10] R. L. Varela, Diccionario de la lengua española. Madrid: Everest, 2012.

- [11] E. Expertos en Ciencia y Tecnología (2021, noviembre 2) Procesamiento de la información en IoT. VIU España. Accedido el 1 de mayo de 2023 Available: https://www.universidadviu.com/es/actualidad/nuestrosexpertos/procesamiento-de-la-informacion-en-iot A
- [12] L. J. Aguilar, Internet de las cosas: Un futuro hiperconectado: 5G, inteligencia artificial, Big Data, Cloud, Blockchain y ciberseguridad. Marcombo, 2021.
- [13] Equipo de Enciclopedia. “Microprocesador: qué es, para qué sirve, características y partes”. Enciclopedia Significados. Accedido el 18 de diciembre de 2024. [En línea]. Disponible: https://www.significados.com/microprocesador/
- [14] C y R. Ronald y Solís M, Microprocesadores Fundamentos y Aplicaciones DISEÑO EMBEBIDO CON SIMULACIONES INTERACTIVAS. Ecuador: LaTIn, 2019.
- [15] A. Leandro. “Definición de Celular (telecomunicaciones)”. Alegsa.com.ar. Accedido el 1 de agosto de 2023. [En línea]. Disponible: https://www.alegsa.com.ar/Dic/celular.php
- [16] T.-Mobile Support. “Tarjeta SIM e eSIM”. Accedido el 30 de septiembre de 2024. [En línea]. Disponible: https://es.t-mobile.com/support/devices/sim-esim
- [17] Cisco. “¿Qué es una red inalámbrica? - Cableada frente vs. inalámbrica”. Cisco. Accedido el 20 de octubre de 2023. [En línea]. Disponible:

- https://www.cisco.com/c/es_mx/solutions/smallbusiness/resourcecenter/networking/wireless-network.html

[18] Solectroshop. “¿Qué es MQTT? Definición y detalles”. Paessler - The Monitoring Experts. Accedido el 13 de diciembre de 2024. [En línea]. Disponible: https://www.paessler.com/es/itexplained/mqtt#:~:text=MQTT%20son%20las%20siglas%20de,cuanto%20al%20ancho%20de%20banda.

- [19] L. Llamas. “¿Qué es MQTT? Su importancia como protocolo IoT”. Luis Llamas Ingeniería, Informática y diseño. Accedido el 15 de abril de 2024. [En línea]. Disponible: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/
- [20] Amazon Web Services. “¿Qué es un certificado SSL? - Explicación del certificado SSL/TLS - AWS”. Amazon Web Services, Inc. Accedido el 16 de diciembre de 2024. [En línea]. Disponible: https://aws.amazon.com/es/what-is/ssl-certificate/
- [21] Solectro. “¿Qué es MQTT? El protocolo de comunicación para IoT”. Solectroshop.com. Accedido el 1 de mayo de 2023. [En línea]. Disponible: https://solectroshop.com/es/blog/que-es-mqtt-el-protocolo-de-comunicacion-para-iotn117?srsltid=AfmBOoqy7b2Qd4iErpYRWVJwuzica4HNn2NyOHa1OP4HWH8Wv5tyM553
- [22] P. Galvan. “Hola Mundo IoT”. SG Buzz. Accedido el 16 de diciembre de 2023. [En línea]. Disponible: https://sg.com.mx/revista/51/hola-mundo-iot

- [23] IBM. (s.f.). ¿Qué es el software de código abierto (OSS)? Accedido el 13 de diciembre de 2024. IBM - United States. https://www.ibm.com/mx-es/topics/open-source
- [24] “Plataforma”. Ubidots : IoT industrial potente pero sencillo. Accedido el 18 de junio de 2024. [En línea]. Disponible: https://es.ubidots.com/platform

[25] Eclipse Foundation. (s.f.). Eclipse Mosquitto. Eclipse Mosquitto.

https://mosquitto.org/

- [26] “Que© son y cÃ³mo usar los Topics en MQTT correctamente”. Luis Llamas.
- Accedido el 19 de junio de 2024. [En línea]. Disponible:
- https://www.luisllamas.es/que-son-y-como-usar-los-topics-en-mqtt-correctamente/
- [27] InfluxDB | Real-time insights at any scale | InfluxData. (s.f.). InfluxData. https://www.influxdata.com/
- [28] Telegraf | InfluxData. (s.f.). InfluxData. https://www.influxdata.com/time-seriesplatform/telegraf/
- [29] Raspberry pi. “Raspberry pi”. raspberrypi.com. Accedido el 11 de julio de 2024. [En línea]. Disponible: https://www.raspberrypi.com/products/raspberry-pi-pico/
- [30] ESP32-WROOM-32. “Datasheet”. IIS Windows Server. Accedido el 15 de agosto de 2024. [En línea]. Disponible: https://agelectronica.lat/pdfs/textos/E/ESP-WROOM-32.PDF
- [31] S. Aguirre. “MicroPYTHON CONOCE CARACTERISTICAS Y COMO APROVECHARLO - RedUSERS”. RedUSERS. Accedido el 16 de julio de 2024. [En

- línea]. Disponible: https://www.redusers.com/noticias/publicaciones/micropythonconoce-caracteristicas-y-como-aprovecharlo/
- [32] “Qué es lenguaje C: el origen, las ventajas, las características y la sintaxis del lenguaje de programación”, Ebac, 1 de junio de 2023. Accedido el 16 de agosto de 2024. [En línea]. Disponible: https://ebac.mx/blog/que-es-lenguaje-c

[33] admin. “Industrias GSL | Sensor de luz”. Industrias GSL. Accedido el 17 de julio 2024. [En línea]. Disponible: https://industriasgsl.com/blogs/automatizacion/sensor_de_luz?srsltid=AfmBOop73OJ_Ar4OVync1Q_YzEu5F3RwX7YZ8CKUqxBdPtB6sy2GpmaG

- [34] “Sensor de humedad | ¿Qué es y cómo funciona? | SDI”. SDI Industrial. Accedido el 9 de agosto de 2024. [En línea]. Disponible: https://sdindustrial.com.mx/blog/sensorde-humedad/
- [35] Microside. “X-NODE Temp-Hum / Prox-Lum”. Tienda Microside. Accedido el 22 de septiembre de 2024. [En línea]. Disponible: https://store.microside.com/productos/xnode-temp-hum-prox-lum/
- [36] Microside. (s.f.). X-NODE LTE GNSS SIM7080G. Tienda Microside.

https://store.microside.com/productos/x-node-lte-gnss-sim7080g/

- [37] Microside. (s.f.-b). X-NODE MCU WIFI BLE (ESP32-WROOM). Tienda Microside. https://store.microside.com/productos/x-node-mcu-wifi-ble-esp32-wroom/

-
- [38] R. Awati, B. Lutkevich y K. Gerwig. “What is a Network Node? | Definition from TechTarget”. Search Networking. Accedido el 19 de mayo de 2025. [En línea]. Disponible: https://www.techtarget.com/searchnetworking/definition/node

APENDICE A. Programación en C.

A.1 Código para el Módulo esp32

La configuración y programación del módulo se ha realizado mediante el software Arduino

A.2 Explicación del código

//Se incluyen las bibliotecas para el manejo de WIFI, MQTT y UART (comunicación serial)

- #include <WiFi.h>
- #include <PubSubClient.h>
- #include <HardwareSerial.h>

//Se definen las credenciales de la red WIFI y del servidor MQTT al que se conectara a el dispositivo

- const char* ssid = "---";
- const char* password = "----";
- const char* mqttServer = "---- ";
- const int mqttPort = 1883;
- const char* mqttUser = "Usuario";
- const char* mqttPassword = "-----";

//Se crean objetos para el cliente WIFI y el cliente MQTT

- WiFiClient espClient;
- PubSubClient client(espClient);

//Se define un objeto de tipo Hardware Serial para comunicacion UART y se especifican los pines RX Y TX

```
HardwareSerial MySerial (1); // Definir el serial para UART1
```

- const int MySerialRX = 16;
- const int MySerialTX = 17;

```
//Se inicializa la comunicación serial y UART  void
setup ()
{
```

- Serial.begin(115200);
- MySerial.begin(115200, SERIAL_8N1, MySerialRX, MySerialTX);

```
//Se conecta el dispositivo a la red WIFI especificada
```

- WiFi.begin(ssid, password);
- Serial.println("Connecting to WiFi...");
- while (WiFi.status() != WL_CONNECTED)

```
 {
delay(500);
Serial.print(".");
 }
Serial.println("Connected to WiFi network");
```

```
//Se conecta el cliente MQTT al servidor
client.setServer(mqttServer, mqttPort);  while
(!client.connected())
{
Serial.println("Connecting to MQTT...");  if
(client.connect("ESP32Client", mqttUser, mqttPassword))
{
 Serial.println("Connected to MQTT");
}
else
{
 Serial.print("Failed with state ");
Serial.println(client.state());  delay(2000);
}
 }
 }

 //Si el cliente MQTT esta conectado, se mantiene activa su conexion y se comprueba si hay
mensajes entrantes  void loop()
{
```

```
 if (client.connected())

 {

 client.loop();

 //Se envía un comando al dispositivo externo (moduloXIDE) a través de UART para verificar si la comunicación estableció correctamente, y se imprime la respuesta
MySerial.println("XN04A?");

String str = MySerial.readString();

Serial.println(str);

//Se envía un comando para obtener la temperatura, luminosidad y humedad del sensor, se procesa la respuesta y se publica en el MQTT

MySerial.println("XN04A+GT"); // Temperatura del sensor

String str1 = MySerial.readString();  str1 =

str1.substring(6); // Corta los primeros 6 caracteres

client.publish("Sensor/Temperatura", str1.c_str());

Serial.println(str1);

 MySerial.println("XN04A+GL"); // Luminosidad del sensor  String str2 = MySerial.readString(); str2 = str2.substring(6);  client.publish("Sensor/Luminosidad", str2.c_str());

Serial.println(str2);

MySerial.println("XN04A+GH"); // Humedad del sensor

String str3 = MySerial.readString();  str3 =
```

```
str3.substring(6); client.publish("Sensor/Humedad",

str3.c_str());

Serial.println(str3);

 //Si el cliente MQTT no está conectado, se imprime un mensaje de conexion no aceptada

 }

else

{

 Serial.println("MQTT not connected");

} delay(500);
```

APENDICE B. Programación en Python

B.1 Código de la Rapsberry pi pico H

La configuración y programación se ha realizado mediante el software Thonny utilización el intérprete de MicroPython

B.2 Explicación del Código

Importación de módulos y configuración inicial

```
from machine import UART, Pin from

xide import xn03, xn04, xc04 import

time

TOKEN = "*****************************************"

DEVICE_NAME = "x-node-xc04"
```

- UART y Pin son clases proporcionadas por el módulo machine en MicroPython para manejar la comunicación serial y los pines GPIO respectivamente.
- xn03, xn04, xc04 son módulos o clases específicas importadas desde xide, que presumiblemente contienen funciones para interactuar con sensores y el módulo XC04.
- time es un módulo estándar en Python para operaciones relacionadas con el tiempo.

Función principal main ()

```
def main():
```

```
MIKROBUS_SERIAL = UART(0, baudrate=115200, tx=Pin(0), rx=Pin(1))

xc04.init(MIKROBUS_SERIAL, Pin_PK=10)

xc04.CSQ(MIKROBUS_SERIAL) xc04.COPS(MIKROBUS_SERIAL)

_, success = xc04.QIACT(MIKROBUS_SERIAL, "1") while

not success:

time.sleep(1)

_, success = xc04.QIACT(MIKROBUS_SERIAL, "1")

_, success = xc04.QMTCFG(MIKROBUS_SERIAL, ["keepalive", 0, 60]) while

not success:

time.sleep(1)

_, success = xc04.QMTCFG(MIKROBUS_SERIAL, ["keepalive", 0, 60]) success

= False

_, success = xc04.QMTOPEN(MIKROBUS_SERIAL, args=[0, "industrial.api.ubidots.com",

1883]) while not

success:

time.sleep(1)

_, success = xc04.QMTOPEN(MIKROBUS_SERIAL, args=[0, "industrial.api.ubidots.com",

1883])

_, success = xc04.QMTCONN(MIKROBUS_SERIAL, [0, "UBI-2", TOKEN]) while

not success:
```

time.sleep(1) var, success = xc04.QMTCONN(MIKROBUS_SERIAL, [0,

"UBI-2", TOKEN])

Configuración inicial del módulo XC04

Se inicializa la UART (MIKROBUS_SERIAL) con una velocidad de baudios de 115200 y se especifican los pines tx y rx.

xc04.init(MIKROBUS_SERIAL, Pin_PK=10) inicializa el módulo XC04 en la UART especificada, con el pin PK (Positio Key) configurado como 10.

xc04.CSQ(MIKROBUS_SERIAL) y xc04.COPS(MIKROBUS_SERIAL) realizan consultas específicas al módem XC04 para obtener información de señal y red respectivamente.

Conexión MQTT con Ubidots

xc04.QIACT(MIKROBUS_SERIAL, "1") intenta activar la conexión de datos del módem. Esto es ser parte de la configuración de red móvil.

xc04.QMTCFG(MIKROBUS_SERIAL, ["keepalive", 0, 60]) configura un parámetro de keepalive para mantener la conexión MQTT activa.

xc04.QMTOPEN(MIKROBUS_SERIAL, args=[0, "industrial.api.ubidots.com", 1883]) abre una conexión MQTT con el servidor de Ubidots en el puerto 1883.

xc04.QMTCONN(MIKROBUS_SERIAL, [0, "UBI-2", TOKEN]) intenta conectar el dispositivo al servidor MQTT de Ubidots usando un identificador y un token de autenticación.

Bucle principal de envío de datos

```
try:

while True:

temperatura = xn04.getTemp(MIKROBUS_SERIAL) humedad =

xn04.getHum(MIKROBUS_SERIAL) luminosidad = xn04.getLum(MIKROBUS_SERIAL)

proximidad = xn04.getProx(MIKROBUS_SERIAL) posicion =

xn03.getPos(MIKROBUS_SERIAL) time.sleep(0.1) var =

xc04.QMTPUB(MIKROBUS_SERIAL, [0, 0, 0, 0, "/v1.6/devices/" + DEVICE_NAME], {

"temperatura": temperatura,

"humedad": humedad,

"luminosidad": luminosidad,

"proximidad": proximidad,

"posicion": posicion

})

time.sleep(10)        except

KeyboardInterrupt:

xc04.QMTCLOSE(MIKROBUS_SERIAL)
```

Obtención de datos de sensores

xn04.getTemp(MIKROBUS_SERIAL), xn04.getHum(MIKROBUS_SERIAL), xn04.getLum(MIKROBUS_SERIAL), xn04.getProx(MIKROBUS_SERIAL), xn03.getPos(MIKROBUS_SERIAL) son funciones que obtienen datos de temperatura, humedad, luminosidad, proximidad y posición respectivamente, desde los sensores conectados a la UART.

Envío de datos a Ubidots

xc04.QMTPUB(MIKROBUS_SERIAL, [0, 0, 0, 0, "/v1.6/devices/" + DEVICE_NAME], { ... }) publica los datos de los sensores en Ubidots a través de MQTT.

Los datos se envían en un formato específico que incluye el nombre del dispositivo (DEVICE_NAME) y los valores de temperatura, humedad, luminosidad, proximidad y posición.

Manejo de excepciones

except KeyboardInterrupt: maneja la interrupción del programa por el usuario (presumiblemente con Ctrl+C), cerrando correctamente la conexión MQTT con xc04.QMTCLOSE(MIKROBUS_SERIAL).

Lo que en fin hace este código es que establece una conexión MQTT con Ubidots a través del módem XC04 y envía datos de múltiples sensores (temperatura, humedad, luminosidad, proximidad y posición) a intervalos regulares. Es útil para aplicaciones de monitoreo remoto y control basadas en IoT (Internet of Things).

Tomar las siguientes consideraciones:

Asegurarse de que la configuración de pines (tx, rx, Pin_PK) y la configuración de red (QIACT, QMTCFG, QMTOPEN, QMTCONN) sean adecuadas para el entorno y dispositivo específico.

El manejo de errores y la gestión de desconexiones pueden necesitar ser refinados para garantizar la estabilidad y fiabilidad de la aplicación en entornos de producción.

Finalmente se observa el Dashboard y ver las gráficas en tiempo real del monitoreo de los sensores