Compás
capacitación e investigación

2025

Fundamentos y Aplicaciones del Internet de las Cosas:
Arquitectura, Comunicación e Implementación Práctica

Leonardo Hernández Rojas

Fundamentos y Aplicaciones del Internet de las Cosas:

Arquitectura, Comunicación e Implementación Práctica

Leonardo Hernández Rojas

ISBN: 978-9942-53-020-2 DOI: http://doi.org/10.48190/9789942530202

Compás
capacitación e investigación

ISBN: 978-9942-53-020-2
DOI: http://doi.org/10.48190/9789942530202
Distribución online
Acceso abierto

Primera edición 2025-11-02

Cita
Hernández, L. (2025) Fundamentos y Aplicaciones del Internet de las Cosas: Arquitectura, Comunicación e Implementación Práctica. Editorial Grupo Compás

Este libro es parte de la colección de la Univesidad Técnica de Machala y ha sido debidamente examinado y valorado en la modalidad doble par ciego con fin de garantizar la calidad de la publicación.

Prefacio:

Introducción

Vivimos en una era donde los objetos cotidianos han adquirido la capacidad de conectarse, recopilar y transmitir datos, transformando radicalmente la forma en que interactuamos con el entorno. Esta revolución tecnológica es liderada por el Internet de las Cosas (IoT), una disciplina transversal que abarca desde sensores inteligentes hasta plataformas en la nube, pasando por protocolos de comunicación, procesamiento distribuido y automatización. Este paradigma ha dado origen a nuevas formas de interacción digital en sectores tan diversos como la salud, la agricultura, la industria, la educación, el transporte o la gestión urbana.

La importancia del IoT no radica solamente en su capacidad para interconectar objetos, sino en su potencial para generar **impacto social, económico y ambiental** mediante la optimización de procesos, el análisis de datos en tiempo real y la automatización inteligente de tareas. Su relevancia se acentúa en un mundo cada vez más interdependiente, donde los datos se convierten en activos estratégicos para la toma de decisiones y la mejora de la calidad de vida. Asimilar el IoT, por tanto, no es solo una necesidad tecnológica, sino también una competencia clave para enfrentar los desafíos del presente y construir soluciones sostenibles para el futuro.

A pesar del vertiginoso crecimiento del IoT y su incorporación progresiva en la vida cotidiana y profesional, aún existen importantes retos para su adopción masiva y efectiva: dificultad para integrar múltiples tecnologías y estándares, y la escasa sistematización en la enseñanza de sus fundamentos. En contextos académicos y profesionales, muchas veces se abordan las soluciones IoT de manera fragmentada, sin una comprensión estructurada de sus componentes, protocolos, plataformas y aplicaciones reales.

En este escenario, se hace urgente y necesaria la existencia de materiales didácticos que aborden el IoT desde una perspectiva integral, que combinen teoría y práctica, y que faciliten el aprendizaje progresivo desde los fundamentos hasta la implementación avanzada. El presente libro nace con ese propósito: servir como una guía completa, estructurada y accesible para estudiantes universitarios, docentes, técnicos e ingenieros interesados en adquirir competencias sólidas en el diseño, construcción e implementación de soluciones basadas en Internet de las Cosas. A lo largo de sus capítulos, se proporcionan conceptos esenciales, modelos arquitectónicos, tecnologías de

comunicación como REST-API y MQTT, herramientas de desarrollo como Node-RED y ESPHome, y plataformas como Home Assistant y AWS IoT. Todo el contenido se presenta con un enfoque pedagógico, ilustrado con ejemplos, diagramas explicativos y estudios de caso reales en sectores como salud, domótica, agricultura y ciudades inteligentes.

Objetivo general

Brindar una formación integral en Internet de las Cosas (IoT), que permita al lector entender sus fundamentos conceptuales, arquitecturas, protocolos de comunicación, plataformas de desarrollo y metodologías de implementación práctica, mediante un enfoque técnico y aplicado en contextos reales.

A través de su contenido, el lector podrá:

- Comprender los principios y componentes esenciales de IoT., ver Capítulo 1.
- Explorar la arquitectura y los estándares que rigen los sistemas IoT, ver Capítulo 1.
- Profundizar en protocolos clave como **REST-API y MQTT**, ver Capítulo 2.
- Implementar soluciones IoT utilizando **Node-RED, ESPHome y Home Assistant**, ver Capítulo 3.
- Analizar estudios de caso en sectores como **salud, agricultura, domótica, seguridad y automatización industrial**, ver Capítulo 3.
- Desarrollar aplicaciones IoT mediante programación en **C con VSCode y PlatformIO** para microcontroladores.

Estructura del libro

El libro está organizado en **tres capítulos**, estructurados de forma progresiva y lógica:

📗 *Capítulo 1: Fundamentos del Internet de las Cosas*

Introduce los conceptos esenciales de IoT, su evolución histórica y el impacto en diversos sectores. Se abordan las arquitecturas típicas (en capas, orientadas a servicios, estándares como oneM2M o ITU-T), los componentes físicos y lógicos (sensores, actuadores, microcontroladores) y las tecnologías de red y conectividad. Se incluyen diagramas explicativos, ejemplos de dispositivos y modelos de referencia.

📘 *Capítulo 2: Comunicación en IoT*

Explora los principales protocolos de comunicación como REST-API y MQTT, explicando su funcionamiento, integración en arquitecturas IoT, y métodos HTTP. Se detallan estructuras de mensajes, mecanismos de seguridad y niveles de servicio. Además, se introducen conceptos de Edge Computing, Fog Computing y su relación con el procesamiento de datos y la inteligencia artificial (AIoT), con gráficos comparativos y casos de uso.

📙 *Capítulo 3: Plataformas IoT y su Implementación*

Aborda herramientas y plataformas clave como Home Assistant, AWS IoT, Google Cloud IoT y Azure IoT. Se enseña a integrar flujos de datos mediante Node-RED, programar dispositivos en C usando PlatformIO, y simplificar implementaciones con ESPHome. El capítulo finaliza con estudios de caso en sectores clave que consolidan los conocimientos adquiridos a través de soluciones reales y contextualizadas.

Características pedagógicas

- **Enfoque progresivo**: Los contenidos avanzan desde lo teórico a lo práctico, facilitando la construcción gradual del conocimiento.
- **Diagramas y figuras explicativas**: Cada sección incluye esquemas visuales que ayudan a representar conceptos técnicos de forma clara.
- **Consejos pedagógicos destacados**: A lo largo del texto se incluyen recomendaciones y consejos para docentes y estudiantes, facilitando el aprendizaje guiado.
- **Estudios de caso reales**: Se presentan aplicaciones prácticas del IoT en diversos dominios, lo que permite contextualizar el conocimiento y fomentar la transferencia tecnológica.
- **Lenguaje técnico accesible**: Se emplea un estilo claro y riguroso, explicando cada término técnico en su primera aparición.

Preguntas de revisión y autoevaluación: Al final de cada capítulo, se incluyen preguntas de opción múltiple, verdadero/falso y respuesta corta para reforzar el aprendizaje.

Tabla de contenido

Tabla de figuras

1. Capítulo 1: Fundamentos del Internet de las Cosas

Elementos de Apertura

Objetivos de aprendizaje:

- Definir el concepto y evolución histórica del Internet de las Cosas (IoT).
- Analizar su impacto en la sociedad y diferentes industrias.
- Reconocer las áreas de aplicación más relevantes de IoT.

Competencias:

- Interpretar los fundamentos teóricos del IoT.
- Identificar componentes y arquitecturas esenciales del ecosistema IoT.

Preguntas de enfoque:

- ¿Qué factores tecnológicos impulsaron el surgimiento del IoT?
- ¿Cómo afecta el IoT a sectores como la salud, la industria y la vida cotidiana?
- ¿Cuáles son los componentes básicos de una arquitectura IoT?

La Figura 1.1 muestra de forma resumida la evolución e impacto actual que tiene la IoT en nuestra vida. A lo largo de este libro haremos una introducción a todos los fundamentos necesarios que ayuden a explicar dicha figura. Comencemos …

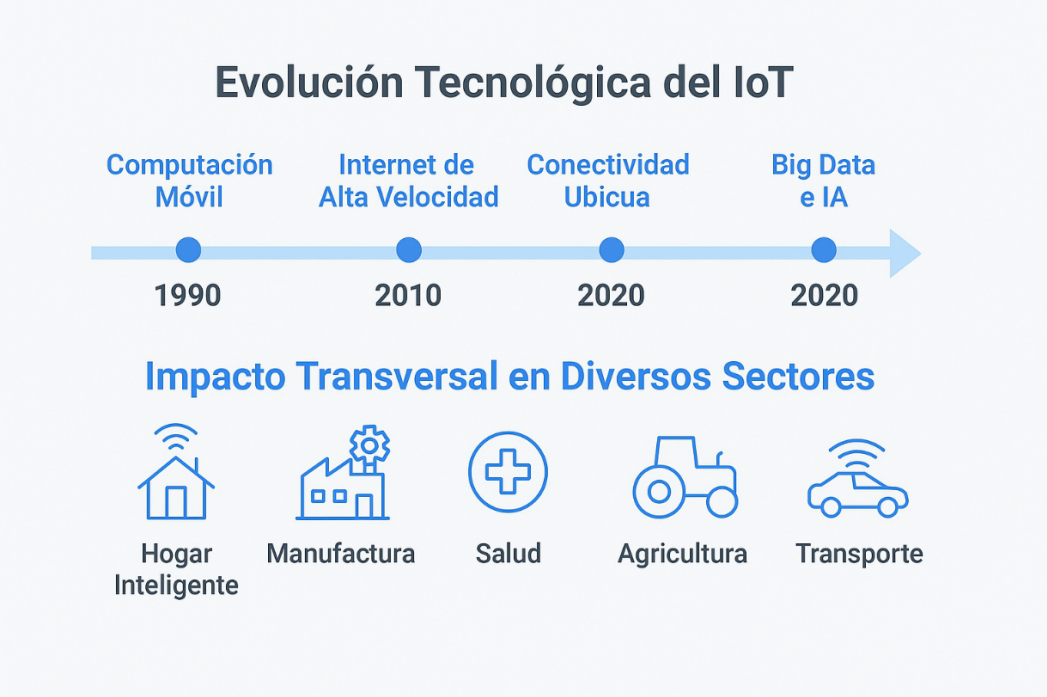


Figura 1.1 Evolución y áreas de impacto del IoT. Elaboración propia.

1.1 Introducción a IoT

1.1.1 Concepto y Evolución del IoT

El Internet de las Cosas (IoT) se define como la red de objetos físicos interconectados, que incorporan sensores, software y otras tecnologías para recopilar e intercambiar datos a través de internet (Atzori, Iera, & Morabito, 2021). Por ejemplo, un sistema de riego automatizado que ajusta el suministro de agua según datos meteorológicos y la humedad del suelo, optimizando el uso de recursos en la agricultura.

Diversos autores han definido el concepto de IoT desde diferentes perspectivas:

- ✓ "Computadoras conectadas a Internet con sensores y actuadores" –@ tamberg.
- ✓ "Objetos físicos con una API web" –@ hansamann.
- ✓ "Red global de computadoras, sensores y actuadores conectados a través de protocolos de Internet".
- ✓ Web of Things (WoT): Se basa en servicios web RESTful que permiten la medición y manipulación de propiedades físicas.
- ✓ Recomendación ITU-T Y.2060: Define al IoT como "una infraestructura global para la sociedad de la información que habilita servicios

avanzados mediante la interconexión de cosas físicas y virtuales a través de tecnologías de comunicación interoperables existentes y en evolución".

Al mismo tiempo, han aparecido algunos términos que han ido evolucionando con el tiempo como son:

- ✓ Internet of Things (IoT): Concepto introducido por Kevin Ashton (MIT).
- ✓ Web of Things (WoT): IoT basado en servicios web.
- ✓ Internet of Everything (IoE): Término propuesto por Cisco para una interconexión global.
- ✓ Industrial IoT (IIoT): Aplicación del IoT en entornos industriales.
- ✓ Smart IoT: Combinación de inteligencia en objetos y redes.
- ✓ AIoT: Artificial Intelligence of Things

El estudio de del IoT, debe comenzar por definir y caracterizar "las cosas", para luego poder definir las interconexiones de todos los elementos que puedan estar involucrados de forma organizada o mejor aún que pueda ser considerada como un estándar. Estas formas de organización son conocidas como arquitecturas que veremos más adelante.

Comencemos por las "cosas" en IoT, donde **ITU-T Y.2060** plantea que estas pueden ser:

- **Físicas:** Objetos tangibles como robots industriales, electrodomésticos y dispositivos electrónicos.
- **Virtuales:** Información almacenada y procesada en sistemas digitales, como contenido multimedia o software.

En cambio, la definición de un "**dispositivo IoT**" la podríamos definir como un equipo con capacidad obligatoria de comunicación y capacidades opcionales de medición, actuación, captura de datos, almacenamiento y procesamiento. Conceptos que abordaremos más adelante.

La evolución del Iot, implica contar un poco su historia y aunque el concepto de IoT ha ganado relevancia en los últimos años, su origen se remonta a principios del siglo XX con el desarrollo de sistemas de telemetría, por ejemplo:

- ✓ 1912: Se implementó el primer sistema de telemetría en Chicago utilizando líneas telefónicas para monitorear datos de plantas de energía.

- ✓ 1930s: Se introdujo la radiosonda, dispositivo para monitoreo meteorológico mediante globos aerostáticos.
- ✓ 1980s: Se popularizó la tecnología M2M (Machine-to-Machine) con sistemas SCADA (Supervisory Control and Data Adquisition) en fábricas.
- ✓ 1990s: ADEMCO desarrolló su propia red de radio privada debido al alto costo de la conectividad celular.
- ✓ 1995: Siemens lanzó el primer módulo celular diseñado para aplicaciones M2M.

Esta evolución de IoT ha sido impulsada por diversos factores como los siguientes y de forma resumida en la Figura 1.2

- Avances en conectividad inalámbrica (Wi-Fi, Bluetooth, 5G).
- Miniaturización: Sensores y procesadores cada vez más pequeños y eficientes.
- Reducción de costos en microcontroladores y módulos de comunicación.
- Conectividad ubicua: Ampliación de redes inalámbricas y móviles.
- Adopción generalizada de IP: Protocolo estándar de comunicación.
- Economía de la computación: Reducción de costos en hardware y software.
- Avances en analítica de datos: Mejoras en procesamiento y toma de decisiones basada en datos.
- Crecimiento de la computación en la nube: Infraestructura de almacenamiento y procesamiento a gran escala.

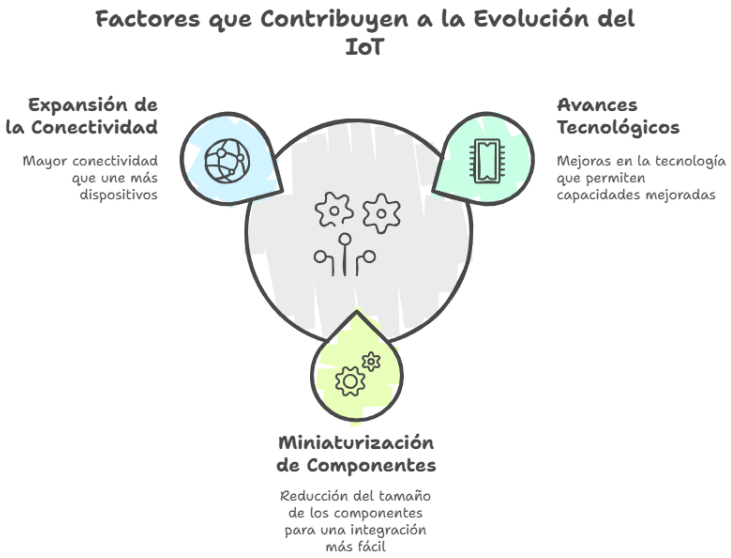


Figura 1.2 Factores que han impulsado el crecimiento del IoT, destacando conectividad, avances tecnológicos y miniaturización de componentes. Elaboración propia.

La evolución del IoT puede entenderse como una pirámide de conocimiento que consta de tres niveles fundamentales: **Fundamentos de IoT**, **Comunicación y Procesamiento**, e **Implementación de Soluciones**. En la base se encuentran los conceptos básicos y la arquitectura de IoT, seguidos de la comprensión de protocolos y análisis de datos, hasta llegar al desarrollo práctico de plataformas y soluciones IoT, ver Figura 1.3.

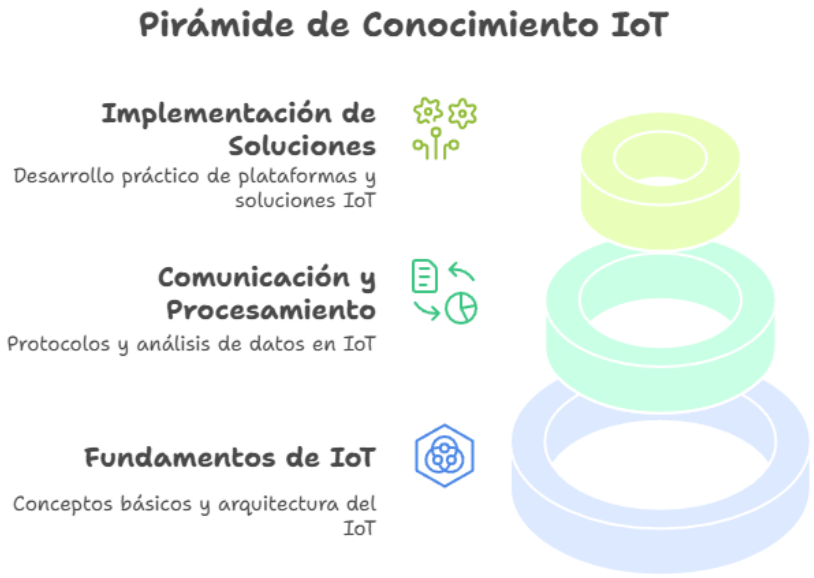


Figura 1.3 Pirámide de conocimiento IoT, que ilustra la progresión desde la teoría hasta la implementación en el ecosistema IoT. Elaboración propia.

Por último, definamos características fundamentales que debemos tener en cuenta cuando hablamos de IoT:

- ✓ Interconectividad: Cualquier objeto puede estar conectado y comunicarse con otros mediante redes cableadas o inalámbricas.
- ✓ Heterogeneidad: Los dispositivos IoT operan en distintas plataformas de hardware y redes de comunicación.
- ✓ Cambios Dinámicos: Los dispositivos IoT pueden cambiar de estado dinámicamente, afectando la cantidad y naturaleza de los datos transmitidos.
- ✓ Gran Escala: Existen actualmente **miles de millones de dispositivos conectados**, lo que exige soluciones escalables.

1.1.2 Impacto del IoT en la Sociedad y la Industria

IoT ha revolucionado diversos sectores, incluyendo la salud, la industria, el transporte y la domótica. Según Sinha, S. (2025, 22 abril), se estima que para 2030 existirán más de 40 billones de dispositivos IoT conectados a nivel mundial, como se puede ver en la Figura 1.4. Su aplicación permite una mejor gestión de recursos, mayor eficiencia operativa y la optimización de procesos de producción. En el sector salud, los dispositivos IoT permiten el monitoreo

remoto de pacientes mediante sensores de signos vitales conectados a plataformas de análisis en la nube.

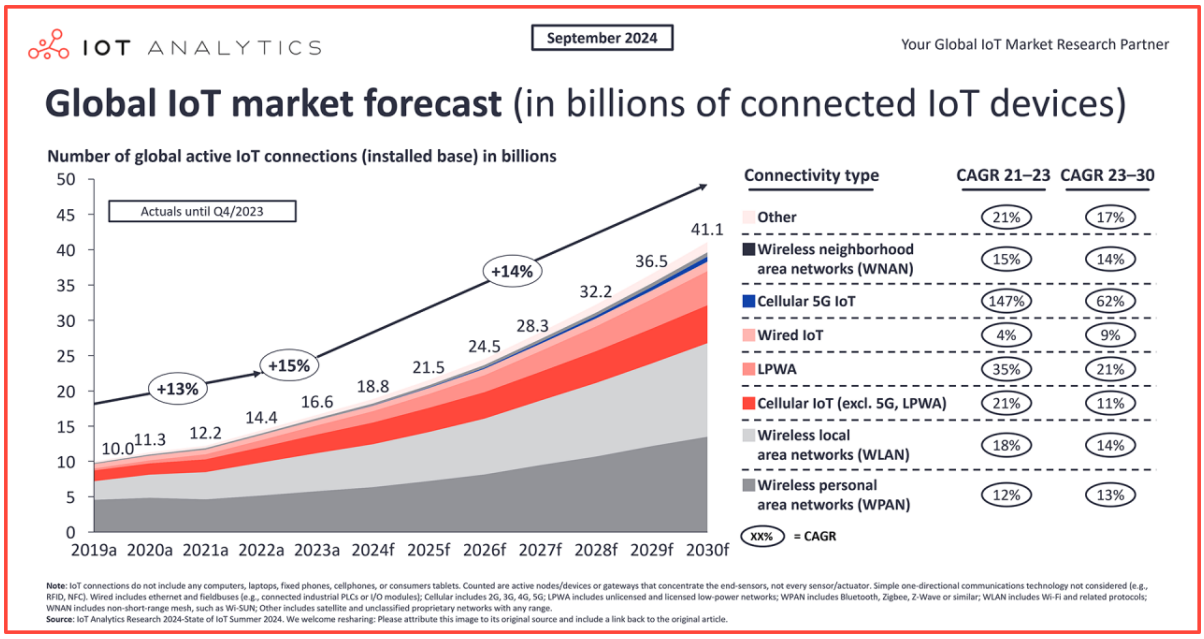


Figura 1.4 Estado del mercado global de IoT según IoT Analytics. Con un pronóstico de crecimiento del 13% al 15% de dispositivos IoT conectados hasta el 2030. Tomado de Sinha, S. (2025, 22 abril)

El impacto del IoT se extiende a múltiples sectores, generando mejoras en **eficiencia operativa**, **optimización de recursos**, y la aparición de **nuevas oportunidades de negocio**. En la Figura 1.5 se representa cómo la tecnología IoT ha generado cambios disruptivos en diferentes industrias, tales como **salud, agricultura y manufactura**. Este impacto no solo radica en la automatización y el monitoreo, sino también en la capacidad de tomar decisiones basadas en datos en tiempo real.

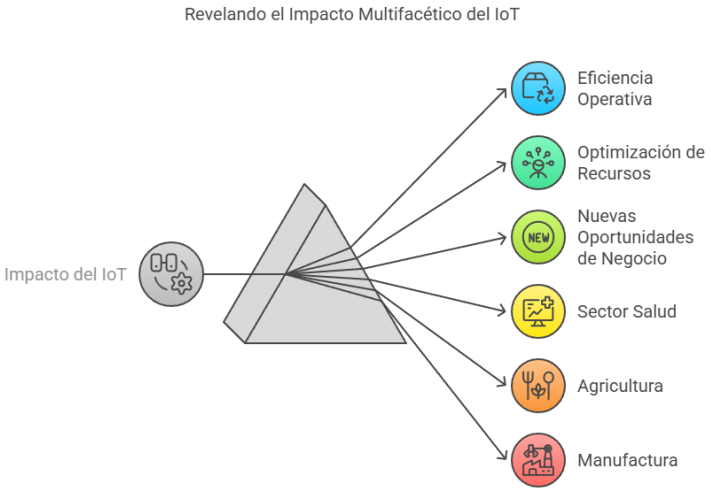


Figura 1.5 Impacto multifacético del IoT en distintos sectores, desde la salud y la manufactura hasta la optimización de recursos y la eficiencia operativa. Elaboración propia.

1.1.3 Principales Áreas de Aplicación del IoT

Las aplicaciones de IoT abarcan diversas áreas, entre muchas otras podemos encontrar:

- **Salud**: Monitoreo remoto de pacientes, dispositivos portátiles para control de enfermedades crónicas.
- **Industria 4.0**: Sensores en líneas de producción para mantenimiento predictivo y control de calidad.
- **Domótica**: Sistemas inteligentes para el control de iluminación, climatización y seguridad en el hogar.

IoT también tiene aplicaciones en **ciudades inteligentes**, **monitoreo ambiental** y **automatización industrial** Al-Fuqaha, A, Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2021). La**¡Error! No se encuentra el origen de la referencia.** Figura 1.6 sintetiza las principales áreas donde el IoT contribuye significativamente, mostrando beneficios como la optimización de servicios públicos, la mejora en la calidad del aire y el mantenimiento predictivo en la manufactura. Cada uno de estos sectores se ha beneficiado del IoT al reducir costos operativos y aumentar la eficiencia en la gestión de recursos.

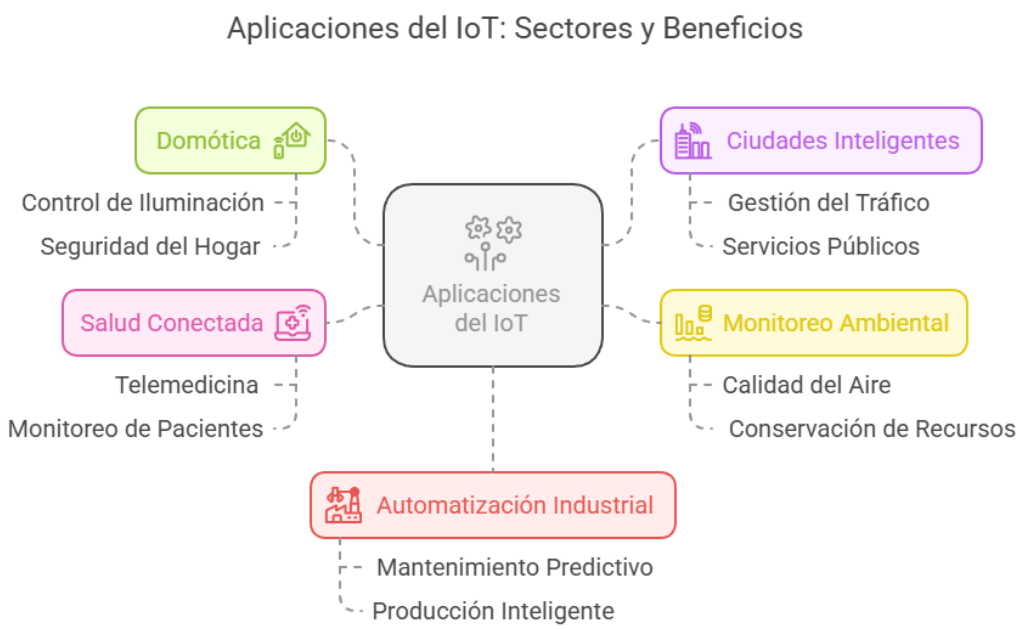


Figura 1.6 Aplicaciones del IoT en distintos sectores, destacando domótica, salud conectada, automatización industrial, monitoreo ambiental y ciudades inteligentes. Elaboración propia.

Para una mayor compresión pondremos como ejemplo un caso práctico por cada uno los sectores antes mencionados:

1. Caso práctico de domótica: Sistema inteligente de control de iluminación y climatización en el hogar.
    a. Sensores de presencia y temperatura instalados en cada habitación envían datos a un sistema central, por ejemplo, a Home Assistant o Node-red.
    b. Las luces y aire acondicionado se encienden solo cuando hay personas presentes y se ajustan automáticamente según la temperatura del ambiente.
    c. Como resultado, se consigue ahorro energético y mayor confort sin necesidad de intervención manual.
2. Caso práctico para la salud: Monitoreo remoto de pacientes críticos.
    a. Pacientes con hipertensión usan un tensiómetro IoT que envía sus valores de presión arterial a la nube.
    b. El médico recibe alertas si los valores superan un umbral crítico y puede revisar tendencias históricas en un dashboard.
    c. Prevención de crisis hipertensivas y seguimiento continuo sin necesidad de visitas presenciales

3. Caso práctico de automatización industrial: Mantenimiento predictivo en una línea de producción.
    a. Motores eléctricos en la planta tienen sensores IoT que miden vibración y temperatura.
    b. Si los datos muestran patrones de desgaste anormal, el sistema genera una alerta para programar el mantenimiento antes de una falla.
    c. Reducción de tiempos muertos y optimización del ciclo de vida de las máquinas.
4. Caso práctico de ciudades inteligentes: Gestión inteligente del tráfico vehicular.
    a. Cámaras y sensores IoT en semáforos recopilan información del flujo vehicular en tiempo real.
    b. Los algoritmos ajustan los tiempos de los semáforos dinámicamente para reducir la congestión.
    c. Disminución de tiempos de viajes, reducción de emisiones contaminantes y mayor seguridad vial.
5. Caso práctico de monitoreo ambiental: Estaciones IoT para medir calidad de aire.
    a. Sensores distribuidos en diferentes zonas de una ciudad miden contaminantes como CO_2, NO_2 y partículas PM2.5
    b. Los datos se envían a una plataforma en la nube que muestra mapas en tiempo real y genera alertas en caso de niveles peligrosos.
    c. Facilita la toma decisiones en políticas ambientales y alerta temprana para población vulnerable.

Consejo Pedagógico:

Tip: En todo sistema IoT, la combinación de monitoreo, conectividad y análisis de datos es crítica para lograr valor agregado real.

1.2 Arquitectura del IoT

1.2.1 Modelos de Arquitectura IoT

La arquitectura de IoT se refiere a la estructura de capas que organiza los componentes del IoT para permitir una comunicación eficiente y segura entre el usuario y la fuente de origen de los datos El Hakim, A. (2018). Cada capa tiene un papel esencial, desde la captura de datos hasta su análisis y uso en aplicaciones específicas. Los modelos de arquitectura pueden clasificarse según niveles, capas, bloques o dominios, dependiendo del enfoque de diseño y aplicación, como se puede apreciar en la siguiente Figura 1.7.

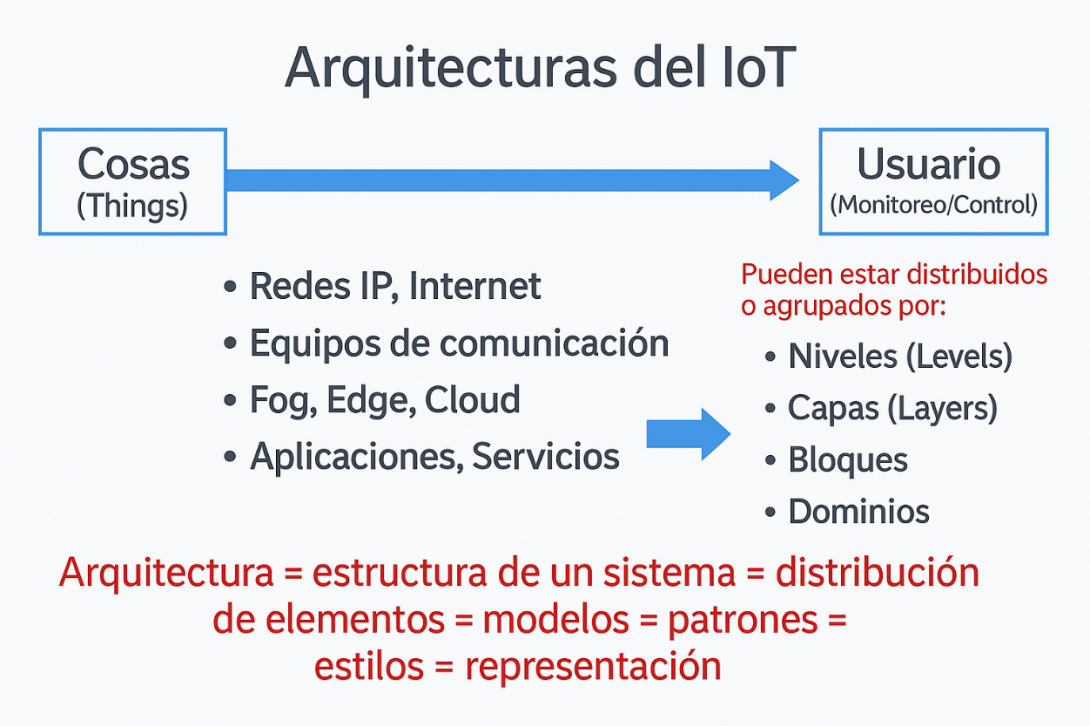


Figura 1.7 Modelo Arquitectónico del IoT: De los Dispositivos al Usuario. Elaboración propia.

Una arquitectura típica de IoT se compone de varias capas que permiten la recolección, transmisión y procesamiento de datos:

- Capa de Percepción: Incluye sensores y actuadores que recopilan información del entorno.
- Capa de Red: Se encarga de la transmisión de datos a través de protocolos como Wi-Fi, Zigbee y LoRaWAN.
- Capa de Procesamiento: Comprende servidores locales y nubes donde se analizan los datos.

- Capa de Aplicación: Interfaces de usuario para el control y la visualización de información.

Existen dos modelos principales de arquitectura en IoT: el modelo en capas y el modelo orientado a servicios. El primero divide las funciones en niveles distintos, lo que facilita la gestión y la comprensión, mientras que el segundo enfatiza la interoperabilidad, siendo ideal para entornos diversos y dinámicos, como se puede apreciar en la Figura 1.8.

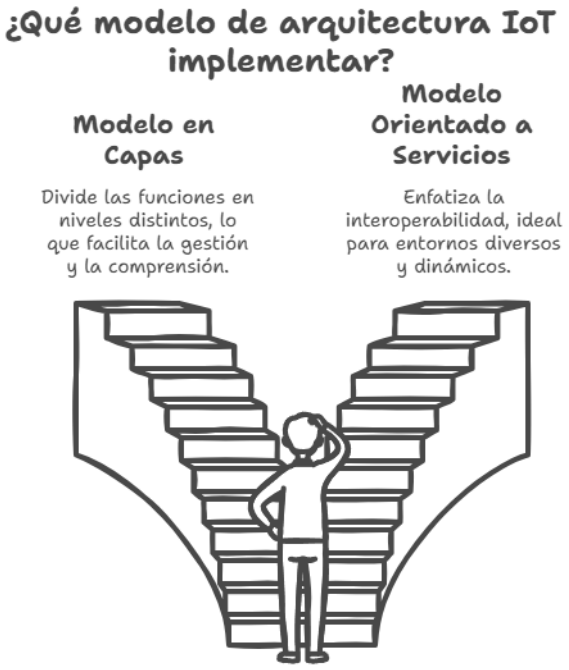


Figura 1.8 Comparación entre los modelos de arquitectura en capas y orientado a servicios en IoT. Elaboración propia.

Asimismo, un enfoque comúnmente utilizado en IoT es la **arquitectura de tres capas,** que aparece en la Figura 1.9, que abarca **Dispositivos IoT**, **Comunicación** y **Aplicaciones**. Este modelo proporciona una estructura clara donde los sensores y actuadores recopilan datos o actúan sobre el sistema, los protocolos de comunicación y la infraestructura de red (alámbrica o inalámbrica) hacen posible la transmisión de los datos entre la capas superiores e inferiores y las plataformas de aplicación procesan, monitorean, almacenan los datos para generar información útil al usuario.

Ejemplo: Un sistema de gestión de energía en edificios inteligentes que recopila datos de sensores inalámbricos, mostrando datos de Voltaje,

Corriente, Potencia y Energía en tiempo real en un dashboard en la cloud y ajusta el consumo en función de la demanda.

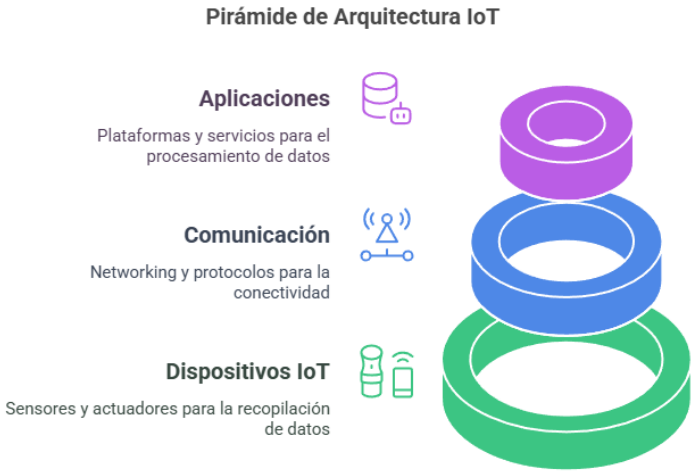


Figura 1.9 Pirámide de Arquitectura IoT, que representa el modelo de tres capas con dispositivos, comunicación y aplicaciones. Elaboración propia.

Existen múltiples arquitecturas definidas para IoT, algunas creadas por instituciones de renombre y otras aceptadas por la comunidad científica. Entre las más destacadas se encuentras la siguientes:

Arquitectura Basada en Servicios (SOA-Based Architecture)

- Basada en Service-Oriented Architecture (SOA).
- Facilita la interoperabilidad entre dispositivos heterogéneos.
- La Figura 1.10 muestra estructura de cuatro capas:

    1. **Capa de sensores (Sensing Layer) → Sensores y dispositivos que capturan datos.**
    2. **Capa de red (Network Layer) → Enlace de comunicación entre dispositivos.**
    3. **Capa de servicio (Service Layer) → Provisión de funcionalidades mediante servicios. Capa de interfaz (Interface Layer) → Interacción con el usuario.**

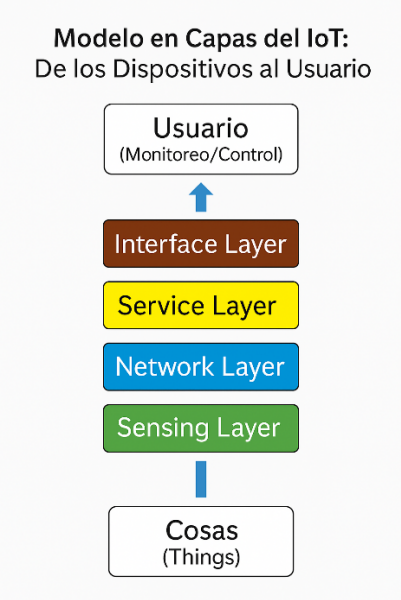


Figura 1.10 Arquitectura IoT SOA, formada por 4 capas. Elaboración propia.

Arquitectura Basada en APIs (API-Oriented Architecture)

- Se basa en el uso de **Web APIs** y el protocolo **REST**.
- Utiliza **JSON** en lugar de XML, lo que lo hace más ligero.
- Soporta protocolos como **CoAP** para IoT de baja latencia.
- La Figura 1.11 muestra estructura **cuatro capas**:

    1. **Capa de sensores (Sensing Layer)**
    2. **Capa de red (Network Layer)**
    3. **Capa de API REST (REST API Layer)**
    4. **Capa de aplicación web (Application Layer)**

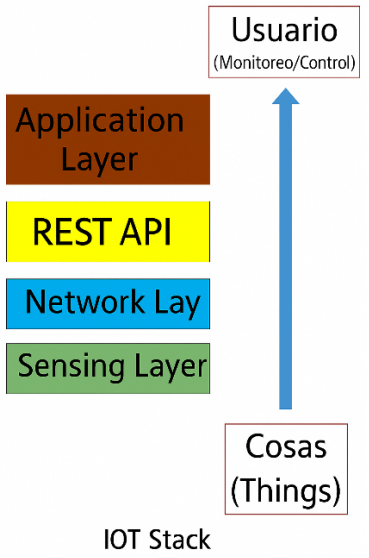


Figura 1.11 Arquitectura IoT API, formada por 4 capas. Elaboración propia.

Arquitectura Estándar oneM2M

- Desarrollada por el **European Telecommunications Standards Institute (ETSI)**.
- Se enfoca en la interoperabilidad entre dispositivos IoT y redes de comunicación
- La Figura 1.12Figura 1.12 muestra estructura de **tres capas principales**:
    1. **Capa de aplicación (Application Layer)**
    2. **Capa de servicios (Services Layer)**
    3. **Capa de red (Network Layer)**

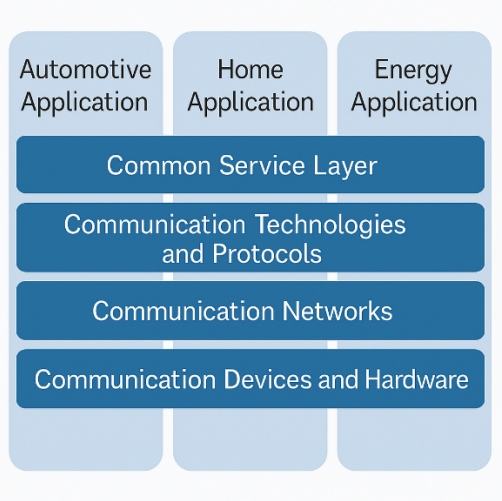


Figura 1.12 Arquitectura IoT oneM2M, formada por 3 capas. Tomadas de oneM2M (2025, 10 de mayo)

Arquitectura IoT del World Forum (IoTWF)

- Creada por **Cisco, IBM y Rockwell**.
- La Figura 1.13 se muestra estructura de **siete capas**:
    1. **Dispositivos físicos y controladores**
    2. **Conectividad**
    3. **Edge Computing**
    4. **Acumulación de datos (almacenamiento)**
    5. **Abstracción de datos**
    6. **Aplicaciones**
    7. **Colaboración y procesos**

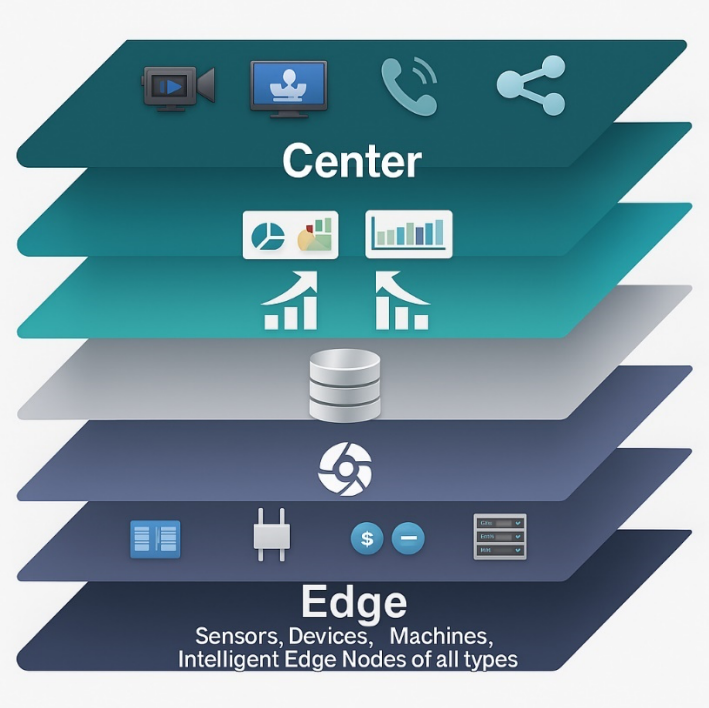


Figura 1.13 Arquitectura IoT IoTWF, formada por 7 capas. Tomadas de Hakim (2018)

Arquitectura de Referencia ITU-T

- Basada en el modelo de capas propuesto por la **International Telecommunication Union (ITU-T)**.
- Organiza la arquitectura IoT en **capas bien definidas** para garantizar interoperabilidad y eficiencia en el procesamiento de datos.

- La Figura 1.14Figura 1.14 muestra estructura de cuatro capas:

1. **Capa de dispositivos (Device Layer) → Sensores y Gateway.**
2. **Capa de red (Network Layer) → Enlace de comunicación entre dispositivos.**
3. **Capa de servicio y soporte de aplicaciones (Service Layer an Applicaciont support layer) Soporte para aplicaciones genéricas y específicas.**
4. **Capa de aplicaciones (Application Layer) → Aplicaciones IoT.**

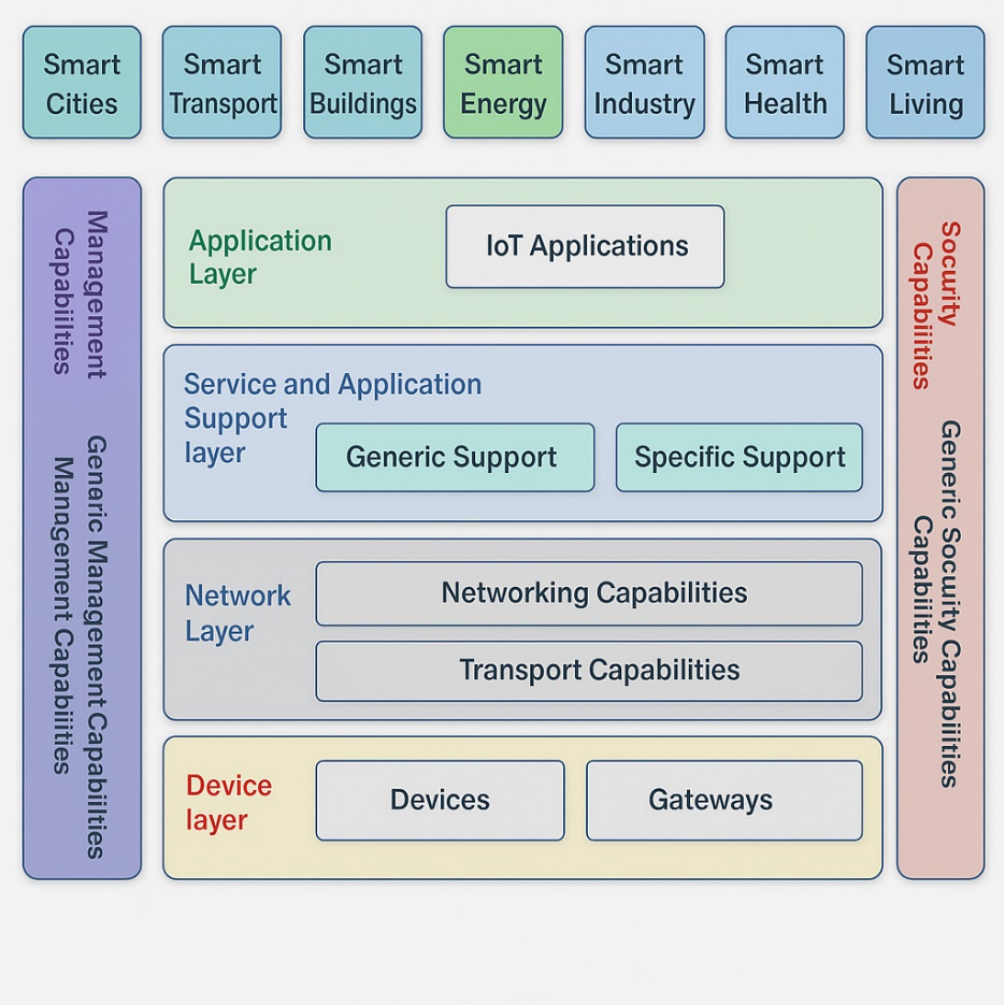


Figura 1.14 Arquitectura IoT de la IUT-T, formada por 4 capas, pero incorpora módulos de gestión, de seguridad y dominios de aplicaciones como ejemplo. Tomadas de Rueda y Talavera Portocarrero (2017)

Existen una gran cantidad de arquitecturas definidas por instituciones y/o autores de artículos científicos, solo una muestra de ellas la que aparece en la Figura 1.15, la cual muestra una clasificación más extensa basado en diferentes capas.

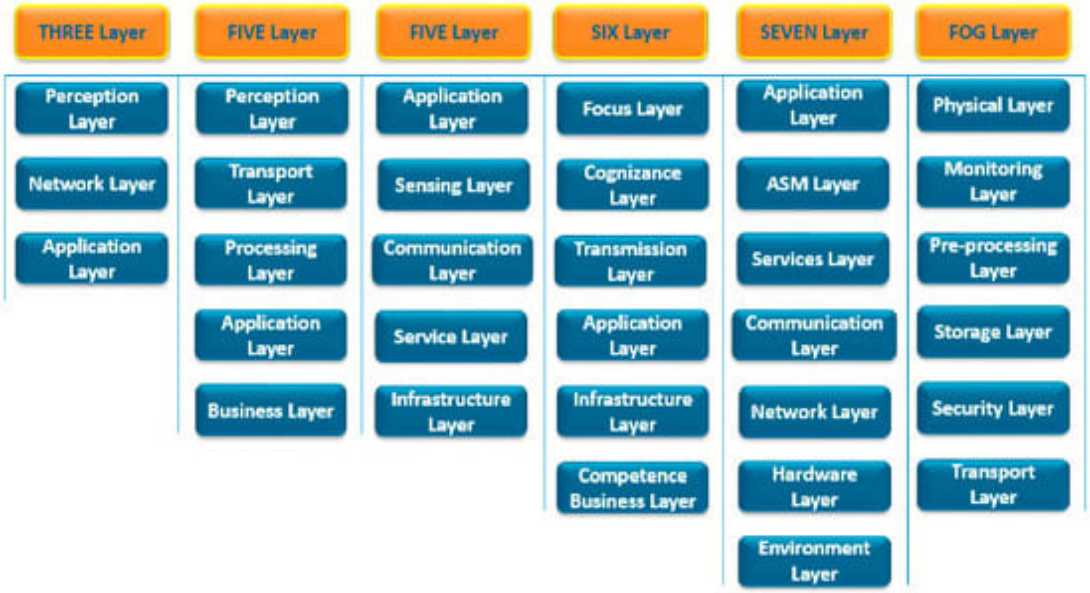


Figura 1.15 Arquitectura con diferentes capas, tomada de Kumar y Mallick (2018)

1.2.2 Componentes Clave de IoT

Una vez que ya conocemos las arquitecturas de IoT, vamos a hacer un recorrido por las diferentes capas de la misma para identificar a los "actores" (componentes de IoT) que pueden estar presente y conocer sus funcionalidades básicas.

Componentes de la capa de dispositivos o percepción.

Comenzando por la capa más baja en el stack de la arquitectura, tenemos la capa de percepción o capa de dispositivos. Aquí, los principales componentes son los sensores, actuadores y procesadores.

- **Sensores**: Dispositivos que capturan información del entorno (temperatura, humedad, movimiento). Son dispositivos que convierten una propiedad física en una señal eléctrica. Ejemplos de variables físicas monitoreadas: Temperatura, sonido, flujo, presión, distancia, luz, etc. y se clasifican en analógicos y digitales, dependiendo del tipo de señal que generan. La Figura 1.16 muestra algunos ejemplos de estos sensores.

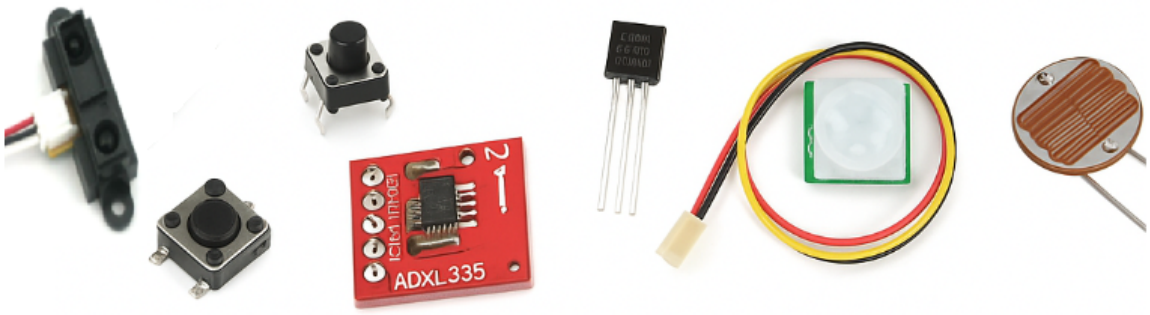


Figura 1.16 Sensores de la capa de dispositivos: RF, botoneras, acelerómetro, temperatura, PIR y de luminosidad. Elaboración propia.

La Figura 1.17Figura 1.17, muestra los diferentes sensores integrados en un típico smartphone, como acelerómetros, giroscopios, sensores de proximidad y sensores de luz ambiental.

Figura 1.17 Ejemplo de sensores integrados en un smartphone. Elaboración propia.

Un concepto adicional debemos añadir en esta parte: el canal de medición, formado por varios elementos como se observa en la siguiente Figura 1.18 que garantizan la integridad de la señal correspondiente a la variable medida.

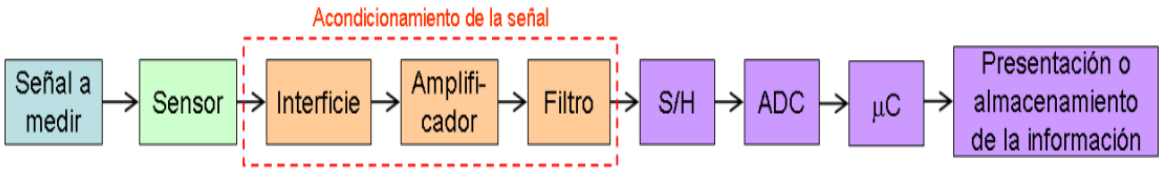


Figura 1.18 Canal de medición de un sistema embebido usado en la capa de dispositivos. Elaboración propia.

- Señal: Magnitud física a medir (Presión, Temperatura, posición, tensión, etc.).
- Sensor: convierte magnitud física a eléctrica
- Interfaz: Adecua la señal, por ejemplo, de corriente a voltaje
- Amplificador: Adecua niveles de voltaje a la entrada del ADC
- Filtro: Elimina componentes de ruido en el dominio de la frecuencia.
- S/H: Sample & Hold, muestreo y retención de señal a medir
- ADC: Convertidor análogo-digital, entrega una señal digital proporcional a la magnitud física.
- uC: procesador digital (microprocesador, microcontrolador, PLC, etc.)
- Interfaz: Optoacopladores (Protección)
- Uso de amplificadores operaciones, fundamentalmente los de instrumentación
- Filtros pasa-bajos, pasa-altos o pasa-bandas.
- ADC: Conversor análogo-digital, #bits, tiempo conversión, # de canales, canales diferenciales, errores, rango de entrada.
- Sensores con salida serial: I2C, SPI, 1 wire, entre otros.

❖ **Actuadores**: Permiten realizar acciones físicas como encender luces o ajustar válvulas. Transforman señales eléctricas en acciones físicas, como: Luz (LEDs), movimiento (motores), calor (resistencias), sonido (bocinas) Los actuadores pueden ser analógicos (DAC) o digitales (PWM, salidas binarias). La Figura 1.19 muestran algunos ejemplos de estos sensores.

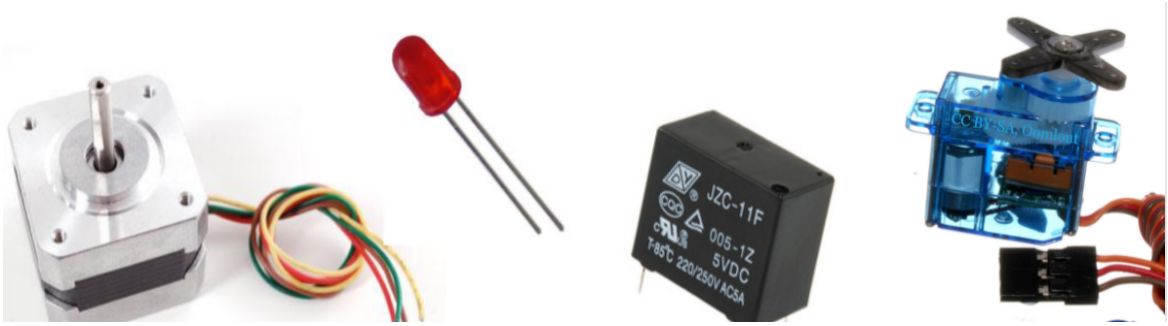


Figura 1.19 Ejemplo de algunos actuadores que encontramos la capa de dispositivos. De Izquierda a derecha encontramos un motor de paso, un LED, un relay y un servo motor. Elaboración propia.

- Dispositivos de Procesamiento: IoT usa distintos tipos de procesadores según los requerimientos de computación: Microcontroladores (MCU): PIC, STM32, Sistemas en Chip (SoC): ESP32, NRF52, Computadoras de placa única (SBC): Raspberry Pi, Beaglebone o PLCs y FPGA para el control industrial, los cuales aparecen en la Figura 1.20.

Figura 1.20 Ejemplo de algunos procesadores que encontramos la capa de dispositivos. De Izquierda a derecha encontramos un PIC, STM32, ESP32, NRF52 y Rpi. Elaboración propia.

Estos CPUs son usados en dispositivos IoT que típicamente hacen uso de baterías, bajo consumo, poca memoria, latencias, reducida capacidad de procesamiento (comparada con procesadores usados en PC y Smartphones actuales), uso sistemas operativos en tiempo Real (RTOS), baja velocidad de transmisión de datos y son categorizados en un nuevo concepto llamado: **constrain device** (ingles) y han sido categorizados por clases en la RFC 7228 Information On RFC 7228 » RFC Editor (2025, 10 de mayo).

- Clase 0: Memoria datos << 10 KB y Código de programa << 100 KB
- Clase 1: Memoria datos ~ 10 KB y Código de programa ~ 100 KB
- Clase 2: Memoria datos ~ 50 KB y Código de programa ~ 250 KB

Todos estos elementos en su conjunto, ubicados en la capa de dispositivos, forman un ecosistema muy potente y diverso, típicamente son llamados como

motes (por su pequeño tamaño) o Smart transducers (transductores inteligentes) o Smart sensors (sensores inteligentes). En la Figura 1.21Figura 1.21 se observa un esquema generalizado del mismo.

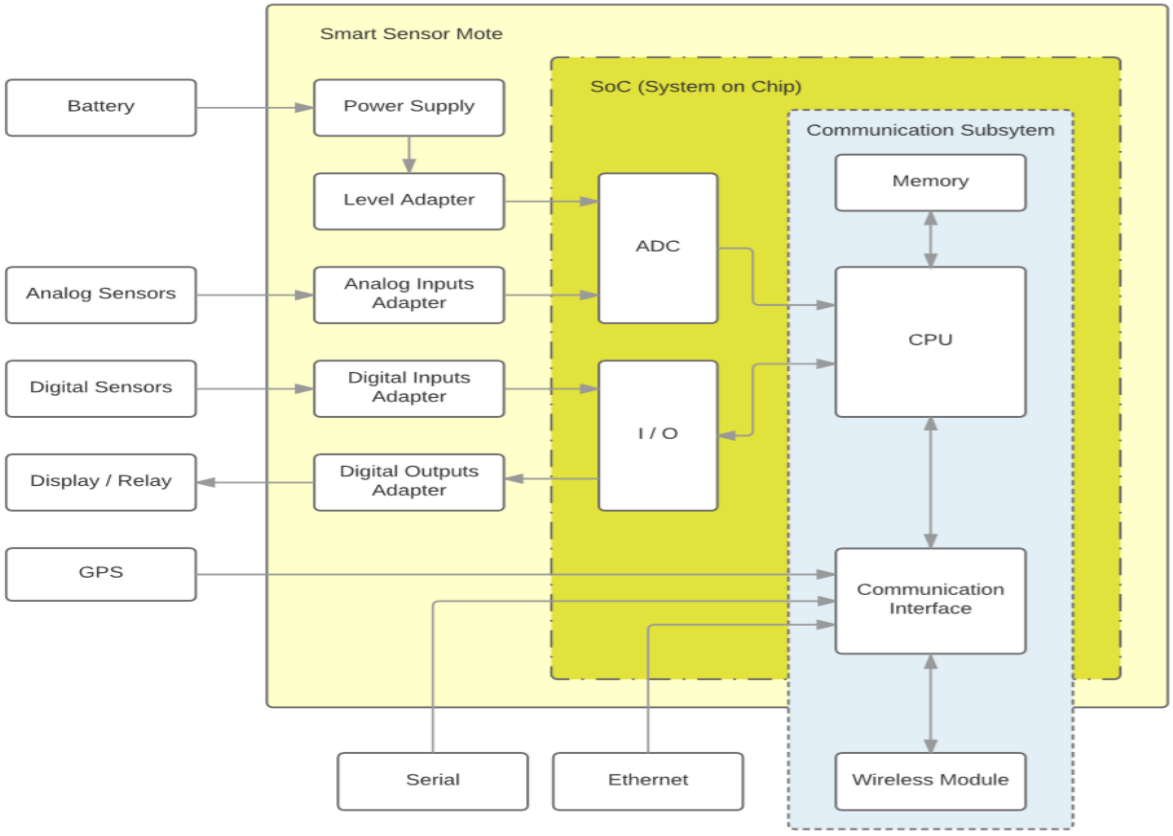


Figura 1.21 Smart transducer. tomada de Hernández-Rojas, Fernández-Caramés, Fraga-Lamas y Escudero (2017).

Cada uno de estos componentes desempeña un papel crucial en el funcionamiento de IoT. Los sensores recopilan datos en tiempo real, los actuadores responden a dichos datos generando una acción física, y los dispositivos de procesamiento analizan la información para optimizar el rendimiento del sistema.

Componentes de comunicaciones.

En esta capa los dispositivos que encontramos están relacionados con la infraestructura de red de la aplicación IoT en sí. Es decir, si los dispositivos de la capa de percepción utilizan módulos de comunicación Wifi (muchas veces integrados en los Smart transducers), entonces, encontraremos switches, routers y Access points según la familia de Wifi soporta dígase Wifi 4, Wifi 5, Wifi 6 y recientemente la Wifi 7. Algo similar si el medio es cableado vía

Ethernet soportando diversos protocolos, incluso industriales como Modbus, Profibus, etc.

Por ejemplo, un ESP32, es capaz de comunicarse vía Wifi hacia la capa de aplicaciones, por tanto, dicho dispositivo debe conectarse a una red Wifi en modo Station o Access Point.

En cuanto a otros protocolos de comunicación inalámbricos que no disponen de capacidad IP (como Wifi y Ethernet), entonces se requiere un dispositivo denominado Gateway IOT o Border Router, ya que estos dispositivos pasan en frontera y muchos autores lo han ubicado tanto en la capa de comunicaciones como la de dispositivos. En mi caso, prefiero ubicarlos en la capa de comunicaciones, para poder estudiarlos junto con los protocolos de comunicación propios de esta capa más usados en IoT y que veremos en el próximo capítulo. Aquí, encontramos dispositivos BLE (Bluetooth Low Energy), Zigbee, Treath, entre otros. Muchas veces los fabricantes ofertan sus Gateways propietarios cerrados, como se aprecian en la Figura 1.22 Componentes típicos de la capa de networking. De izquierda a derecha encontramos un gateway Zigbee, Border router Thread y un router Wifi 6. Elaboración propia., no obstante, podemos construir nuestro propios gateways usando plataformas SBC robustas como Raspberry Pi, incluso con un ESP32.

Figura 1.22 Componentes típicos de la capa de networking. De izquierda a derecha encontramos un gateway Zigbee, Border router Thread y un router Wifi 6. Elaboración propia.

En los próximos epígrafes abundaremos más al respecto.

Componentes de la capa de aplicación.

En esta capa superior, encontramos los dispositivos con mayor computo posible, desde servidores hasta la nube. Por tanto, varía mucho el alcance del proyecto, el volumen de datos a procesar y las aplicaciones IoT propiamente

necesarias. Dado el caso una Rpi o un teléfono inteligente puede correr las aplicaciones en esta capa, como se aprecia en la Figura 1.23.

Figura 1.23 Capa de aplicaciones, encontramos tanto plataformas de hardware como aplicaciones, de izquierda a derecha tenemos: servidores en la nube, servidores, locales, plataformas de bajo recursos como RRi, teléfonos inteligente y el dashboard Grafana. Elaboración propia

Las aplicaciones, infraestructuras y servicios disponibles en IoT en la nube no serán tratadas a fondo en el marco de este libro introductorio a IoT.

Para resumir esta parte del capítulo nos vamos a referir a la Figura 1.24, donde podemos identificar los diferentes dispositivos IoT antes estudiados ubicados en la capa correspondiente de una arquitectura de 3 capas según la aplicación o proyecto dado. Es importante aclarar esto ya que algunos elementos pueden ser usados según su funcionalidad en más de una capa. Analicemos dicha figura, supongamos un proyecto de domótica, donde se desea el detectar el movimiento o presencia de personas no deseadas en cierto pasillo de una casa. Cuando una persona es detectada por el sensor PIR, el cual está conectado a un dispositivo controlador ESP32, este envía una notificación de alarma hacia un celular, la cual tambien aparece en el dashboard remoto de la empresa de seguridad contratada y de forma local se enciende un LED y por medio del relé se activa una alarma sonora.

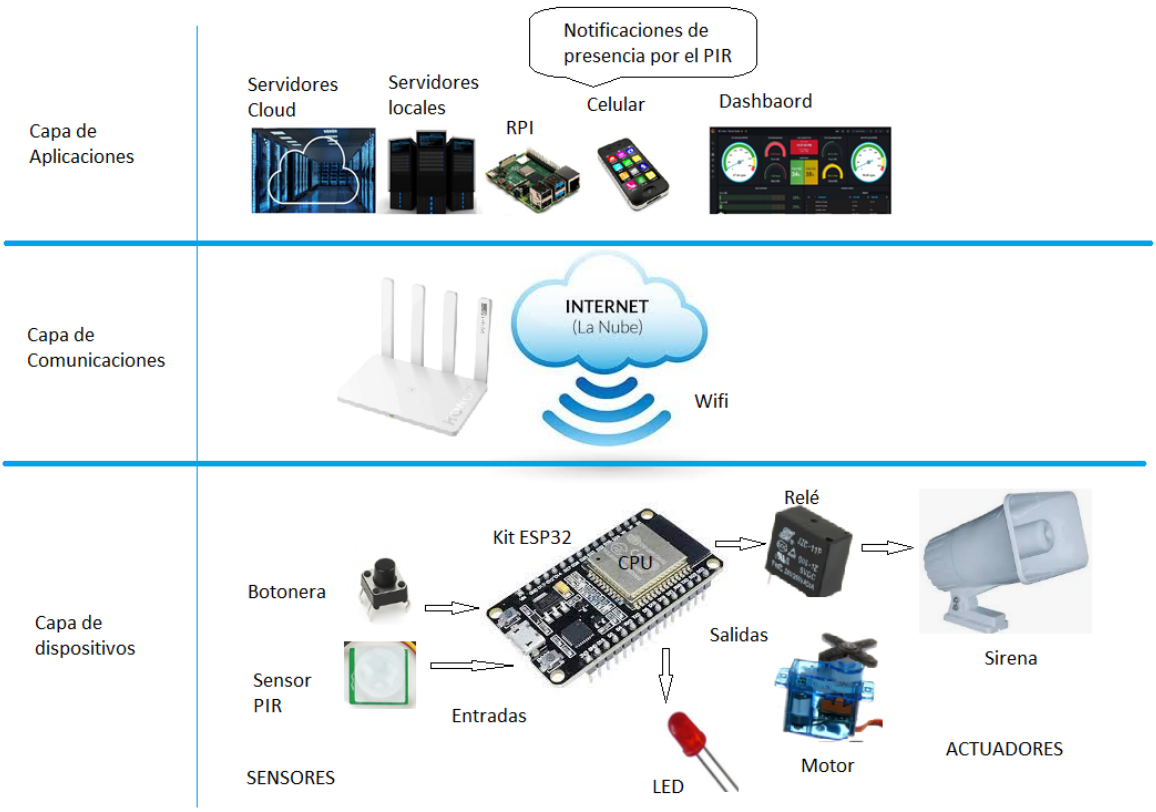


Figura 1.24 Dispositivos IoT ubicados en las diferentes capas de un típica arquitectura IoT de tres capas. Elaboración propia

1.2.3 Infraestructura de Red y Conectividad en IoT

Subiendo en el stack de IoT nos encontramos típicamente con la capa de comunicaciones. Es aquí donde aparece un nuevo elemento llamado gateway, que analizaremos en este epigrafe, dado que esta capa de networking garantiza independientemente de la tecnología de comunicación usada que los datos recolectados por la capa inferior de los "motes" y que no son procesados en dicha capa puedan escalar a la capa de aplicaciones.

Las tecnologías de comunicación en IoT son muy diversas y cambiantes, dinámicas, IoT se adapta a ellas y ellas aparecen para cubrir nuevas necesidades de IoT. Entre las tecnologías más comunes encontramos a:

- Wi-Fi y Ethernet: Para conexiones de alta velocidad.
- LPWAN (Low Power Wide Area Networks): Como LoRaWAN y Sigfox, utilizadas para sensores de largo alcance con bajo consumo energético.
- Redes celulares (4G/5G): Permiten la conectividad de IoT en movilidad y aplicaciones industriales.

Cada una de estas tecnologías, que aparecen en la Figura 1.25, ofrecen ventajas según el contexto de implementación. Wi-Fi es ideal para redes locales con alta velocidad, mientras que LPWAN se utiliza en aplicaciones donde la eficiencia energética y el largo alcance son prioritarios. Por su parte, las redes celulares permiten la conectividad en movilidad y en áreas extensas donde otras tecnologías no llegan.

Ejemplo: Sensores de calidad del aire desplegados en una ciudad que transmiten datos mediante redes LoRaWAN a una plataforma de gestión

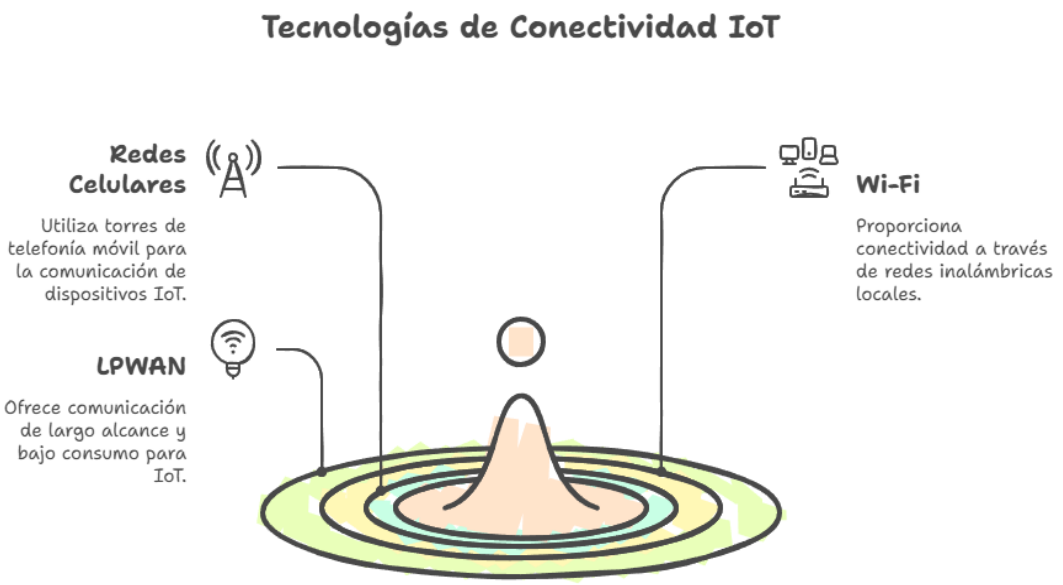


Figura 1.25 Tecnologías de conectividad en IoT, destacando Wi-Fi, LPWAN y redes celulares. Elaboración propia.

ambiental.

Gateway IoT

El concepto de gateway típicamente en networking está relacionado con la puerta de enlace, digamos la dirección IP de nuestro router con salida de nuestro proveedor de internet. Pero, en IoT el gateway adquiere una nueva connotación y aumento considerable de funciones y responsabilidades. Debemos recordar que un sistema IoT sin comunicación no es IoT. A continuación, algunos elementos de un gateway IoT mostrado en la Figura 1.26Figura 1.25:

- Puerta de enlace de comunicaciones entre la capa de dispositivos hacia los niveles superiores
- Es un dispositivo físico capaz de encapsular diferentes protocolos.

- Necesidad de compatibilidad de medio de comunicación y de protocolos de comunicación
- Brinda acceso a internet a todo dispositivo que no la tenga, sensores, actuadores, etc.
- Típicamente dispone de Wifi y/o Ethernet
- Tiene la capacidad de admitir conexiones de diferentes módulos de comunicación, vía puertos seriales UART, I2C y SPI
- Por su funcionalidad, puede ser integrado en diferentes tecnologías de hardware, incluso en servidores.

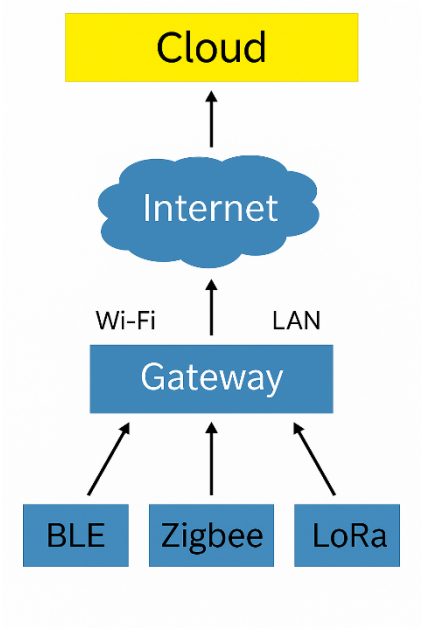


Figura 1.26 Gateway IoT. Permite comunicar dispositivos que "hablan" diferentes "idiomas", es una especie de traductor IoT. Por ejemplo, Zigbee con Wifi. Elaboración propia.

1.3 Protocolos y Estándares en IoT

1.3.1 Estándares de Interoperabilidad

Para garantizar la interoperabilidad, se han desarrollado estándares que se agrupan en tres categorías principales:

- Estándares de Comunicación: Protocolos que aseguran el intercambio eficiente de datos entre dispositivos IoT.
- Estándares de Datos: Formatos que facilitan la compresión e interpretación de la información entre distintos sistemas.
- Estándares de Seguridad: Medidas que protegen la integridad y privacidad de la comunicación y los datos transmitidos.

Los estándares mostrados en la Figura 1.27 permiten que los dispositivos IoT de diferentes fabricantes puedan interactuar sin problemas, promoviendo la escalabilidad y la integración de nuevas tecnologías dentro del ecosistema IoT.

Figura 1.27 Representación de los principales estándares de interoperabilidad en IoT, abarcando comunicación, datos y seguridad. Elaboración propia

1.3.2 Protocolos de Comunicación en IoT

Los protocolos de comunicación en IoT pueden clasificarse según el alcance y el consumo energético, permitiendo seleccionar la mejor opción para cada aplicación específica, como se puede apreciar en la Figura 1.28:

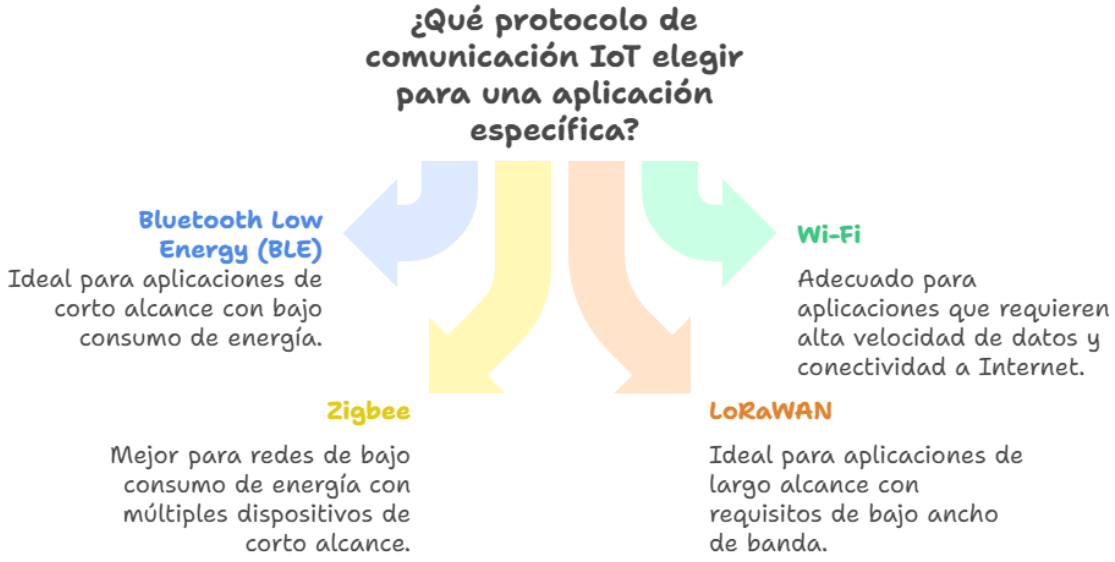


Figura 1.28 Comparación de protocolos de comunicación en IoT, destacando su uso en diferentes aplicaciones según alcance y consumo energético. Elaboración propia.

La Figura 1.29 permite ver la relación entre algunos protocolos IoT con el modelo OSI de la ISO y en la Figura 1.30 el alcance de las mismas. En el próximo capítulo estudiaremos con mayor profundidad alguno de ellos.

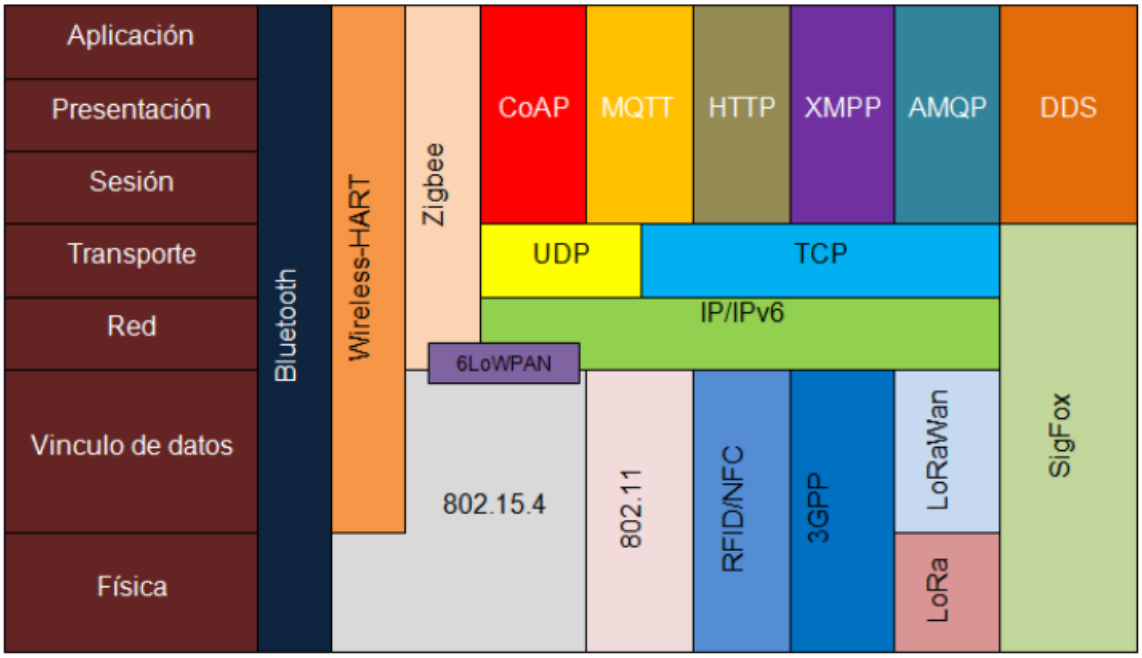


Figura 1.29 Comparación de protocolos IoT con el modelo OSI, adaptada de Mocnej, Pekar, Seah y Zolotova (2018).

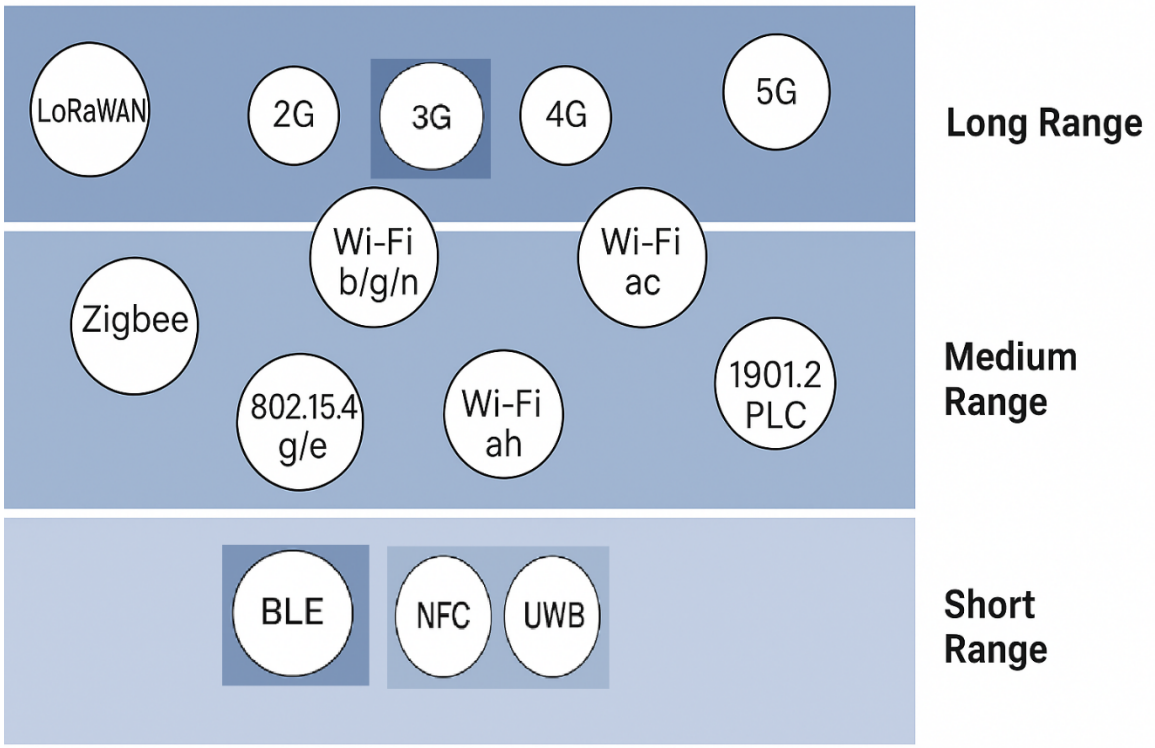


Figura 1.30 Comparación de protocolos IoT según su rango de alcance. Elaboración propia.

Estos protocolos permiten la interconexión de dispositivos IoT en distintos entornos, optimizando el consumo energético y la transmisión de datos. Por

ejemplo, MQTT es ideal para redes limitadas en ancho de banda, HTTP/REST es más pesado, usado en entornos de integración web y BLE es muy eficiente en consumo energético ideal para wearables.

1.3.3 Redes de Sensores Inalámbricos (WSN)

Las Redes de Sensores Inalámbricos, conocidas como WSN por sus siglas en inglés (Wireless Sensor Networks), constituyen una tecnología esencial en el ecosistema del Internet de las Cosas (IoT). Estas redes permiten la conexión y comunicación de múltiples nodos sensores que monitorean variables del entorno y transmiten datos de forma inalámbrica hacia un nodo central o pasarela (gateway).

Una WSN se compone típicamente de:

- **Nodos sensores**: Dispositivos con capacidad de detección, procesamiento y comunicación.
- **Nodo coordinador o gateway**: Punto central que recibe la información y la transmite a la nube o a otros sistemas de procesamiento.
- **Medio inalámbrico de comunicación**: Puede incluir tecnologías como Zigbee, Wi-Fi, LoRa, Bluetooth Low Energy (BLE), entre otras.

Características de una WSN en IoT

- **Despliegue masivo**: Los nodos pueden estar distribuidos en grandes áreas geográficas.
- **Topologías adaptativas**: Se pueden organizar en mallas, árboles o estrella, dependiendo del entorno y la aplicación.
- **Consumo energético optimizado**: El diseño de protocolos y hardware busca maximizar la autonomía de los nodos.
- **Procesamiento distribuido**: Parte del análisis puede realizarse localmente en los nodos, reduciendo la necesidad de envío continuo.

Aplicaciones de las WSN en IoT

- **Agricultura de precisión**: Monitoreo de humedad del suelo, temperatura y condiciones ambientales.
- **Domótica**: Sensores para iluminación, seguridad, presencia y confort térmico.

- **Salud**: Ropa inteligente, sensores biomédicos, monitoreo de pacientes a distancia.
- **Ciudades inteligentes**: Monitoreo ambiental, tráfico vehicular, luminarias públicas.

Desafíos y Consideraciones

- **Escalabilidad**: Soportar cientos o miles de nodos sin comprometer la eficiencia.
- **Confiabilidad y latencia**: Garantizar que los datos lleguen de forma precisa y oportuna.
- **Seguridad**: Protección de datos y autenticación de nodos en redes abiertas.

La Figura 1.31 ilustra la arquitectura básica de una WSN en un escenario IoT típico. En ella se observan múltiples nodos sensores distribuidos espacialmente que capturan datos del entorno (como temperatura o humedad) y los transmiten de manera inalámbrica a través de protocolos de bajo consumo energético. Todos los datos convergen en un nodo coordinador o gateway, que actúa como intermediario hacia una plataforma de gestión en la nube. Esta estructura resalta la eficiencia de la WSN para monitorear variables distribuidas, facilitando su integración con servicios avanzados como inteligencia artificial o análisis predictivo.

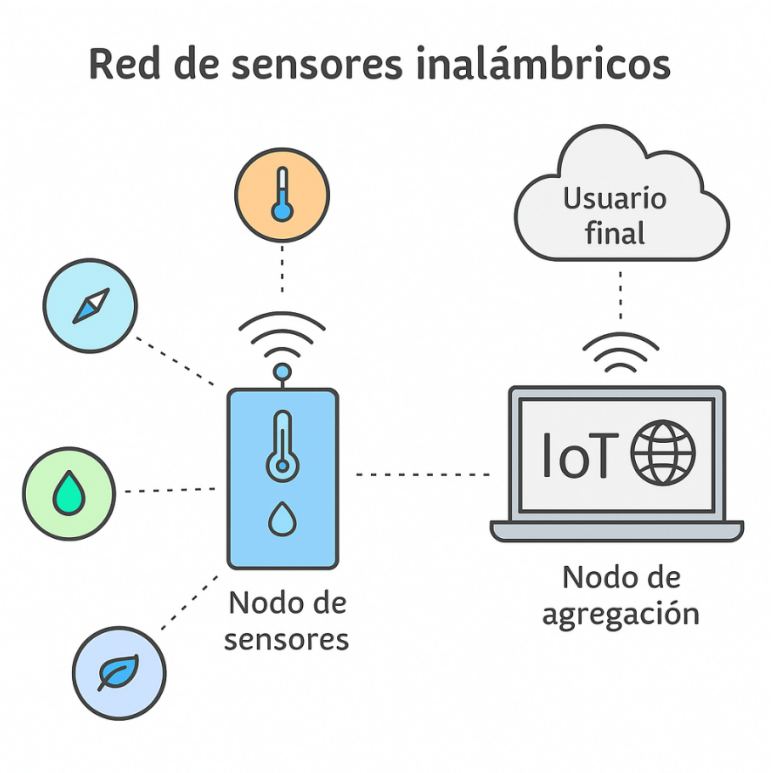


Figura 1.31 Representación de una Red de Sensores Inalámbricos (WSN) en un entorno IoT, mostrando la interacción entre nodos, gateway y servicios en la nube. Elaboración propia.

Existen varias tecnologías de comunicación inalámbricas aplicables a una red de sensores inteligentes, es decir, donde proliferan los "constrain devices", basados típicamente en microcontroladores de bajo consumo y usando baterías. A continuación, explicaremos algunas de las tecnologías más populares en las WSN actuales.

1.3.4 Wi-Fi

Wi-Fi es una de las tecnologías inalámbricas más utilizadas en IoT debido a su alta velocidad de transmisión y amplia compatibilidad. En el contexto de WSN, permite la conexión de nodos sensores en entornos con infraestructura de red existente, como edificios o fábricas. Aunque presenta un mayor consumo energético que otros protocolos, es ideal para aplicaciones donde la disponibilidad de energía no es una limitación.

Wi-Fi (Wireless Fidelity) es una tecnología de comunicación inalámbrica basada en el estándar IEEE 802.11. Es ampliamente utilizada en entornos IoT por su alta tasa de transferencia de datos y compatibilidad universal. A pesar de su consumo energético relativamente alto, su facilidad de integración y

cobertura lo hacen ideal para aplicaciones donde la disponibilidad de energía y el ancho de banda son prioritarios, como en sistemas domóticos o instalaciones industriales.

Modos de operación

Wi-Fi puede operar en dos modos principales:

- **Station (STA)**: El dispositivo actúa como cliente y se conecta a un punto de acceso existente.
- **Access Point (AP)**: El dispositivo crea su propia red Wi-Fi, permitiendo que otros dispositivos se conecten a él.

La Figura 1.32Figura 1.32 muestra los dos modos fundamentales en que un dispositivo con capacidad Wi-Fi puede operar dentro de una red IoT. En el modo Station (STA), el nodo sensor se conecta a un punto de acceso existente, como un router o un gateway, para enviar sus datos hacia la nube u otro sistema. En contraste, en el modo Access Point (AP), el propio nodo actúa como centro de conexión y permite que otros dispositivos se conecten directamente a él, formando una red local. Ambos modos son esenciales para arquitecturas flexibles de WSN: STA es común en entornos con infraestructura establecida, mientras que AP resulta útil para despliegues autónomos o provisionales donde no se dispone de red fija.

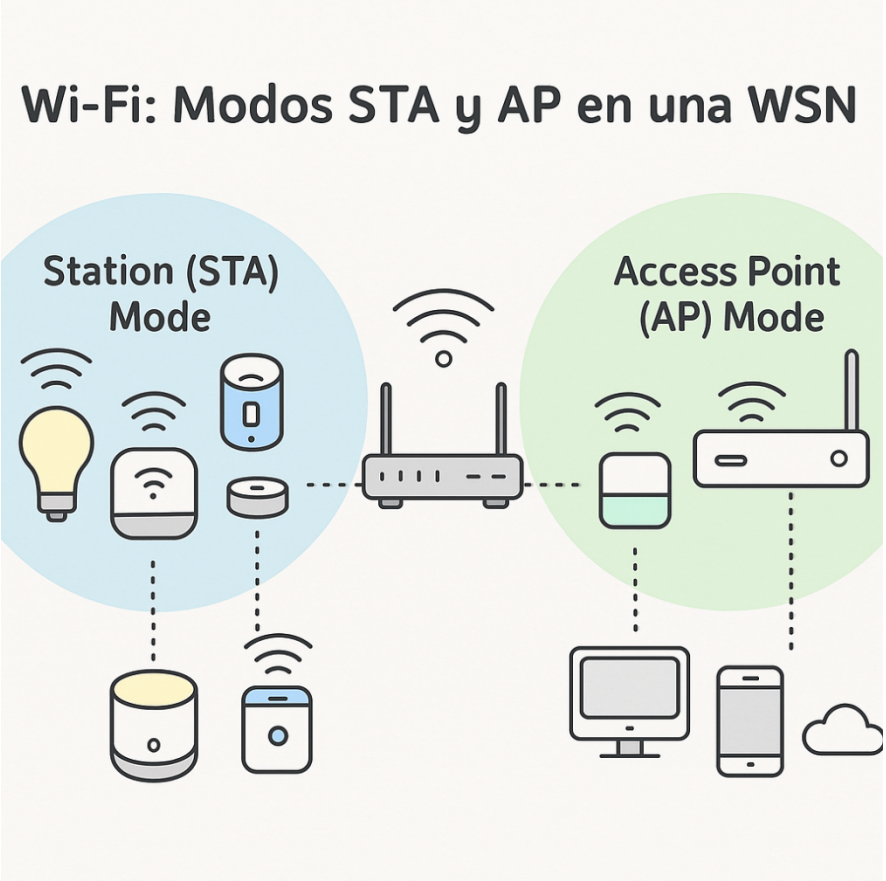


Figura 1.32 Modos de operación Station (STA) y Access Point (AP) en una red de sensores inalámbricos (WSN). Elaboración propia.

Evolución de estándares Wi-Fi

La tecnología Wi-Fi ha evolucionado significativamente para adaptarse a los requerimientos de conectividad actuales, como se puede apreciar en la Tabla 1:

Tabla 1.1 Evolución de los estándares Wifi

Estándar	Año de lanzamiento	Velocidad máxima	Frecuencia	Características principales
Wi-Fi 4 (802.11n)	2009	600 Mbps	2.4 / 5 GHz	Primer estándar con soporte para MIMO
Wi-Fi 5 (802.11ac)	2014	3.5 Gbps	5 GHz	Mayor velocidad y eficiencia espectral

Wi-Fi 6 (802.11ax)	2019	9.6 Gbps	2.4 / 5 GHz	OFDMA, mejor eficiencia y gestión de dispositivos
Wi-Fi 6E	2020	9.6 Gbps	6 GHz	Mayor capacidad y canales más amplios
Wi-Fi 7 (802.11be)	2024	> 30 Gbps	2.4 / 5 / 6 GHz	Latencia ultra baja y canales de hasta 320 MHz

Estos avances han permitido que Wi-Fi continúe siendo una opción viable en escenarios IoT que demandan altas velocidades, especialmente en aplicaciones multimedia, Edge computing o automatización con sensores complejos.

1.3.5 Bluetooth Low Energy (BLE)

BLE es una evolución de la tecnología Bluetooth diseñada para reducir significativamente el consumo energético. Se utiliza ampliamente en dispositivos portátiles y sensores biométricos, permitiendo la transmisión de datos con baja latencia en rangos cortos. Su compatibilidad con dispositivos móviles lo convierte en una opción atractiva para WSN personales o domésticas.

Bluetooth Low Energy (BLE) es una tecnología inalámbrica desarrollada como parte de la especificación Bluetooth 4.0. Fue diseñada específicamente para aplicaciones que requieren bajo consumo energético y comunicación esporádica, como dispositivos portátiles, sensores biomédicos y nodos en redes de sensores inalámbricos y opera en el rango de 2.4 a 2.4825 GHZ con 40 canales de 2 Mhz.

A diferencia del Bluetooth clásico, BLE permite mantener dispositivos en modo de bajo consumo durante largos periodos, activándose únicamente para transmitir o recibir pequeños paquetes de datos. Esto lo convierte en una tecnología ideal para soluciones IoT que necesitan operatividad continua sin reemplazo frecuente de baterías.

Características fundamentales de BLE:

- **Bajo consumo de energía**: Optimizado para aplicaciones donde la duración de la batería es crítica.

- **Alcance medio**: Usualmente entre 10 a 100 metros, dependiendo del entorno y configuración.
- **Baja latencia**: Requiere poco tiempo para establecer una conexión, útil para respuestas rápidas.
- **Topología punto a punto o estrella**: El dispositivo central se comunica directamente con periféricos.
- **Compatibilidad con smartphones**: Permite integración sencilla con aplicaciones móviles.

Configuraciones comunes en dispositivos BLE:

1. **Peripheral**: Es el dispositivo que anuncia su presencia. Generalmente un sensor, baliza o wearable.
2. **Central**: Dispositivo que escanea, detecta y se conecta a uno o más periféricos. Por ejemplo, un smartphone o gateway.
3. **Broadcaster/Observer**: Comunicación unidireccional. El broadcaster transmite y el observer solo escucha, sin establecer conexión.

La Figura 1.33Figura 1.33 representa un escenario típico en el que un dispositivo BLE actúa como beacon (baliza) emitiendo paquetes de datos periódicos en modo broadcast. En este caso, el sensor mide una variable ambiental, como la temperatura, y transmite los valores de forma continua sin requerir una conexión persistente. Un teléfono móvil, actuando como observer, recibe esta información pasivamente sin necesidad de emparejamiento, lo que permite una comunicación rápida, eficiente y de bajo consumo energético, ideal para aplicaciones en monitoreo ambiental, retail o geolocalización.

Figura 1.33 Comunicación de un sensor inteligente de temperatura usando un Beacon BLE. Elaboración propia.

Perfiles y servicios

BLE opera a través de perfiles estandarizados, lo que garantiza la interoperabilidad entre dispositivos. Ejemplos incluyen:

- **Heart Rate Profile (HRP)** para monitores de ritmo cardíaco.
- **Battery Service** para indicar nivel de batería.
- **Environmental Sensing** para datos como temperatura, humedad o presión atmosférica.

Aplicaciones en IoT

BLE se encuentra presente en:

- Sistemas de monitoreo de salud y fitness.
- Sensores ambientales domésticos o industriales.
- Etiquetas de localización y rastreo.
- Sistemas de control de acceso sin contacto.

BLE ha tenido una evolución significativa desde su introducción con la versión Bluetooth 4.0. Esta evolución ha estado enfocada principalmente en ampliar la capacidad, eficiencia y aplicaciones de la tecnología, especialmente en entornos de IoT. La siguiente línea de tiempo resume los principales hitos de BLE desde 2010 hasta la actualidad:

- **BLE 4.0 (2010)**: Introducción oficial de BLE. Orientado al bajo consumo energético para dispositivos simples como sensores.

- **BLE 4.1 (2013)**: Mejora en la coexistencia con redes LTE y soporte para rol dual (central y periférico simultáneo).
- **BLE 4.2 (2014)**: Seguridad mejorada con privacidad y cifrado, y aumento de la velocidad de transferencia de datos.
- **BLE 5.0 (2016)**: Introducción de rangos extendidos (hasta 4 veces más) y velocidad duplicada (2 Mbps), permitiendo la de difusión de mensajes en redes más grandes.
- **BLE 5.1 (2019)**: Agrega capacidades de localización más precisa gracias a los métodos de dirección de señal (AoA/AoD).
- **BLE 5.2 (2020)**: Inclusión de canales lógicos y soporte para el nuevo estándar de audio (LE Audio).
- **BLE 5.3 (2021)**: Mejoras en eficiencia energética y simplificación de procesos de emparejamiento y gestión de roles.
- **BLE 6.0 (2023)**: Introducción de características avanzadas para redes masivas de IoT, incluyendo mejoras en seguridad, descubrimiento de servicios y rendimiento en ambientes densos.
- **BLE 6.1 (2024)**: Aporta mejoras en la gestión simultánea de múltiples conexiones, reducción de latencia, y soporte ampliado para topologías distribuidas y actualizaciones inalámbricas optimizadas.

Esta evolución demuestra que BLE continúa adaptándose a las necesidades crecientes del ecosistema IoT, incorporando mejoras en seguridad, rendimiento, localización y consumo energético. Gracias a su eficiencia energética, compatibilidad con dispositivos móviles y facilidad de implementación, BLE se posiciona como una de las tecnologías más relevantes para soluciones IoT personales, domésticas e incluso comerciales. Se puede profundizar aún más en las especificaciones oficiales en Bluetooth SIG (2025, 10 de mayo).

1.3.6 Zigbee

Zigbee es un protocolo de comunicación inalámbrico basado en el estándar IEEE 802.15.4, diseñado específicamente para aplicaciones de bajo consumo energético, bajo ancho de banda y bajo costo. Es ampliamente utilizado en redes de sensores inalámbricos (WSN), sistemas de automatización del hogar (domótica), entornos industriales y dispositivos conectados en general.

Una de las características más distintivas de Zigbee es su capacidad para operar en topologías de red de tipo malla, lo que le permite extender el alcance de la red mediante la retransmisión de mensajes a través de nodos

intermedios. Esta arquitectura mejora la confiabilidad y cobertura, ya que los datos pueden seguir rutas alternativas en caso de fallo de un nodo.

Características técnicas de Zigbee:

- **Frecuencias de operación**: 2.4 GHz a nivel global, aunque también puede operar en 868 MHz (Europa) y 915 MHz (EE.UU.).
- **Velocidad de datos**: Hasta 250 kbps (en 2.4 GHz).
- **Rango típico**: 10–100 metros, ampliable con topología de malla.
- **Número máximo de nodos**: Hasta 65.000 nodos por red.
- **Topologías compatibles**: Estrella, árbol y malla.
- **Seguridad**: Basada en AES-128 para cifrado de datos.

Componentes de una red Zigbee:

1. **Coordinador Zigbee**: Nodo central encargado de iniciar la red, asignar direcciones y coordinar el tráfico de datos.
2. **Router Zigbee**: Extiende la cobertura de la red, retransmitiendo datos entre nodos.
3. **End Device Zigbee**: Nodo final con funcionalidad reducida que comunica información al coordinador a través de un router.

La Figura 1.34 representa una red Zigbee típica utilizada en aplicaciones IoT. En el centro se encuentra el **coordinador Zigbee**, responsable de establecer la red, asignar direcciones y gestionar el tráfico. A su alrededor, los **routers Zigbee** amplían la cobertura mediante la retransmisión de datos hacia el coordinador, formando una red en malla robusta y resiliente. Finalmente, los **dispositivos finales (End Devices)** se conectan a los routers para enviar datos, aunque no retransmiten información. Esta organización permite el despliegue de redes escalables y eficientes, capaces de operar con miles de nodos distribuidos y con bajo consumo energético, características clave para muchas soluciones IoT.

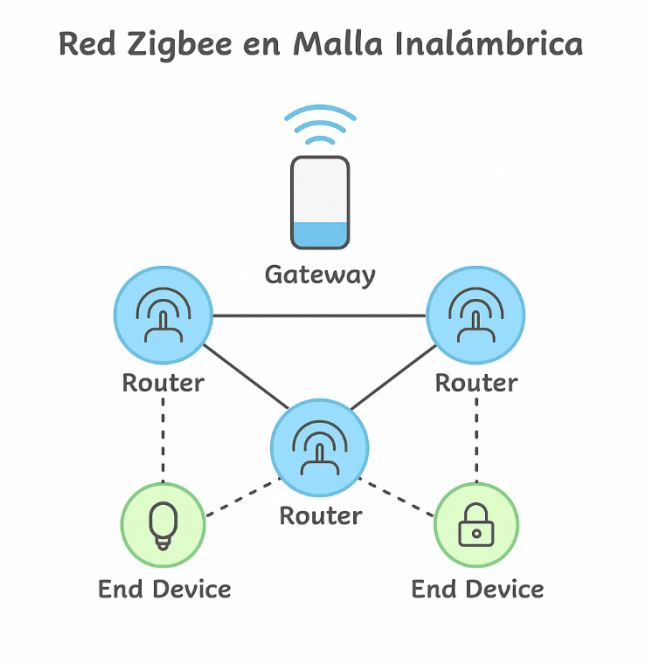


Figura 1.34 Estructura de una red Zigbee en topología malla. Elaboración propia.

Ventajas de Zigbee en IoT:

- **Consumo energético muy bajo**: Ideal para dispositivos alimentados por batería.
- **Alta escalabilidad**: Soporta grandes redes de nodos distribuidos.
- **Interoperabilidad**: Compatible con dispositivos certificados bajo el mismo perfil Zigbee.
- **Costos reducidos**: Hardware económico y de implementación sencilla.

Aplicaciones típicas:

- Sistemas de iluminación inteligente.
- Control climático y sensores ambientales.
- Alarmas y sensores de seguridad.
- Automatización industrial y control de procesos.

Gracias a su eficiencia energética, escalabilidad y confiabilidad en entornos con muchos nodos, Zigbee es uno de los estándares más consolidados y utilizados en redes IoT de baja potencia y corto alcance.

1.3.7 LoRa

LoRa (Long Range) es una tecnología de modulación de espectro ensanchado desarrollada por Semtech, especialmente diseñada para comunicaciones inalámbricas de largo alcance y bajo consumo energético. No es un protocolo en sí mismo, sino la capa física sobre la que se construyen redes LPWAN (Low Power Wide Area Networks), como LoRaWAN.

LoRa es ideal para redes de sensores distribuidos en áreas amplias, donde no se requiere alta velocidad de transmisión, pero sí gran autonomía energética y alcance geográfico. Se utiliza extensamente en sistemas de monitoreo remoto, ciudades inteligentes, agricultura de precisión, rastreo logístico y redes industriales.

Características técnicas de LoRa:

- **Modulación**: Chirp Spread Spectrum (CSS), robusta ante interferencias y ruido.
- **Rango de comunicación**: De 2 a 15 km en zonas urbanas y rurales.
- **Frecuencias**: 868 MHz (Europa), 915 MHz (América), 433 MHz (Asia), sin licencia.
- **Velocidad de datos**: De 0.3 kbps a 50 kbps.
- **Topología de red**: Estrella, con nodos sensores que se comunican con gateways.
- **Consumo energético**: Muy bajo, adecuado para dispositivos alimentados con baterías de varios años de duración.

Componentes de una red LoRa:

1. **Nodos LoRa**: Dispositivos sensores o actuadores que transmiten datos a los gateways.
2. **Gateway LoRa**: Recibe señales de múltiples nodos y las reenvía a través de Internet (usualmente usando LoRaWAN).
3. **Servidor de red**: Gestiona la autenticación, el enrutamiento y la agregación de datos.
4. **Aplicación final**: Plataforma que analiza y visualiza los datos recibidos (por ejemplo, Home Assistant, Node-RED, ThingsBoard).

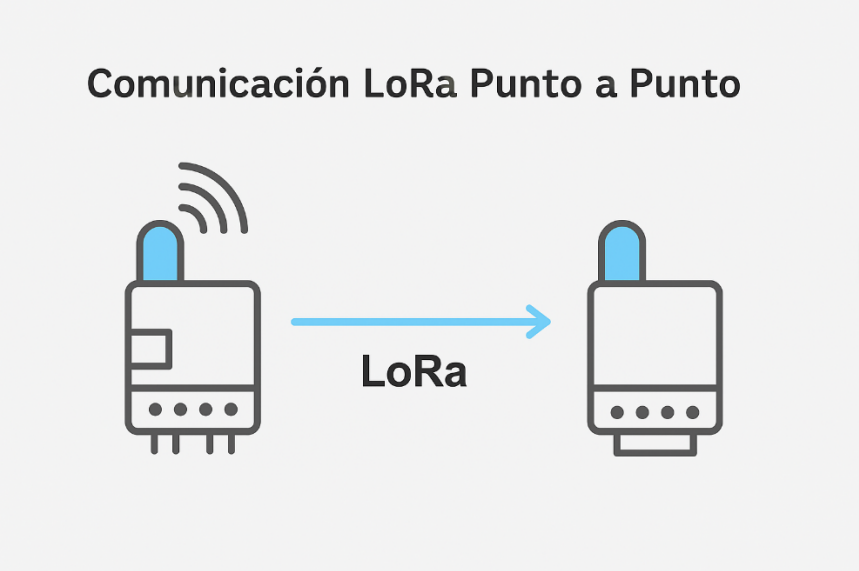


Figura 1.35 Comunicación punto a punto utilizando tecnología LoRa. Elaboración propia.

Ambas tecnologías LoRA y LoRaWAN se utilizan en las WSN actuales. LoRA, es una tecnología de modulación de capa física, utilizada principalmente en comunicaciones punto a punto (peer to peer). La Figura 1.35 representa un escenario simplificado en el que dos dispositivos equipados con módulos LoRa se comunican directamente sin necesidad de gateway ni servidor de red. Esta modalidad, conocida como LoRa P2P (Point-to-Point), es útil en aplicaciones donde se requiere una conexión directa entre nodos –por ejemplo, entre un sensor remoto y un recolector móvil– sin involucrar infraestructura adicional. Esta arquitectura es común en zonas rurales, sistemas de respaldo o despliegues experimentales, donde se valora la autonomía y simplicidad operativa.

En cambio, la tecnología LoRaWan, constituye un protocolo de red completo que se construye sobre LoRa, dicho protocolo define como se manejan los datos y como se comunican los dispositivos. En la Figura 1.36 muestra una red LoRaWAN compuesta por múltiples nodos sensores que transmiten datos a uno o más **gateways**. Estos gateways funcionan como puentes entre los dispositivos finales y el **servidor de red**, que se encarga de gestionar la autenticación, el enrutamiento y la seguridad de los datos. Posteriormente, la información se entrega al **servidor de aplicaciones**, donde es procesada y visualizada. Esta arquitectura de tipo estrella es altamente escalable y energéticamente eficiente, ideal para entornos de gran cobertura geográfica como agricultura, ciudades inteligentes y monitoreo ambiental distribuido.

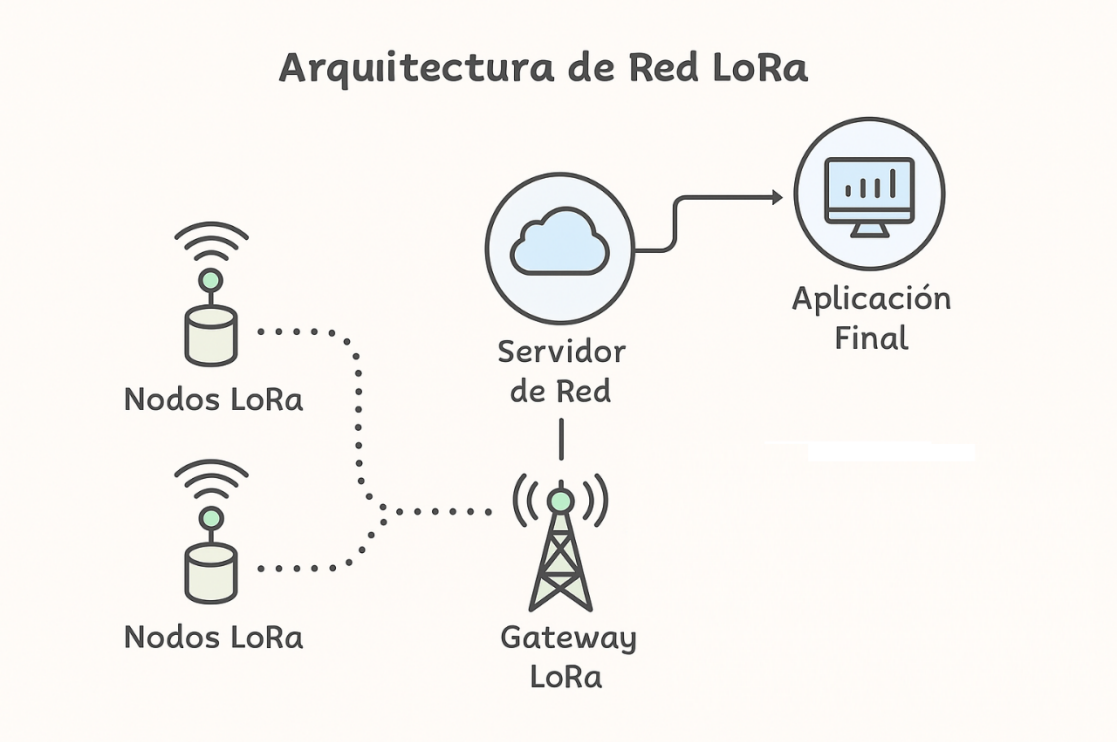


Figura 1.36 Arquitectura de red LoRaWAN en un entorno IoT. Elaboración propia.

Ventajas de LoRa en IoT:

- **Cobertura masiva**: Ideal para zonas rurales, agrícolas o de difícil acceso.
- **Autonomía prolongada**: Perfecta para sensores con mantenimiento limitado.
- **Escalabilidad**: Una sola puerta de enlace puede gestionar miles de dispositivos.
- **Seguridad**: Incluye cifrado AES de extremo a extremo en implementaciones LoRaWAN.

Aplicaciones comunes:

- Medición remota de consumo de agua y energía.
- Rastreo de activos y vehículos.
- Sensores agrícolas de humedad y clima.
- Redes de alerta temprana ante inundaciones o incendios.

La combinación de largo alcance, bajo consumo, simplicidad de infraestructura y escalabilidad hace de LoRa una de las tecnologías más potentes y versátiles para soluciones IoT distribuidas y autónomas.

1.3.8 Z-Wave

Z-Wave es un protocolo de comunicación inalámbrica especialmente desarrollado para aplicaciones de domótica y automatización del hogar. Fue diseñado por Sigma Designs (actualmente controlado por Silicon Labs) con el objetivo de ser eficiente, confiable y fácil de integrar. A diferencia de otros protocolos como Zigbee o Wi-Fi, Z-Wave opera en la banda de frecuencia **sub-GHz**, lo que le permite ofrecer una mejor penetración de señales en interiores y evitar interferencias con redes Wi-Fi.

Características técnicas:

- **Frecuencia**: 868 MHz (Europa), 908 MHz (EE.UU.), dependiendo del país.
- **Velocidad de transmisión**: entre 9.6 kbps y 100 kbps.
- **Alcance**: hasta 100 metros en espacios abiertos; 30–50 metros en interiores.
- **Topología**: red en malla con un nodo principal (controlador) y múltiples dispositivos secundarios.
- **Cantidad de nodos**: hasta 232 dispositivos por red.
- **Seguridad**: utiliza el esquema de cifrado S2 (Security 2) con AES-128 para asegurar las comunicaciones.

Componentes de una red Z-Wave:

1. **Controlador primario**: Nodo central que gestiona la red y configura los dispositivos (puede estar integrado en una central domótica).
2. **Dispositivos esclavos**: Actuadores (luces, cerraduras, interruptores) o sensores (presencia, temperatura, apertura) que siguen instrucciones del controlador.
3. **Repetidores**: Muchos dispositivos Z-Wave también actúan como repetidores para extender la cobertura de la red mediante enrutamiento dinámico.

La Figura 1.37 ilustra una red Z-Wave típica implementada en un entorno doméstico. En el centro se encuentra el **controlador primario**, como una central domótica o hub inteligente, que gestiona la configuración y comunicación de los dispositivos conectados. Alrededor de él se conectan múltiples **dispositivos esclavos** como sensores de movimiento, interruptores, cerraduras inteligentes y termostatos, organizados en una red **en malla**. Esta estructura permite que los dispositivos se comuniquen entre sí y retransmitan

mensajes para extender el alcance, garantizando mayor cobertura y confiabilidad dentro del hogar.

Figura 1.37 Arquitectura de una red Z-Wave para automatización del hogar. Elaboración propia.

Ventajas de Z-Wave en IoT:

- **Alta interoperabilidad**: Todos los dispositivos certificados por Z-Wave Alliance son compatibles entre sí, sin importar el fabricante.
- **Red en malla robusta**: Mejora la confiabilidad al permitir rutas redundantes.
- **Bajo consumo energético**: Ideal para sensores con batería.
- **Ideal para interiores**: Excelente penetración de señal en paredes y estructuras.

Aplicaciones comunes:

- Control de iluminación inteligente.
- Automatización de persianas, enchufes y termostatos.

- Cerraduras electrónicas y sistemas de seguridad.
- Integración con asistentes de voz y hubs domóticos como SmartThings, Fibaro o Vera.

Z-Wave continúa siendo una de las tecnologías más utilizadas en soluciones domésticas por su confiabilidad, facilidad de instalación y amplio ecosistema de dispositivos interoperables.

1.3.9 Seguridad y Privacidad en Sistemas IoT

Dado que IoT maneja grandes volúmenes de datos sensibles, es fundamental implementar estrategias de seguridad:

- Cifrado de datos: Uso de TLS/SSL para proteger la información transmitida.
- Autenticación robusta de dispositivos: Mecanismos de verificación para evitar accesos no autorizados con certificados digitales.
- Gestión de vulnerabilidades: Actualización y mantenimiento de firmware para prevenir ataques.

Si bien la seguridad en IoT ofrece múltiples ventajas, también presenta desafíos significativos. En la Figura 1.38 se comparan los pros y contras de implementar medidas de seguridad en entornos IoT. Entre los beneficios destacan la protección de datos, la privacidad del usuario y la prevención de ataques. No obstante, los costos elevados, la complejidad en la implementación y la necesidad de actualizaciones constantes son desafíos que deben abordarse para garantizar un equilibrio entre seguridad y funcionalidad.

Figura 1.38 Comparación entre los beneficios y desafíos de la seguridad en IoT. Elaboración propia.

Elementos de Cierre

Resumen del Capítulo

Este capítulo exploró los fundamentos de IoT, su evolución tecnológica, su impacto transversal en múltiples sectores, las arquitecturas típicas de sistemas IoT, los principales componentes involucrados, tecnologías de comunicación más relevantes y los retos de seguridad y privacidad. Una vez, que conocemos la arquitectura IoT de forma global y los elementos claves que aparecen en la capa de percepción o de dispositivos, la estructura del libro nos llevará a las diferentes capas de dicha estructura, es por ello, que el próximo capítulo abordaremos la capa de comunicaciones que nos permitirá enlazar a los dispositivos con la capa superior de aplicaciones que veremos más adelante en el capítulo 3.

Preguntas de Revisión

❓ Opción múltiple:
¿Cuál de las siguientes tecnologías es ideal para comunicación de largo alcance y bajo consumo en IoT?
a) Wi-Fi
b) Bluetooth
c) LoRaWAN
d) Zigbee

✅ Verdadero/Falso:
La capa de percepción en IoT está encargada exclusivamente del procesamiento de datos.
(Verdadero / Falso)

✍️ Respuesta corta:
Menciona dos factores clave que favorecieron la expansión del IoT en los últimos años.

Clave de Respuestas

- ❓ Opción múltiple: c) LoRaWAN
- ✅ Verdadero/Falso: Falso (la percepción es para capturar datos, no procesarlos)
- ✍️ Respuesta corta: Miniaturización de sensores, expansión de conectividad inalámbrica.

Referencias bibliográficas

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials, 17*(4), 2347–2376. https://doi.org/10.1109/COMST.2015.2444095*

Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. Computer Networks, 54*(15), 2787–2805. https://doi.org/10.1016/j.comnet.2010.05.010*

Bluetooth SIG. (2025). Specifications and documents. Bluetooth. *Recuperado el 10 de mayo de 2025, de https://www.bluetooth.com/specifications/specs/*

El Hakim, A. (2018). Internet of Things (IoT) system architecture and technologies [White paper]. ResearchGate. https://www.researchgate.net/publication/323525875_Internet_of_Things_IoT_System_Architecture_and_Technologies_White_Pape

Hernández-Rojas, D. L., Fernández-Caramés, T. M., Fraga-Lamas, P., & Escudero, C. J. (2018). Design and practical evaluation of a family of lightweight protocols for heterogeneous sensing through BLE beacons in IoT telemetry applications. Sensors, 18*(1), 57. https://doi.org/10.3390/s18010057*

Information on RFC 7228 » RFC Editor. (2025, 10 de mayo). https://www.rfc-editor.org/info/rfc7228

Kumar, N. M., & Mallick, P. K. (2018). The Internet of Things: Insights into the building blocks, component interactions, and architecture layers. Procedia Computer Science, 132*, 109–117. https://doi.org/10.1016/j.procs.2018.05.170*

Mocnej, J., Pekar, A., Seah, W. K. G., & Zolotova, I. (2018). Network traffic characteristics of the IoT application use cases *(Technical Report ECSTR18-01). School of Engineering and Computer Science, Victoria University of Wellington. https://ecs.wgtn.ac.nz/Main/TechnicalReportSeries*

oneM2M. (2025, 10 de mayo). Standards for M2M and the Internet of Things. *https://www.onem2m.org/*

Rueda, J. S., & Talavera Portocarrero, J. M. (2017). Similitudes y diferencias entre Redes de Sensores Inalámbricas e Internet de las Cosas: Hacia una postura clarificadora. Revista Colombiana De Computación, 18*(2), 58–74. https://doi.org/10.29375/25392115.3218*

Sinha, S. (2024, 3 de septiembre). State of IoT 2024: Number of connected IoT devices growing 13% to 18.8 billion globally. IoT Analytics. https://iot-analytics.com/number-connected-iot-devices/

2 Capítulo 2: Comunicación en IoT

Elementos de Apertura

Objetivos de aprendizaje:

- Caracterizar los protocolos de comunicación más relevantes para IoT.
- Analizar la estructura y el funcionamiento de REST-API y MQTT.
- Interpretar la importancia del procesamiento de datos en arquitecturas IoT modernas.

Competencias:

- Identificar los principales protocolos de comunicación en entornos IoT.
- Aplicar conceptos de Edge Computing y Fog Computing en soluciones IoT.
- Relacionar el procesamiento de datos con el diseño eficiente de sistemas conectados.

Preguntas de enfoque:

- ¿Por qué son importantes los protocolos ligeros en IoT?
- ¿Qué diferencia existe entre Edge Computing y Fog Computing?
- ¿Cómo se integra la Inteligencia Artificial con el procesamiento de datos en IoT?

La Figura 2.1 muestra de forma resumida las interrelaciones entre protocolos, arquitecturas de procesamiento y evolución hacia sistemas inteligentes en IoT.

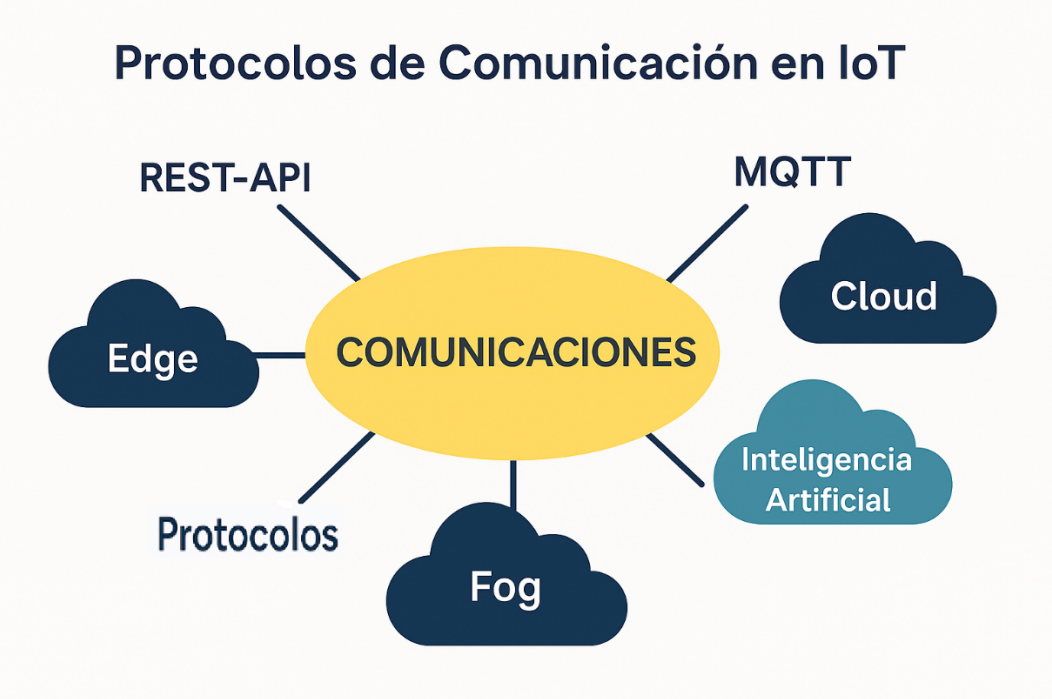


Figura 2.1 Protocolos de comunicación y procesamiento en IoT. Elaboración propia.

2.1 Protocolo REST-API en IoT

Los protocolos de comunicación en IoT juegan un papel fundamental en la transferencia de datos entre dispositivos, servidores y aplicaciones. Dado que los sistemas IoT suelen estar compuestos por sensores, actuadores y plataformas de análisis de datos, es esencial contar con protocolos eficientes y escalables que faciliten la interoperabilidad y la integración de servicios.

Uno de los enfoques más utilizados en la comunicación de IoT es el uso de REST-API y HTTP, los cuales permiten la transmisión de datos estructurados entre dispositivos IoT y aplicaciones web. Estos protocolos facilitan la integración con servicios en la nube, plataformas de análisis y aplicaciones móviles, asegurando la interoperabilidad entre diferentes sistemas.

2.1.1 Protocolo REST-API y HTTP en IoT

REST-API en IoT

Las APIs REST (Representational State Transfer) se utilizan ampliamente en la comunicación IoT debido a su simplicidad y compatibilidad con múltiples plataformas. REST permite que los dispositivos IoT intercambien datos

utilizando métodos estándar de HTTP, como GET, POST, PUT y DELETE, facilitando la manipulación de datos en la nube y en servidores.

Ejemplo: Un sensor de temperatura que envía datos a un servidor remoto mediante una petición POST en una API REST, permitiendo su visualización en una aplicación web.

HTTP como Protocolo de Comunicación en IoT

El Protocolo de Transferencia de Hipertexto (HTTP) es ampliamente utilizado en sistemas IoT para la comunicación entre dispositivos y servidores. Aunque no está diseñado específicamente para aplicaciones IoT de baja latencia, es útil en arquitecturas que requieren integración con servicios web y almacenamiento en la nube.

Ventajas del uso de HTTP en IoT:

- Facilidad de integración con aplicaciones web y móviles.
- Compatibilidad con arquitecturas basadas en la nube.
- Uso de estándares abiertos que facilitan la interoperabilidad.

Sin embargo, HTTP no es la mejor opción para aplicaciones en tiempo real o con restricciones de energía, ya que implica una sobrecarga mayor en comparación con otros protocolos como MQTT o CoAP.

2.1.2 Integración de REST-API y HTTP en IoT

La integración de REST-API y HTTP en IoT permite la comunicación eficiente entre dispositivos y plataformas, facilitando la recopilación y análisis de datos. Esta integración es clave en la arquitectura IoT moderna, donde los datos de sensores se envían a servicios en la nube para su procesamiento y toma de decisiones automatizada, como se puede apreciar en la Figura 2.2.

El modelo REST (**Request/Response)** es una técnica de comunicación donde un cliente (solicitante) envía una petición (**request**) a un servidor (proveedor de servicio) y espera una respuesta (**response**). Este modelo se basa en la comunicación síncrona y se utiliza en protocolos de aplicación como:

- **HTTP (Hypertext Transfer Protocol)**
- **CoAP (Constrained Application Protocol)**, versión ligera de HTTP para dispositivos IoT, Shelby, Z., & Bormann, C. (2019).

En este modelo:

1. **El cliente envía una solicitud** (request) con una acción específica (ej., obtener datos de un sensor).
2. **El servidor recibe la solicitud**, la procesa y ejecuta la acción requerida.
3. **El servidor responde** (response) con los datos solicitados o con un mensaje de confirmación.

Figura 2.2 Representación de la comunicación IoT utilizando REST-API y HTTP, destacando su papel en la integración de servicios. Elaboración propia.

Las APIs RESTful siguen estos principios:

1. **Cliente-Servidor (la más popular)**
   - Separa la lógica de negocio del cliente y del servidor, permitiendo escalabilidad y modularidad.
2. **Interfaz Uniforme**
   - Define un conjunto estándar de métodos HTTP: **GET, POST, PUT, DELETE**.
3. **Sin Estado (Stateless)**
   - Cada solicitud es independiente; el servidor no almacena el estado de la sesión del cliente.

4. **Cacheable**
    - Las respuestas pueden almacenarse en caché para mejorar el rendimiento.
5. **Uso de Representaciones Estándar**
    - Los datos se intercambian en formatos como **JSON o XML**.
6. **Arquitectura Basada en Recursos**
    - Cada entidad o dispositivo en IoT es tratado como un **recurso** con una URL única.

Estructura de un mensaje HTTP.

Cuando un cliente realiza una solicitud HTTP a una RESTful API, el servidor responde con un mensaje estructurado que contiene información clave. Este mensaje HTTP de respuesta se compone de tres partes principales:

1. **Status Line** (Línea de Estado)
2. **Headers** (Encabezados)
3. **Body** (Cuerpo de la respuesta)

A continuación, se explica en detalle cada una de estas partes y su importancia en la comunicación de APIs RESTful en el contexto de IoT.

Status Line (Línea de Estado)

La línea de estado es la primera línea de la respuesta HTTP y contiene tres elementos:

1. Versión del protocolo HTTP (Ejemplo: *HTTP/1.1* o *HTTP/2*)
2. Código de estado HTTP (Ejemplo: *200 OK*, *404 Not Found*)
3. Mensaje descriptivo del estado (Ejemplo: *OK*, *Not Found*)

Ejemplo:

HTTP/1.1 200 OK

- HTTP/1.1: Versión del protocolo.
- 200: Código de estado.
- OK: Mensaje que indica éxito

Headers (Encabezados HTTP)

Los **encabezados HTTP** proporcionan información adicional sobre la respuesta del servidor. Estos encabezados son clave para la correcta interpretación de los datos por parte del cliente IoT.

Tipos de Encabezados en una Respuesta HTTP

1. Encabezados Generales → Información sobre la comunicación.
2. Encabezados de Respuesta → Información sobre el servidor y los datos.
3. Encabezados de Entidad → Información sobre el contenido del cuerpo de la respuesta.

Ejemplo:

HTTP/1.1 200 OK

Date: Mon, 19 Feb 2025 14:30:00 GMT

Server: Apache/2.4.41 (Ubuntu)

Content-Type: application/json

Content-Length: 85

Connection: keep-alive

- Date: Indica la fecha y hora en que el servidor envió la respuesta.
- Server: Informa sobre el tipo y versión del servidor.
- Content-Type: Especifica el formato del cuerpo de la respuesta (ej., application/json, text/html).
- Content-Length: Indica el tamaño del cuerpo de la respuesta en bytes.
- Connection: Especifica si la conexión debe mantenerse abierta (keep-alive) o cerrarse (close).

Body (Cuerpo de la Respuesta)

El **cuerpo de la respuesta** contiene los datos solicitados en un formato estructurado, como **JSON o XML**. En IoT, estos datos pueden ser mediciones de sensores, configuraciones de dispositivos o respuestas de comandos.

Ejemplo:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 85
{
   "sensor_id": "001",
   "temperature": "25.6",
   "unit": "Celsius",
   "timestamp": "2025-02-19T14:30:00Z"
}
```

Códigos de Estado HTTP en RESTful APIs para IoT

Los códigos de estado HTTP indican el resultado de la solicitud. Se dividen en cinco categorías:

Códigos 1xx - Respuestas Informativas

Códigos 2xx - Respuestas Exitosas

Códigos 3xx – Redirecciones

Códigos 4xx - Errores del Cliente

Códigos 5xx - Errores del Servidor

Ejemplos:

100 : Continue El servidor ha recibido la solicitud inicial y el cliente puede continuar enviando el resto. Para confirmaciones en conexiones persistentes.

102 : Processing La solicitud está en proceso, pero aún no se ha completado. Para operaciones que requieren mucho tiempo.

200 : OK Respuesta exitosa con datos. Obtener datos de un sensor.

201 : Created Un nuevo recurso ha sido creado. Registrar un nuevo dispositivo IoT.

204 : No Content La solicitud fue exitosa, pero no hay contenido en la respuesta. Apagar un dispositivo sin devolver datos.

301 : Moved Permanently El recurso ha cambiado de ubicación permanentemente. Reubicación de un servidor IoT.

304 : Not Modified El recurso no ha cambiado desde la última solicitud. Evita descargas innecesarias de datos en caché.

400 : Bad Request La solicitud es inválida o tiene errores. Formato incorrecto en una API IoT.

401 : Unauthorized Se requiere autenticación. Acceder a un dispositivo sin credenciales.

403 : Forbidden Acceso denegado al recurso. Intento de modificar un sensor sin permisos.

404 : Not Found El recurso solicitado no existe. Buscar un sensor inexistente en una API.

429 : Too Many Requests Demasiadas solicitudes en poco tiempo. API IoT bloquea consultas excesivas.

500 : Internal Server Error Error general del servidor. Falla en la API del servidor IoT.

502 : Bad Gateway Un servidor intermedio recibió una respuesta inválida. Problema con balanceadores de carga.

503 : Service Unavailable El servicio está temporalmente fuera de línea. API IoT en mantenimiento.

504 : Gateway Timeout Tiempo de espera excedido. Sensores IoT no responden a la API.

En los siguientes epígrafes, se explorarán casos de uso específicos y ejemplos prácticos de cómo implementar REST-API y HTTP en entornos IoT, optimizando la comunicación y la interoperabilidad de los dispositivos conectados.

2.1.3 Métodos HTTP y su Aplicación en IoT

Los métodos HTTP que aparecen en la Figura 2.3 desempeñan un rol esencial en la comunicación entre dispositivos IoT y servidores, permitiendo la gestión eficiente de recursos y datos en la nube.

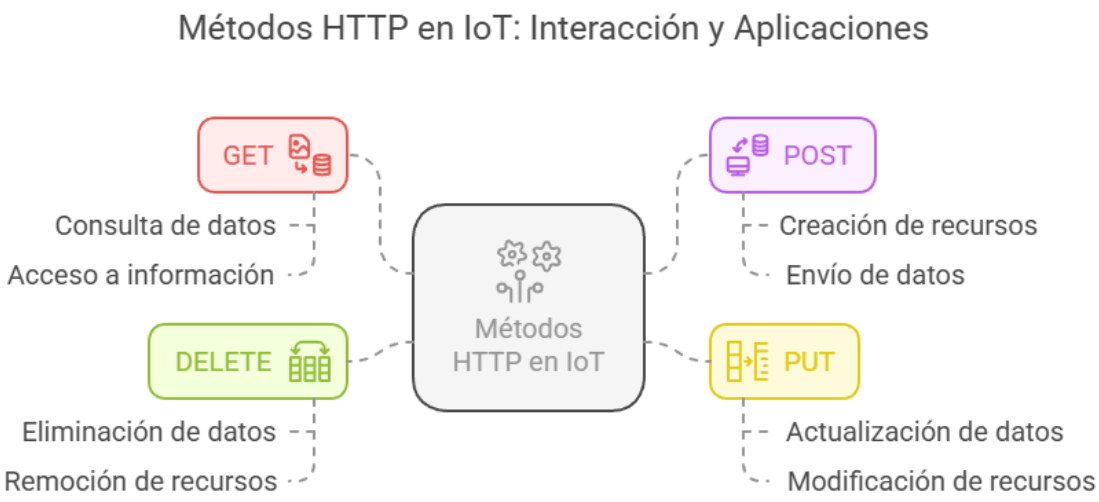


Figura 2.3 Métodos HTTP en IoT, mostrando su uso en la interacción y gestión de recursos entre dispositivos y servidores. Elaboración propia.

Los principales métodos HTTP utilizados en IoT son:

- GET: Se utiliza para solicitar datos de un servidor. En IoT, este método es común en aplicaciones donde un sensor consulta información sobre el estado de un sistema.
- POST: Se emplea para enviar datos a un servidor, permitiendo la creación de nuevos recursos. Por ejemplo, un dispositivo IoT puede enviar mediciones periódicas de temperatura a un servidor en la nube.
- PUT: Se usa para actualizar datos existentes en un servidor. En IoT, esto es útil para modificar configuraciones de un dispositivo remoto.
- DELETE: Permite la eliminación de recursos almacenados en un servidor. Por ejemplo, un usuario podría eliminar un dispositivo registrado en una plataforma IoT.

Estos métodos facilitan la interacción entre los dispositivos IoT y las plataformas de administración de datos, asegurando un flujo de información eficiente y seguro.

2.1.4 Implementación de Servicios REST en IoT

La implementación de servicios REST en IoT sigue un flujo estructurado para garantizar la correcta comunicación entre dispositivos y servidores. Este proceso consta de cinco pasos fundamentales mostrados en la Figura 2.4.

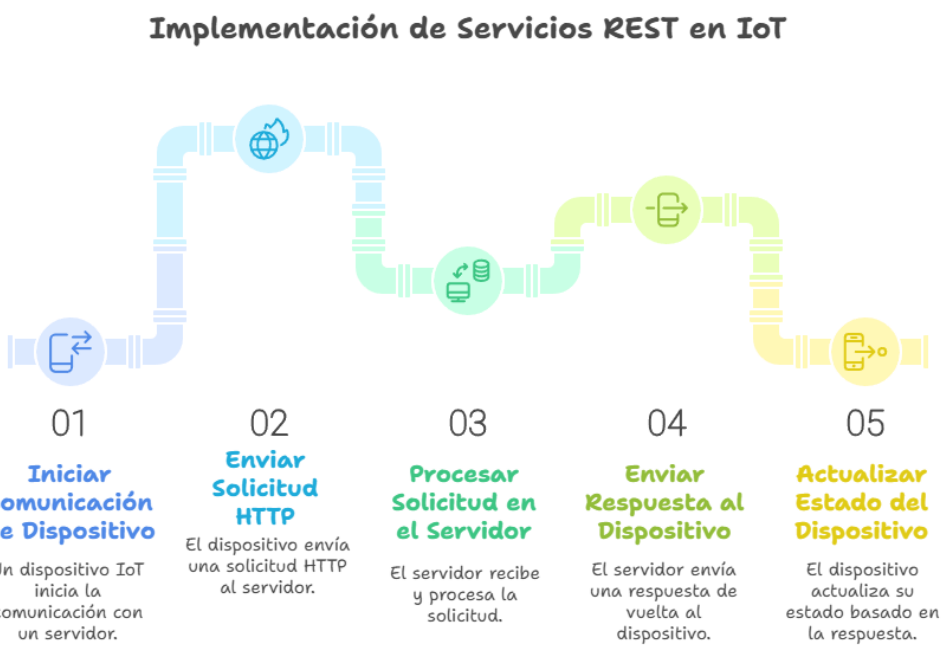


Figura 2.4 Flujo de implementación de servicios REST en IoT, ilustrando los cinco pasos clave para la comunicación efectiva entre dispositivos y servidores. Elaboración propia.

Este modelo de comunicación permite que los dispositivos IoT operen de manera eficiente dentro de una arquitectura basada en la nube, asegurando la interoperabilidad y escalabilidad del sistema.

De forma resumida, se requiere:

- Crear un endpoint en el servidor.
- Configurar el dispositivo IoT como cliente.
- Enviar solicitudes con formato JSON o XML.

Ejemplos de uso de RESTful API en IoT. Supongamos que tenemos una **red de sensores de temperatura** en una fábrica y queremos gestionar los datos mediante una API RESTful.

a) Obtener la lista de sensores disponibles (GET)

GET /sensors HTTP/1.1

Host: api.iotfactory.com

Respuesta:

```
[
  {"id": "001", "location": "Sala 1"},
  {"id": "002", "location": "Sala 2"}
]
```

b) Obtener la temperatura de un sensor específico (GET)

```
GET /sensors/001/temperature HTTP/1.1
```

Respuesta:

```
{
  "sensor_id": "001",
  "temperature": "22.5",
  "unit": "Celsius"
}
```

c) Actualizar la configuración de un sensor (PUT)

```
PUT /sensors/001/config HTTP/1.1
Content-Type: application/json
{
  "sampling_rate": "30s",
  "threshold": "28"
}
```

Respuesta:

```
HTTP/1.1 200 OK
{
  "message": "Configuration updated successfully"
}
```

d) Registrar un nuevo sensor (POST)

```
POST /sensors HTTP/1.1
Content-Type: application/json
```

{

"id": "003",

"location": "Sala 3"

}

Respuesta:

HTTP/1.1 201 Created

e) Eliminar un sensor (DELETE)

DELETE /sensors/003 HTTP/1.1

Respuesta:

HTTP/1.1 200 OK

{

"message": "Sensor deleted successfully"

}

Parámetros y Query Strings en RESTful APIs para IoT

En las **RESTful APIs**, los clientes pueden enviar datos al servidor a través de diferentes mecanismos. Los dos métodos más comunes para proporcionar información en una solicitud son:

- Path Parameters (Parámetros en la URL)
- Query Parameters (Parámetros en la cadena de consulta)

Ambos métodos se utilizan en aplicaciones IoT para filtrar, identificar y modificar recursos, pero tienen diferencias clave en su propósito y uso.

Path Parameters (Parámetros en la URL)

Los **path parameters** son parte de la estructura de la URL y se utilizan para identificar un recurso específico. Se definen dentro de la ruta (path) y se representan mediante {} en la documentación de la API.

Ejemplo de Uso en IoT

Imaginemos un sistema IoT donde queremos acceder a la temperatura de un sensor específico.

GET /sensors/{sensor_id}/temperatura

Si queremos obtener la temperatura del sensor con ID **"123"**, la URL sería:

GET /sensors/123/temperature

Respuesta:

```
{
   "sensor_id": "123",
   "temperature": "25.6",
   "unit": "Celsius",
   "timestamp": "2025-02-19T14:30:00Z"
}
```

Query Parameters (Parámetros en la Cadena de Consulta)

Los **query parameters** son parámetros opcionales que se agregan a la URL después del signo de interrogación (?). Se utilizan para **filtrar, ordenar o personalizar** los resultados.

Cada parámetro tiene una clave y un valor, separados por el signo =. Si hay varios parámetros, se separan con &.

Ejemplo de Uso en IoT

Supongamos que queremos obtener la lista de sensores en un área específica y solo los activos.

Solicitud con Query Parameters:

GET /sensors?location=warehouse&status=active

Respuesta:

```
[
   {"sensor_id": "001", "location": "warehouse", "status": "active"},
   {"sensor_id": "002", "location": "warehouse", "status": "active"}
]
```

Ejemplo de POST con Query Parameters

Se usa cuando se requiere **enviar datos que modifican el comportamiento de un recurso existente**.

Ejemplo: Enviar datos de un sensor IoT sin usar un cuerpo JSON

Solicitud:

POST /sensors/upload?sensor_id=005&temperature=26.3&humidity=55 HTTP/1.1 Host: api.iotplatform.com

Respuesta (*200 OK*):

{

"message": "Data received successfully",

"sensor_id": "005",

"temperature": "26.3",

"humidity": "55"

}

2.2 Protocolo MQTT en IoT

2.2.1 Funcionamiento del Protocolo MQTT

MQTT opera bajo un modelo de publicación-suscripción (Pub/Sub), en el que los dispositivos IoT publican mensajes en temas específicos y otros dispositivos suscritos reciben dichos mensajes a través de un broker MQTT, Banks, A., & Gupta, R. (2019).

Los pasos principales en la comunicación MQTT mostrados en la Figura 2.5 son:

1. Conexión a la Red: Un dispositivo IoT se conecta a la red y establece comunicación con un broker MQTT.
2. Broker MQTT: Actúa como intermediario y gestiona la distribución de mensajes entre dispositivos.
3. Distribución de Mensajes: Los mensajes publicados por un dispositivo se envían a todos los dispositivos suscritos al mismo tema.
4. Otros Dispositivos IoT: Reciben la información transmitida y ejecutan acciones en función del mensaje recibido.

Este modelo reduce la carga en los dispositivos finales y optimiza el uso del ancho de banda, permitiendo comunicaciones en tiempo real con alta eficiencia.

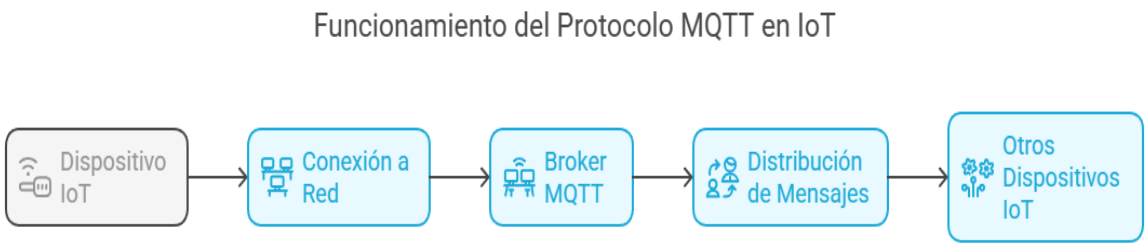


Figura 2.5 Esquema del funcionamiento del protocolo MQTT en IoT, mostrando su arquitectura basada en el modelo publicación-suscripción. Elaboración propia.

2.2.2 Modelo de Publicación/Suscripción en MQTT

El modelo publicación/suscripción en MQTT permite una comunicación eficiente entre dispositivos al eliminar la necesidad de conexiones directas entre ellos. En este modelo, los dispositivos que envían información (publicadores) no interactúan directamente con los receptores (suscriptores), sino que envían mensajes a un broker MQTT, que se encarga de distribuir los mensajes a los dispositivos suscritos a los temas correspondientes.

En este esquema, se incluyen conceptos clave como:

- Publicador: Dispositivo que envía mensajes a un tema específico. (pueden existir concurrentemente ilimitados dispositivos publicando a la vez)
- Broker MQTT: Servidor intermediario que recibe los mensajes y los distribuye a los suscriptores. (servidor de mensajería, donde se crean y se gestionan los tópicos que son cadenas de texto con algunos caracteres especiales, sobre los cuales los publicadores escriben y son escuchados por todos aquellos dispositivos subscritos a dicho tópico)
- Suscriptor: Dispositivo que recibe mensajes al estar suscrito a un tema en particular. (pueden existir concurrentemente ilimitados dispositivos escuchando a la vez, incluso a un mismo tópico)

- Niveles de QoS (Quality of Service): Determinan la confiabilidad en la entrega de mensajes, con opciones como:
    - QoS 0: Entrega sin garantía (puede perderse el mensaje).
    - QoS 1: Garantiza que el mensaje se entregue al menos una vez.

- QoS 2: Asegura que el mensaje se entregue exactamente una vez, con confirmaciones adicionales.

Esta arquitectura permite mayor escalabilidad y eficiencia en sistemas IoT, donde múltiples dispositivos pueden recibir información en tiempo real sin sobrecargar la red, ver Figura 2.6.

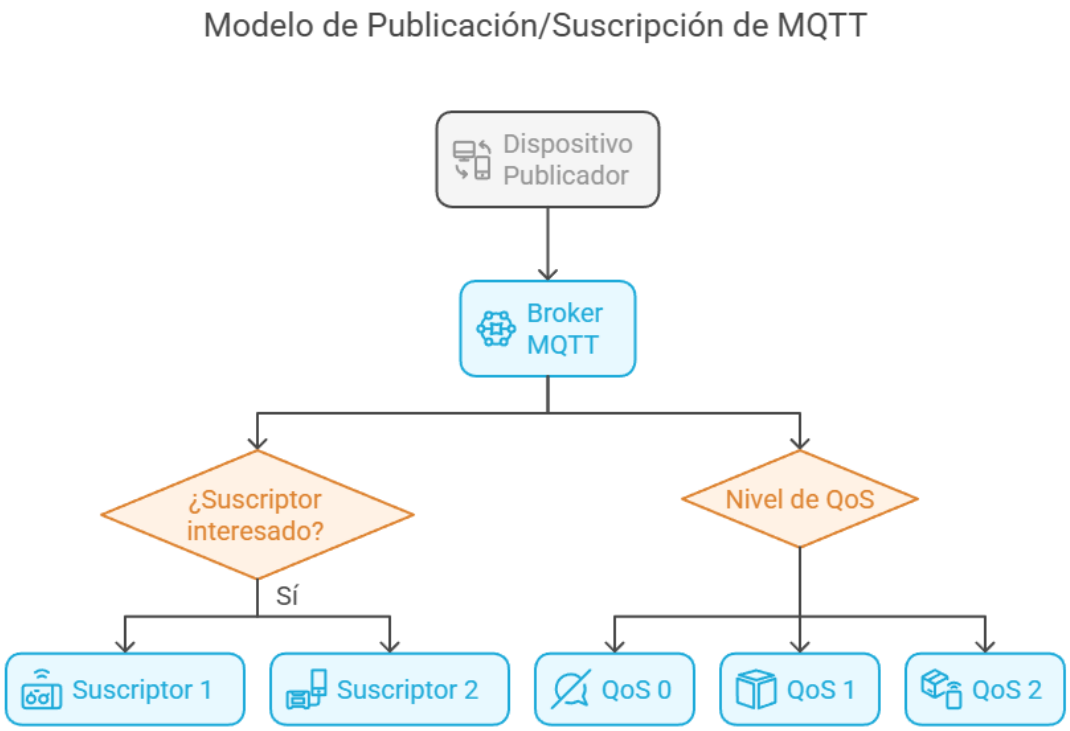


Figura 2.6 Modelo de publicación/suscripción en MQTT, detallando el flujo de mensajes entre publicadores, broker y suscriptores, junto con la gestión de niveles de QoS. Elaboración propia.

Brokers y clientes

Existen varios brokers, algunos de ellos corren en la cloud y otros pueden ser instalados en hardware de cómputo, dígase una PC, un smartphone o un Raspberry pi. De igual forma existen soluciones de clientes para diferentes plataformas de hardware como se aprecia en la siguiente Figura 2.7.

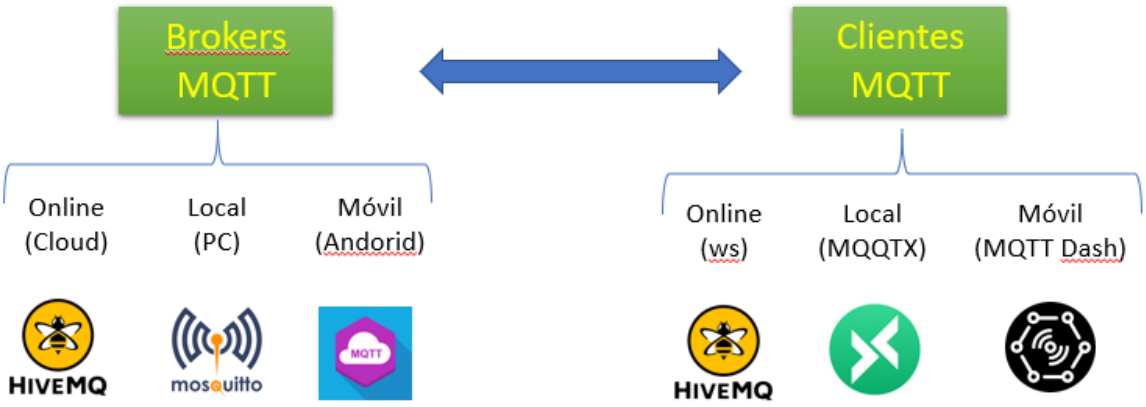


Figura 2.7 Brokers y clientes MQTT corriendo en diferentes plataformas (cloud, PC local, Móvil). Elaboración propia.

El bróker Mosquitto, destaca entre las soluciones "gratis" y ha sido ampliamente usado, tanto en la academia como en la industria y puede ser instalado en un PC, en una Rpi y en la cloud. Típicamente permite conexiones TCP por el puerto 1883 y con niveles de seguridad por el puerto 8883.

En Windows se puede instalar Mosquitto como un servicio, como se puede apreciar en la siguiente Figura 2.8:

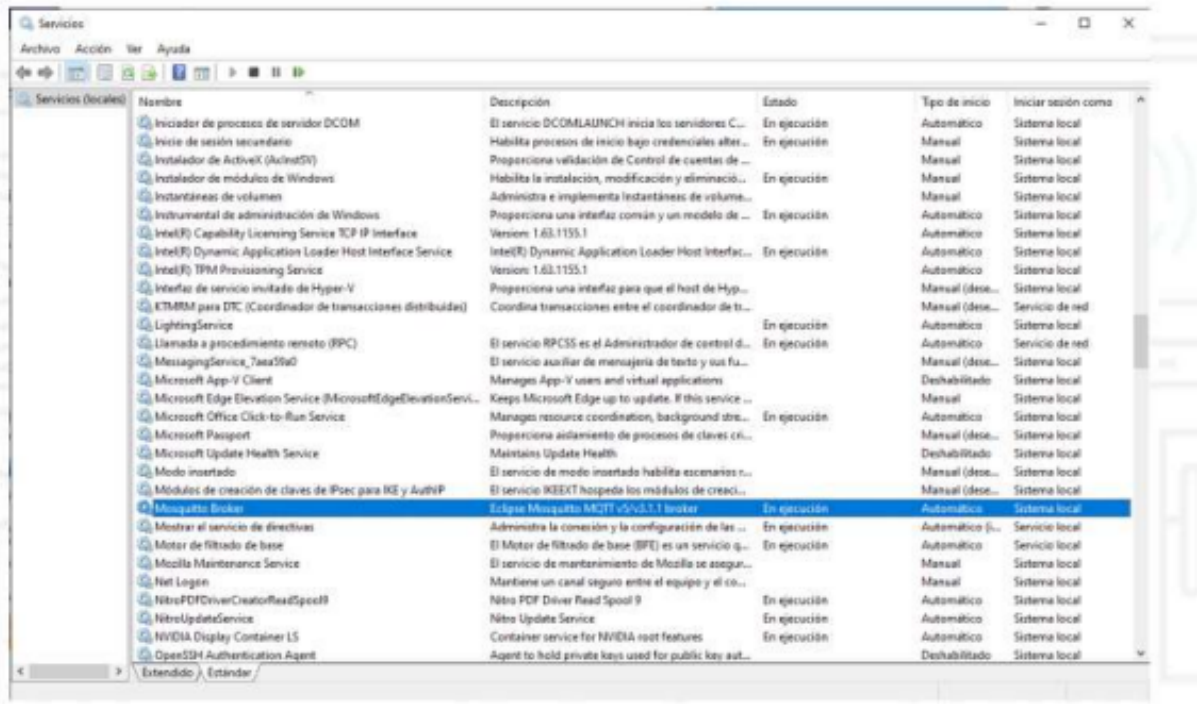


Figura 2.8 Broker Mosquitto, corriendo como un servicio sobre Windows. Elaboración propia.

Otro bróker destacado en la industria IoT es HiveMQ, ver Figura 2.9, que brinda un bróker MQTT **público** en la nube, suficiente para hacer pruebas de contexto de forma gratuita y versiones de paga razonables. El cliente gratuito de HiveMQ tambien disponible en la nube solo permite conexiones por websocket en el puerto 8000.

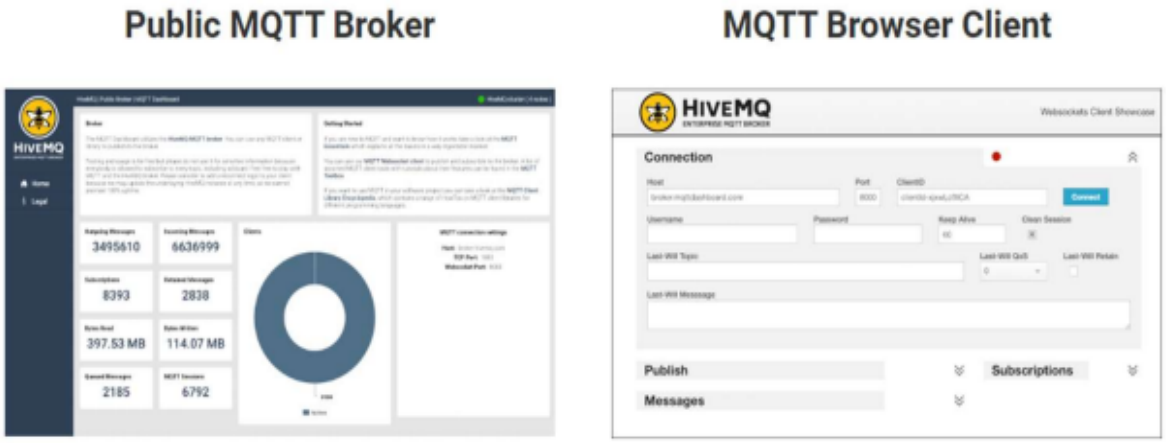


Figura 2.9 Broker y cliente de HiveMQ, corriendo como en la cloud, tomada de HiveMQ (2025, 10 de mayo).

Entre los clientes que podemos instalar en el PC, se encuentra MQTTX, ver Figura 2.10, el cual permite hacer conexiones tcp y ws hacia diferentes brokers

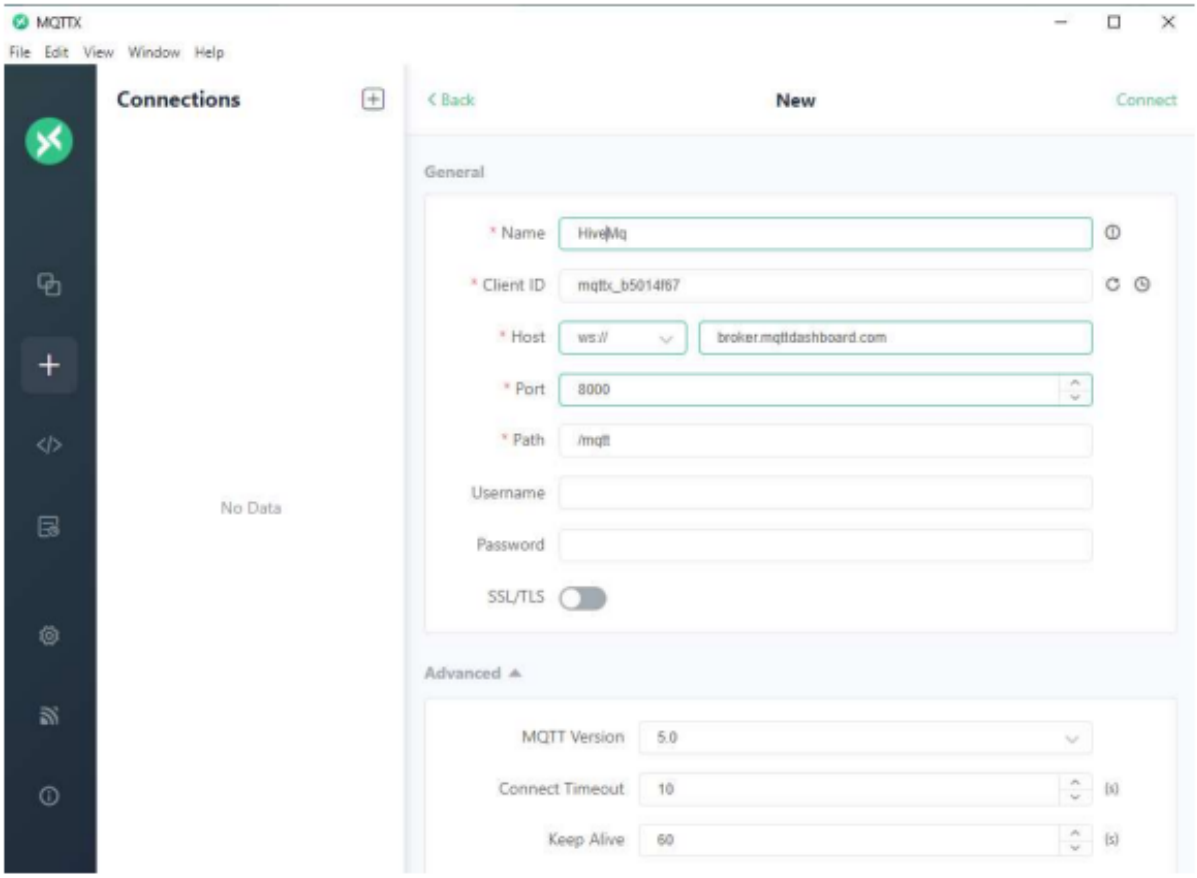


Figura 2.10 Cliente MQTTX, corriendo de forma local sobre una PC con Windows, adaptada de MQTTX (2025, 10 de mayo).

Tópicos

Este concepto es extremadamente importante en el protocolo MQTT. Los tópicos son un conjunto de caracteres, se recomienda usar palabras representativas de la solución IoT. Las palabras o campos se delimitan por un "/" (slash). Diferencia entre mayúsculas y minúsculas (Case sensitive), usa UTF-8 strings y debe existir al menos 1 carácter. Los tópicos se crean en el bróker y se pueden crear tantos tópicos como sean necesario. A un mismo tópico se pueden subscribir ilimitados subscriptores, por tanto, todos ellos escucharían los mensajes que sean publicados en dicho tópico.

Ejemplo:

micasa/piso1/cuarto1/foco1

Dentro de los tópicos se pueden usar caracteres especiales como el '+' : permite hacer un match y el '#'. Escucha todos los mensajes en tópicos cuyos caracteres coincidan hasta el nivel donde aparece el '+'que coincidan. En

cambio, se utiliza '#' cuando se desea escuchar todos los mensajes que coincidan con la raíz.

Ejemplos:

Se disponen de los siguientes tópicos posibles:

Casa/cuarto1/foco1

Casa/cuarto1/alarma

Casa/garage/foco1

Casa/cuarto2

Si un dispositivo se subscribe al tópico: Casa/+/foco1, entonces solo podría escuchar los siguientes tópicos:

Casa/cuarto1/foco1

Casa/garage/foco1

En cambio, si se desean escuchar todos los tópicos, debería usar el tópic**o**:

Casa/#

Existen **más** funcionalidades y conceptos involucrados con MQTT, pero el objetivo de este libro, es dar una pincelada de los principales conceptos sobre IoT, que permita al lector que recién comienza, crear una solución IoT de forma rápida. Para un estudio **más** riguro**s**o con necesidades conexiones **más** complejas se recomienda hacer una búsqueda en la web.

Es importante destacar que MQTT, como sus siglas indican fue pensado para mensajes cortos, muy útil para comandos y datos usados dentro de IoT. Por ejemplo, no se recomienda enviar Video con este protocolo o mensajes muy extensos que implicaría un costo de recursos innecesarios en la gestion de los diferentes "chucks" que tendría un datagrama grande.

2.2.3 Integración de MQTT con Plataformas IoT

El protocolo MQTT se ha convertido en una pieza clave en la integración de dispositivos IoT con plataformas de gestión en la nube y entornos de automatización. Su capacidad para gestionar flujos de datos en tiempo real y su eficiencia en el uso del ancho de banda lo hacen ideal para conectar sistemas distribuidos y heterogéneos.

Pasos para la integración de MQTT en plataformas IoT

La implementación de MQTT en una plataforma IoT sigue un proceso estructurado, compuesto por los siguientes pasos, mostrados en la Figura 2.11:

1. Iniciar Comunicación MQTT: Se establece un canal de comunicación mediante el protocolo MQTT, permitiendo la transmisión de datos entre dispositivos IoT y la plataforma de gestión.
2. Integrar con Plataforma IoT: Se conecta el sistema de mensajería con la infraestructura de la plataforma, asegurando la recepción y el procesamiento adecuado de los mensajes MQTT.
3. Gestionar Datos en Tiempo Real: La plataforma organiza y procesa los datos entrantes, permitiendo la visualización y análisis en tiempo real.
4. Analizar Datos en Tiempo Real: Se extrae información valiosa de los datos recibidos, generando informes y acciones automatizadas para la toma de decisiones.

Esta integración permite a los dispositivos IoT compartir información de manera eficiente con plataformas en la nube, habilitando funciones avanzadas como el monitoreo remoto, la automatización de procesos y la optimización de recursos.

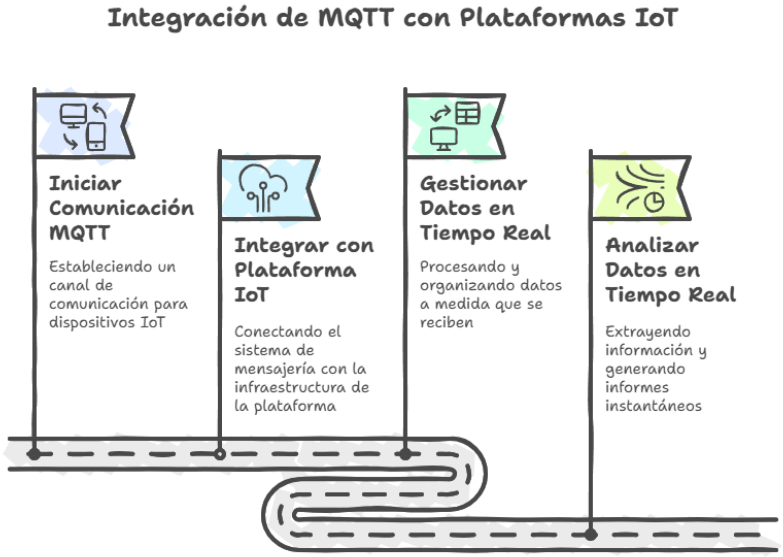


Figura 2.11 Esquema de integración de MQTT con plataformas IoT, mostrando los pasos clave en la gestión y análisis de datos en tiempo real. Elaboración propia.

La incorporación de MQTT en plataformas IoT facilita la interoperabilidad entre dispositivos de diferentes fabricantes y garantiza una comunicación fluida en entornos distribuidos. Al mismo tiempo, reduce el consumo de ancho de banda, tiene baja latencia y soporta *constrain devices*, enunciados en el capítulo anterior.

En los siguientes capítulos, se explorarán casos de uso y ejemplos prácticos de implementación para optimizar la integración de MQTT en aplicaciones industriales, domótica y ciudades inteligentes.

2.3 Procesamiento de datos en IoT

2.3.1 Componentes Clave del Análisis de Datos en IoT

El análisis de datos en IoT es un proceso esencial para extraer valor de la información generada por los dispositivos conectados. Este proceso se basa en los dos componentes mostrados en la Figura 2.12:

1. Técnicas de Análisis: Incluyen métodos estadísticos, algoritmos de aprendizaje automático y técnicas de minería de datos utilizadas para interpretar la información capturada por sensores y dispositivos IoT.
2. Modelos de Almacenamiento: Son las estructuras diseñadas para gestionar y conservar grandes volúmenes de datos provenientes de dispositivos IoT. Pueden incluir bases de datos relacionales, NoSQL y almacenamiento en la nube.

Figura 2.12 Representación de los componentes clave del análisis de datos en IoT, destacando las técnicas de análisis y los modelos de almacenamiento. Elaboración propia.

Importancia del Análisis de Datos en IoT

El análisis de datos en IoT permite:

Consejo Pedagógico:
Tip: Para analizar datos en IoT se requiere:

- **Recolección de datos:** Captura de señales a través de sensores.
- **Almacenamiento:** Bases de datos locales o en la nube.
- **Análisis:** Aplicación de modelos predictivos, machine learning.

- Detectar patrones y tendencias en el comportamiento de los dispositivos conectados.
- Optimizar la eficiencia operativa mediante el monitoreo en tiempo real.
- Mejorar la toma de decisiones basada en datos.

En los siguientes epígrafes, se detallarán las estrategias y herramientas específicas utilizadas en cada uno de estos componentes para optimizar la gestión y procesamiento de datos en entornos IoT.

2.3.2 Comparación entre Edge Computing y Fog Computing en IoT

En la infraestructura de IoT, el procesamiento de datos se puede realizar en diferentes niveles, destacando Edge Computing y Fog Computing como dos enfoques fundamentales para la gestión eficiente de la información generada por los dispositivos conectados. Entonces, podemos preguntarnos: ¿Dónde debe ser procesada la información? Todas las soluciones son válidas, depende del cliente, de la infraestructura ya instalada, del dominio en sí de aplicación. No obstante, podemos sugerir que el procesamiento se debe realizar lo más cercano al origen del dato.

En la Figura 2.13 podemos ver una relación donde típicamente se procesa la información relacionada a las capas de una arquitectura IoT. Debemos separa dichos conceptos, arquitectura y procesamiento de los datos.

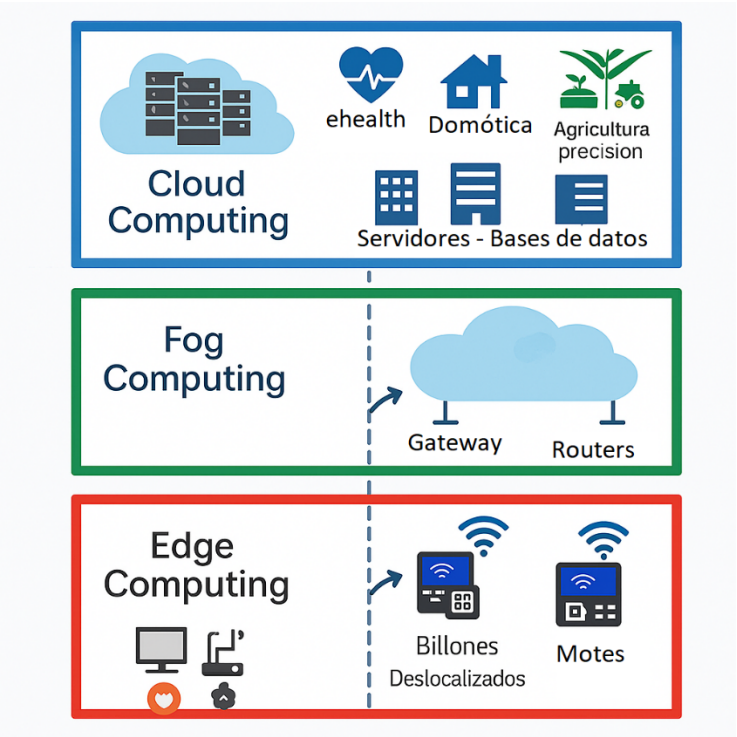


Figura 2.13 Niveles de procesamiento de los datos en IoT. Nivel superior: Cloud computing, nivel intermedio: Fog computing y nivel bajo: Edge computing. Elaboración propia.

Edge Computing

Edge Computing se caracteriza por llevar el procesamiento de datos lo más cerca posible del origen, es decir, en los propios dispositivos IoT o gateways intermedios. Sus principales ventajas son:

- Menor latencia: Al procesar datos en el borde de la red, se minimizan los tiempos de respuesta, lo que es crucial en aplicaciones de tiempo real.
- Reducción de la carga en la nube: Disminuye la cantidad de datos enviados a servidores centrales, optimizando el uso del ancho de banda.
- Mayor privacidad y seguridad: La información sensible se puede procesar localmente sin necesidad de enviarla a servidores externos.

Fog Computing

Fog Computing amplía el concepto de Edge Computing al agregar una capa intermedia de procesamiento entre los dispositivos IoT y la nube. Este modelo permite distribuir la carga de trabajo de manera más eficiente en una red de nodos cercanos a los dispositivos, ofreciendo:

- Mayor cobertura: Distribuye el procesamiento en diferentes niveles, mejorando la escalabilidad del sistema.
- Optimización del tráfico de red: Reduce la congestión en la comunicación con la nube al procesar parte de los datos localmente.
- Soporte para arquitecturas complejas: Facilita la integración de múltiples dispositivos y sensores en entornos industriales y de ciudades inteligentes.

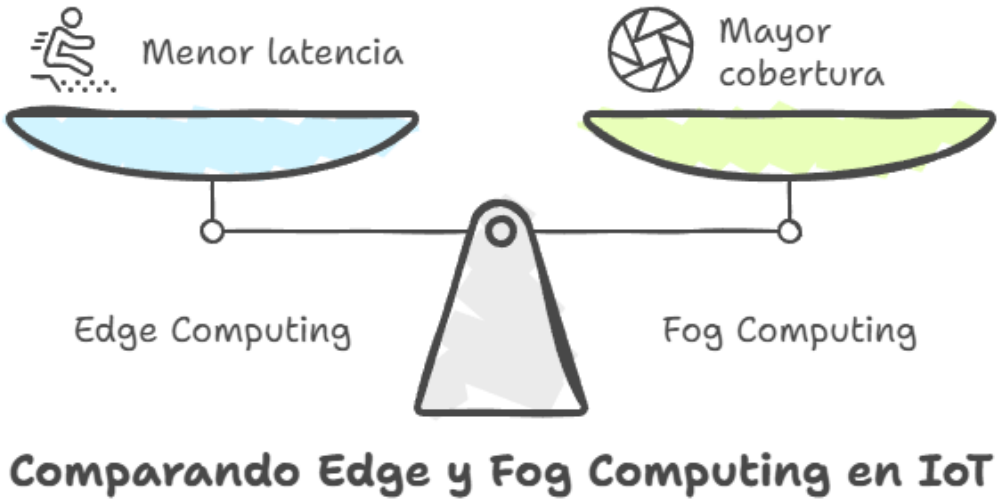


Figura 2.14 Comparación entre Edge Computing y Fog Computing, destacando sus diferencias en latencia y cobertura en IoT. Elaboración propia.

Ambos modelos tienen aplicaciones específicas dependiendo de las necesidades del sistema IoT. Mientras que Edge Computing es ideal para aplicaciones que requieren respuestas inmediatas, como vehículos autónomos o sistemas de seguridad, Fog Computing es más adecuado para arquitecturas distribuidas que manejan grandes volúmenes de datos en múltiples ubicaciones, ver Figura 2.14.

En este epígrafe no hemos mencionado a la Cloud computing, ya que este tema será abordado con mayor profundidad en el próximo capítulo.

✍️ ¿En qué casos preferirías Edge Computing sobre Fog Computing?

2.3.3 Integración de IoT y AI

La integración del Internet de las Cosas (IoT) con la Inteligencia Artificial (AI) ha revolucionado la forma en que los dispositivos recopilan, procesan y utilizan datos para tomar decisiones autónomas. La fusión de estas tecnologías permite la creación de sistemas inteligentes que pueden analizar grandes volúmenes de datos en tiempo real y generar respuestas automatizadas.

Evolución de IoT hacia Sistemas Inteligentes

El desarrollo de IoT y su combinación con AI sigue una evolución progresiva que se puede dividir en las siguientes etapas mostradas en la Figura 2.15:

1. IoT Solo: Sistemas de conectividad básicos donde los dispositivos recopilan datos y los envían a plataformas para su análisis manual.
2. Combinar Tecnologías: Se inicia la fusión de IoT con AI, permitiendo que los sistemas interpreten datos de manera automatizada.
3. Mejorar la Inteligencia: Aplicación de algoritmos avanzados de machine learning para mejorar la toma de decisiones basada en datos en tiempo real.
4. Aumentar la Autonomía: Habilitación de capacidades de auto-operación y respuesta automática en dispositivos IoT.
5. Sistemas Inteligentes: Creación de soluciones totalmente autónomas con capacidad de aprendizaje continuo y optimización en tiempo real.

Beneficios de la Integración de IoT y AI

- Optimización de procesos: Reducción de tiempos de respuesta y mejora en la eficiencia operativa.
- Automatización avanzada: Implementación de sistemas capaces de tomar decisiones sin intervención humana.
- Análisis predictivo: Uso de algoritmos para prever fallos en sistemas y optimizar el mantenimiento de dispositivos IoT.
- Mayor personalización: Adaptación de los servicios basados en los patrones de comportamiento de los usuarios.

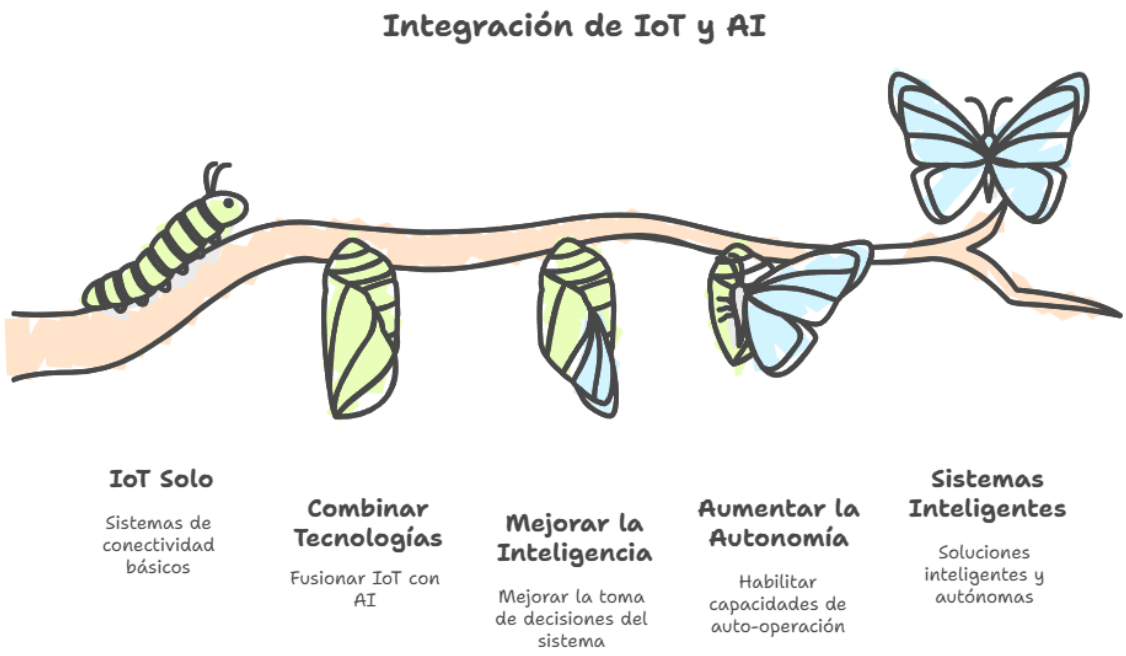


Figura 2.15 Representación del proceso de integración de IoT y AI, mostrando la evolución desde sistemas básicos hasta soluciones inteligentes y autónomas. Elaboración propia.

Con la combinación de IoT y AI, se están desarrollando innovaciones en áreas como ciudades inteligentes, salud, domótica, manufactura y transporte, permitiendo la creación de sistemas interconectados que pueden aprender y evolucionar con el tiempo.

Consejo Pedagógico:
Tip: La fusión de IoT con Inteligencia Artificial permite:

- Toma de decisiones autónoma.
- Detección de patrones complejos.
- Predicción y optimización de procesos.

Elementos de Cierre

Resumen del Capítulo

Se revisaron los principales protocolos de comunicación IoT (REST-API, MQTT) y los esquemas de procesamiento de datos (Edge Computing, Fog Computing) incluyendo su integración con Inteligencia Artificial para el

desarrollo de sistemas inteligentes. En el próximo capítulo terminaremos por revisar algunas de las plataformas IoT "pre-enlatadas" listas para su uso disponibles en el mercado y una aplicación web open-source que nos permitirá implementar y gestionar los datos de capas inferiores en la capa superior de aplicaciones.

Preguntas de Revisión

❓ Opción múltiple:
¿Qué protocolo está basado en un modelo publicación/suscripción?
a) REST
b) HTTP
c) MQTT
d) SOAP

✅ Verdadero/Falso:
Edge Computing significa procesar datos exclusivamente en la nube.
(Verdadero / Falso)

✍️ Respuesta corta:
Menciona dos ventajas principales del protocolo MQTT para dispositivos IoT.

Clave de Respuestas

- ❓ Opción múltiple: c) MQTT
- ✅ Verdadero/Falso: Falso (Edge Computing procesa datos en el borde, no en la nube).
- ✍️ Respuesta corta: Baja latencia, bajo consumo de ancho de banda.

Referencias bibliográficas.

- Banks, A., Briggs, E., Borgendale, K., & Gupta, R. (Eds.). (2019). MQTT Version 5.0. OASIS Standard.
- HiveMQ. (2025, 10 de mayo). *MQTT Broker and IoT Platform*. Recuperado el 10 de mayo de 2025, de https://www.hivemq.com/
- MQTTX. (2025, 10 de mayo). *MQTTX – An Elegant MQTT Client Tool*. Recuperado el 10 de mayo de 2025, de https://mqttx.app/

- Shelby, Z., Hartke, K., & Bormann, C. (2014). *The constrained application protocol (CoAP)* (RFC 7252). RFC Editor. https://doi.org/10.17487/RFC7252

3 Capítulo 3: Plataformas IoT y su Implementación

Elementos de Apertura

Objetivos de aprendizaje:

- Explorar plataformas de desarrollo y gestión de dispositivos IoT.
- Implementar flujos IoT utilizando Node-RED.
- Programar microcontroladores para soluciones IoT.

Competencias:

- Integrar dispositivos en plataformas locales y en la nube.
- Desarrollar flujos de automatización utilizando herramientas visuales.
- Programar y configurar dispositivos embebidos para aplicaciones IoT.

Preguntas de enfoque:

- ¿Qué plataforma IoT conviene seleccionar para un proyecto específico?
- ¿Cómo facilita Node-RED la integración de sensores y servicios?
- ¿Cuáles son los pasos básicos para programar un microcontrolador usando VSCode y PlatformIO?

La Figura 3.1 resume algunos aspectos que serán tratados en este capítulo, por ejemplo, la herramienta de desarrollo Node-red puede ser usada para crear elementos de cualquiera de las capas de arquitectura Iot. Por otro lado, observamos un dashboard típico de la capa de aplicaciones y una combinación efectiva del uso de Vscode con PlatformIO como una herramienta potente y profesional para crear el firmware en la capa de dispositivos.

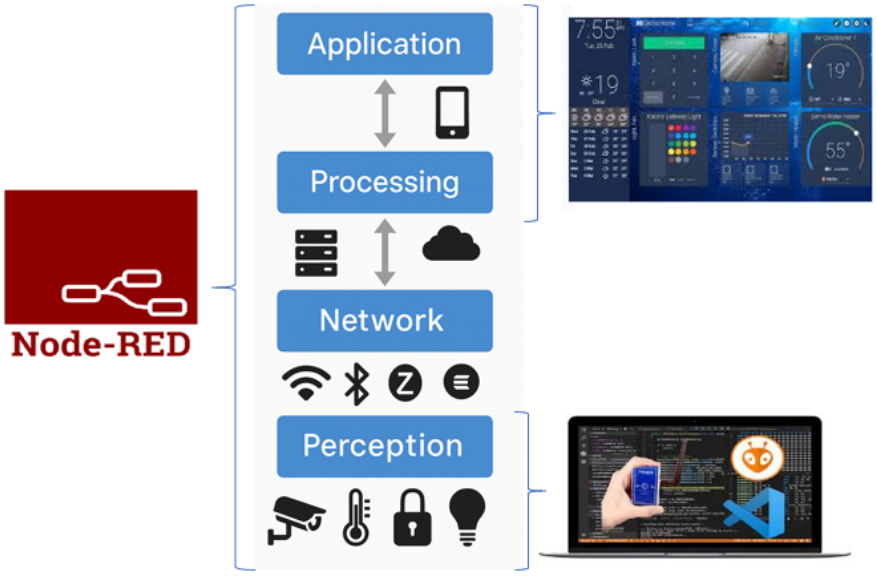


Figura 3.1 Plataformas y herramientas de desarrollo usadas en IoT. Elaboración propia.

3.1 Introducción a las Plataformas IoT

Las plataformas IoT han evolucionado hasta convertirse en el núcleo de la interconectividad entre dispositivos, sensores y servicios en la nube. Estas plataformas facilitan la recolección, procesamiento y análisis de datos, así como la automatización de tareas y la integración con sistemas de inteligencia artificial.

Consejo Pedagógico:
Tip: ¿Por qué usar una plataforma IoT?

- Conectar hardware, como sensores y dispositivos.
- Manejar diferentes protocolos de comunicación entre hardware y software.
- Proporcionar seguridad y autentificación para dispositivos y usuarios.
- Recopilar, visualizar y analizar los datos que los sensores y dispositivos recopilan.

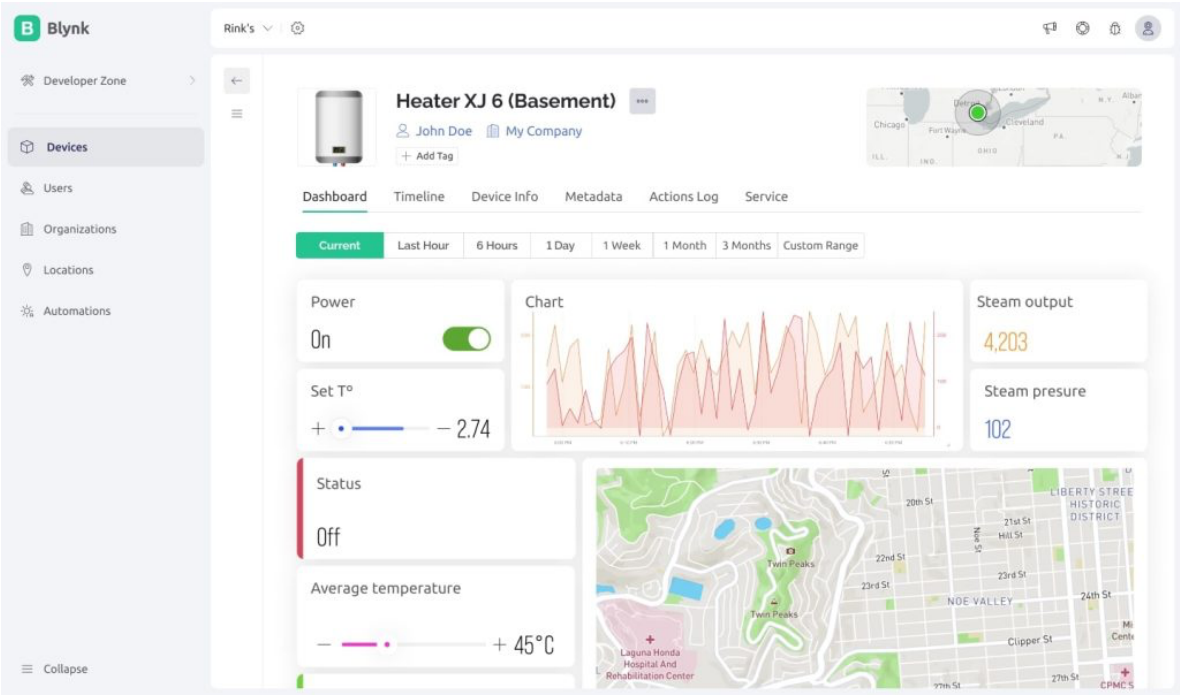


Figura 3.2 Dashboard de la plataforma IoT Blynk, adaptada de Blynk (2025, 10 de mayo).

Existen actualmente varias plataformas ya disponibles para Iot, prácticamente plug and play. Estas plataformas pueden estar en la cloud o instaladas en servidores o hardware dedicado en las capas intermedias o bajas de la arquitectura de IoT. Obviamente, en la cloud, se dispone de mayor potencia, memoria y capacidad de los servidores instalados, con la desventaja que la mayoría son de pago o gratis con funcionalidades limitades a la hora de implementar una solución IoT a un cliente final. En la cloud, podemos encontrar desde pequeñas empresas o emprendimientos hasta los gigantes actuales, dígase Amazon, Microsoft y Google, en la siguiente imagen se aprecia el dashboard de la plataforma de Blynk, ver Figura 3.2 y OpenHab en la Figura 3.3.

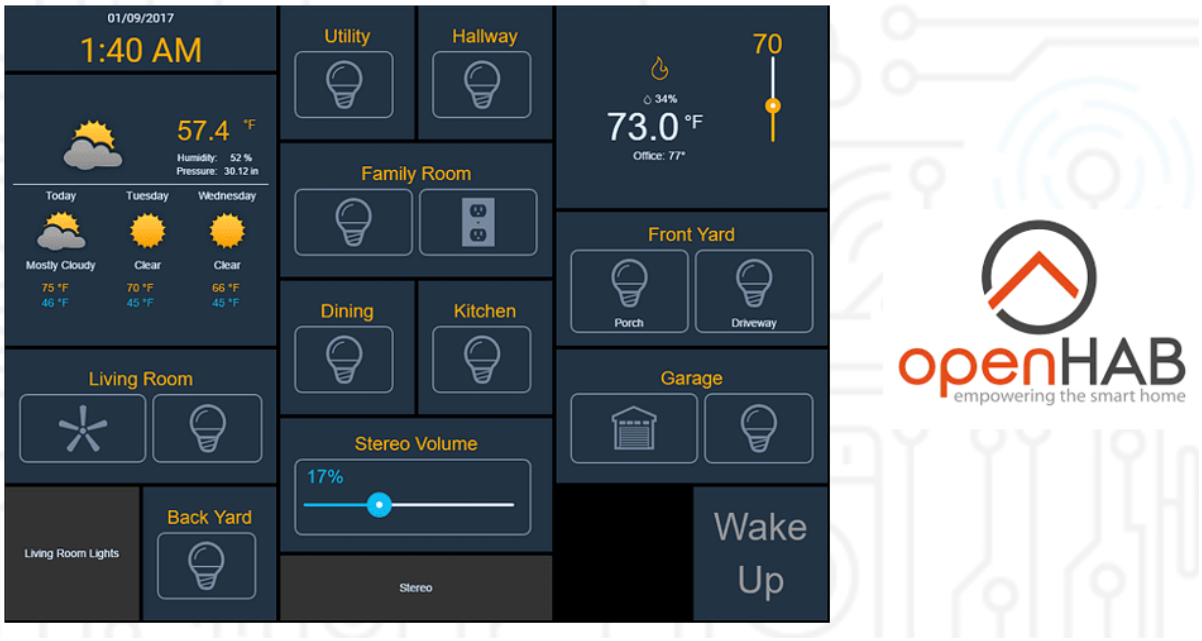


Figura 3.3 Dashboard de la plataforma domótica openHAB, adaptada de openHAB (2025, 10 de mayo).

Entre las plataformas domóticas más utilizadas en IoT se encuentra Home Assistant, un sistema de código abierto diseñado para la automatización del hogar, que permite la integración de múltiples dispositivos y servicios de manera eficiente y escalable.

3.1.1 Home Assistant y la Automatización del Hogar

Home Assistant es una plataforma de automatización de código abierto que permite la integración y gestión de dispositivos IoT en un ecosistema doméstico. Su arquitectura modular facilita la interconexión de dispositivos de diferentes fabricantes, brindando al usuario un alto nivel de personalización y control como se puede observar en la Figura 3.4.

Componentes Claves de Home Assistant

1. Dispositivos IoT: Sensores, actuadores y controladores conectados a la plataforma que permiten la monitorización y ejecución de acciones dentro del hogar.
2. Integración: Capacidad de conectar múltiples dispositivos a través de protocolos como MQTT, Zigbee y Z-Wave, permitiendo la interoperabilidad entre distintos fabricantes.

3. Automatización: Creación de reglas y rutinas que permiten automatizar tareas del hogar, optimizando el uso de la energía y mejorando la comodidad del usuario.

En la Figura 3.5se resumen los componentes claves de Hassio y su funcionamiento.

Figura 3.4 Dashboard de la plataforma domótica Home Assistant, adaptada de Home Assistant (2025, 10 de mayo).

Beneficios de Implementar Home Assistant

- Independencia de la Nube: Permite la gestión local de los dispositivos, reduciendo la dependencia de servicios externos.
- Compatibilidad Extensa: Admite una amplia gama de dispositivos y protocolos de comunicación.
- Seguridad y Privacidad: Al ser una solución local, los datos del usuario permanecen dentro de su red privada.

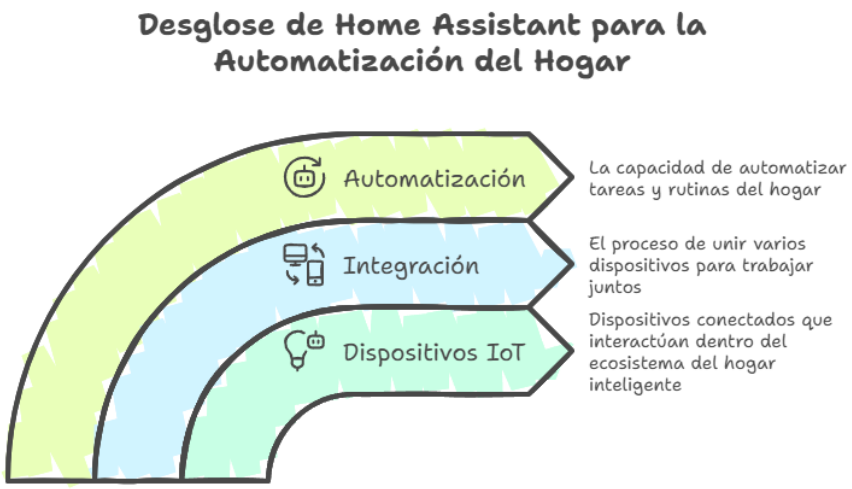


Figura 3.5 Desglose de Home Assistant para la automatización del hogar, ilustrando sus componentes clave y su funcionamiento. Elaboración propia.

En los siguientes epígrafes, se detallarán los métodos de implementación de Home Assistant, su integración con otras plataformas y ejemplos prácticos de automatización avanzada.

Consejo Pedagógico:

Tip: Home Assistant es una plataforma de automatización local que:

- Integra múltiples protocolos como MQTT, Zigbee, Z-Wave, BLE y Matter.
- Permite control local sin depender de servicios externos.
- Facilita la creación de automatizaciones mediante flujos lógicos.
- Permite la integración con muchos fabricantes de dispositivos domóticos.

3.1.2 Plataformas en la Nube para IoT

Las plataformas en la nube han permitido la escalabilidad y flexibilidad de las soluciones IoT, facilitando la gestión remota de dispositivos y el procesamiento avanzado de datos. Entre las principales plataformas en la nube para IoT se encuentran:

1. AWS IoT Amazon Web Services **(**Figura 3.6**)** ofrece una infraestructura escalable y segura para la gestión de dispositivos IoT, con herramientas avanzadas para el análisis de datos y la automatización de procesos.

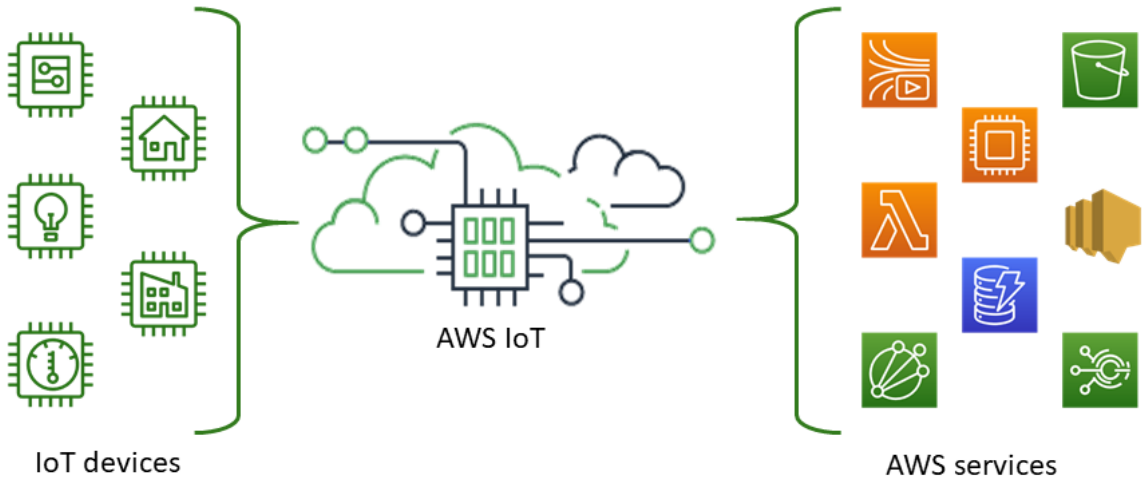


Figura 3.6 Arquitectura resumida de AWS IoT, adaptada de AWS IoT (2025, 10 de mayo)

Un caso práctico a implementar en esta plataforma podría ser el mantenimiento predictivo de una línea de producción. La idea sería sensar la vibración y temperatura en motores para detectar desgaste y prevenir paradas.

Arquitectura (mínima y probada):

1. **Edge:** Sensores → gateway (Raspberry Pi/industrial PC) con **AWS IoT Greengrass v2** para agregación y reglas locales (p. ej., umbrales, downsampling).
2. **Ingesta segura:** Dispositivos/gateway → **AWS IoT Core** (MQTT) con políticas y certificados.
3. **Reglas y storage time-series:** Regla de **IoT Core** que enruta telemetría a **Amazon Timestream** (o DynamoDB) para consultas por tiempo y paneles.
4. **Acciones/Alertas: AWS Lambda** evalúa KPIs (RMS vibración, delta T) y dispara **SNS** (correo/SMS).
5. **Visualización: Amazon QuickSight** o Grafana para tendencias y alarmas (consumiendo Timestream).

Beneficios: Menos downtime, mantenimiento planificado y trazabilidad por activo.

2. Google Cloud IoT: Proporciona herramientas de aprendizaje automático y análisis de datos en tiempo real, permitiendo la optimización de procesos industriales y la toma de decisiones basada en datos, como se aprecia en la Figura 3.7.

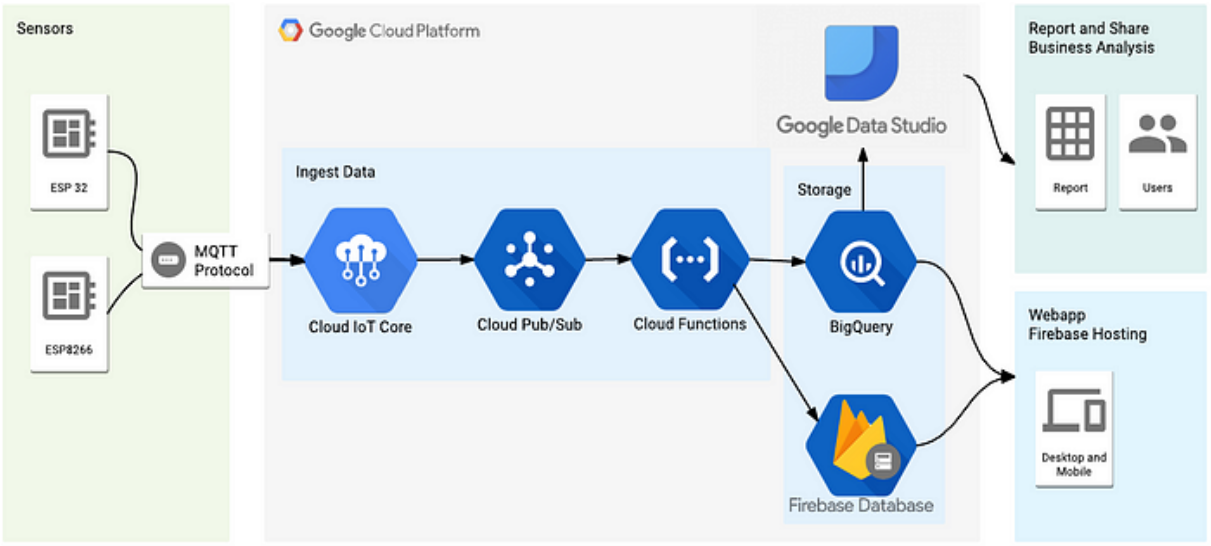


Figura 3.7 Arquitectura resumida de Google Cloud IoT, adaptada de Google Cloud (2025, 10 de mayo).

Con la ayuda de esta plataforma podemos implementar una red de micro estaciones de calidad del aire para una ciudad o un campus universitario.

Arquitectura actual en GCP:

1. **Edge/broker:** Nodos con sensores (PM2.5, NO_2, CO_2) publican MQTT a **HiveMQ/EMQX/Mosquitto gestionado**. Extensión/binding **Pub/Sub** envía los mensajes a **Google Cloud Pub/Sub**
2. **Streaming ETL: Dataflow** consume de Pub/Sub, valida esquemas, enriquece (geo, calibración) y escribe en **BigQuery** (histórico) + **Bigtable**/**Firestore** (lecturas recientes).
3. **Alertas: Cloud Functions/Cloud Run** suscrito a Pub/Sub dispara notificaciones cuando AQI supera umbrales.
4. **Analytics/ML: BigQuery** + **Vertex AI** para detección de anomalías (episodios de contaminación).
5. **Dashboards: Looker Studio** sobre BigQuery para mapas de calor y series temporales.

Beneficios: Mapas en tiempo real, alertas tempranas y evidencia para políticas ambientales sin depender de IoT Core.

3. Azure IoT: La plataforma de Microsoft mostrada en la Figura 3.8 integra soluciones IoT con su ecosistema en la nube, brindando conectividad segura y servicios avanzados de analítica.

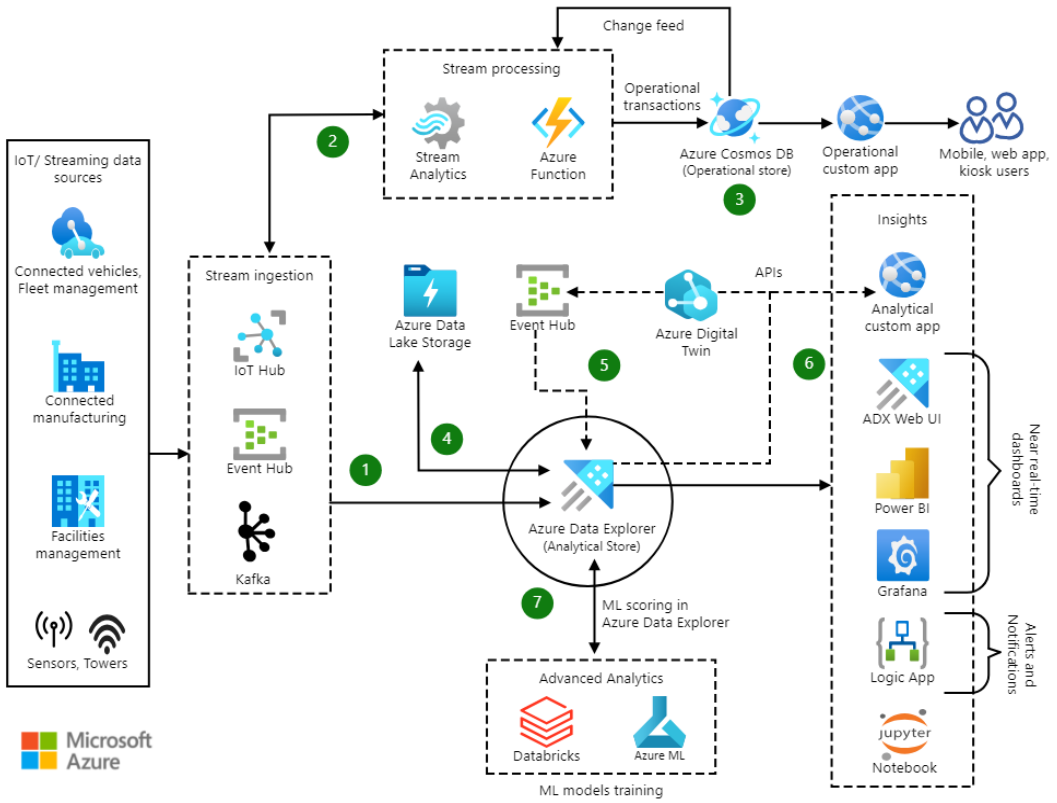


Figura 3.8 Arquitectura resumida de Microsoft Azure IoT, adaptada de Azure IoT (2025, 10 de mayo)

Un caso práctico a implementar con esta plataforma podría ser una cadena de frío conectada para vacunas. La idea es monitorear temperatura y ubicación de contenedores; alertar y registrar eventos de apertura/choque térmico.

Arquitectura recomendada:

1. **Ingesta y control:** Dispositivos (BLE/LTE) → **Azure IoT Hub** (MQTT/AMQP), con identidad y políticas por dispositivo.
2. **Borde inteligente:** En hubs de almacén, **Azure IoT Edge** ejecuta lógica local (p. ej., descartar outliers, buffer offline).
3. **Streaming y reglas: Azure Stream Analytics** enruta desde IoT Hub a **Data Lake / Synapse / Azure Data Explorer** y genera **alertas** (Event Grid + **Logic Apps** o Functions).

4. **Gestión de dispositivos: Device Update for IoT Hub** para OTA (firmware y config).
5. **Visualización: Power BI** con datasets de Synapse/Data Explorer para monitoreo en tiempo real y reporting de cumplimiento.

Beneficios: Cumplimiento normativo, reducción de pérdidas por ruptura de cadena y trazabilidad end-to-end.

En la Figura 3.9 se comparan las 3 plataformas antes mencionadas.

Beneficios de Utilizar Plataformas IoT en la Nube

- Escalabilidad: Permiten gestionar una gran cantidad de dispositivos sin comprometer el rendimiento del sistema.
- Seguridad: Implementan protocolos avanzados de autenticación y cifrado para proteger los datos transmitidos.
- Análisis Avanzado: Integración con inteligencia artificial y big data para la toma de decisiones automatizada.

Consejo Pedagógico:
Tip: Principales opciones:

- **AWS IoT Core:** Alta escalabilidad y robustez.
- **Google Cloud IoT:** Integración nativa con herramientas de Big Data.
- **Azure IoT Hub:** Soluciones híbridas entre nube y borde.

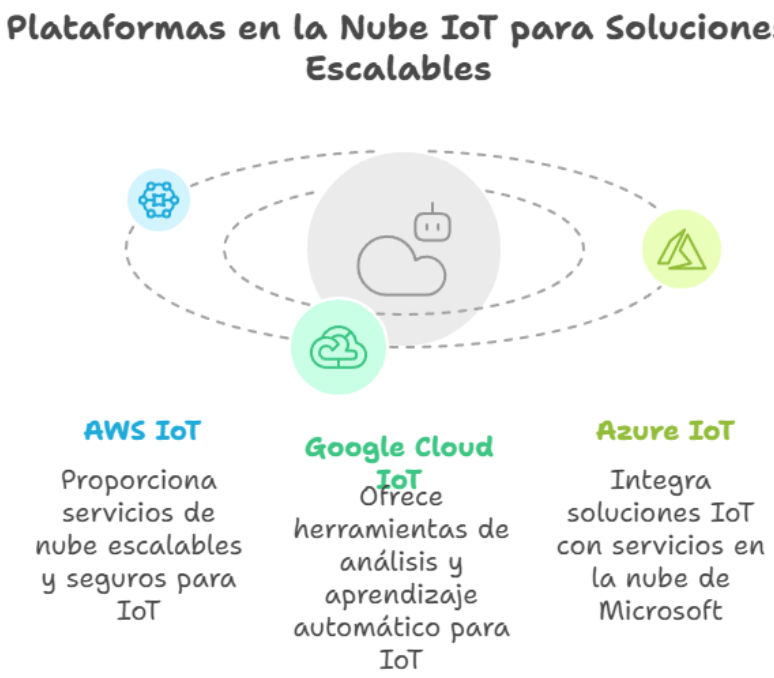


Figura 3.9 Comparación de plataformas en la nube para IoT, destacando AWS IoT, Google Cloud IoT y Azure IoT. Elaboración propia.

En los siguientes epígrafes se analizarán casos de uso específicos y estrategias para la integración eficiente de estas plataformas en soluciones IoT.

3.1.3 Criterios para la Selección de una Plataforma IoT

La selección de una plataforma IoT adecuada es un paso crucial para garantizar el éxito de un proyecto, ya que influye en la escalabilidad, la integración y la operatividad del sistema. Los principales criterios por considerar, mostrados en la Figura 3.10, son:

1. Características: Evaluar las capacidades y herramientas ofrecidas por la plataforma para asegurar que satisfaga las necesidades específicas del proyecto.
2. Facilidad de uso: Considerar la intuitividad y la curva de aprendizaje de la plataforma para garantizar un desarrollo eficiente.
3. Costo: Analizar la estructura de precios de la plataforma para verificar que se ajuste al presupuesto del proyecto.

Importancia de una Selección Adecuada

La correcta elección de una plataforma IoT impacta directamente en la eficiencia del sistema, la capacidad de procesamiento de datos y la facilidad de implementación. Optar por una plataforma que cumpla con estos criterios garantiza un entorno optimizado y escalable.

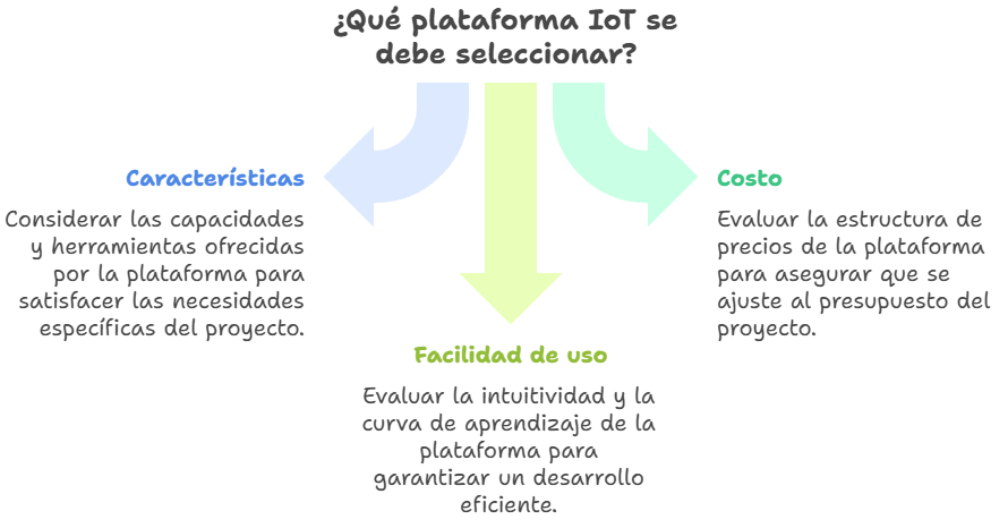


Figura 3.10 Representación de los principales criterios a considerar al seleccionar una plataforma IoT. Elaboración propia.

En los siguientes epígrafes se explorarán estrategias para la integración efectiva de estas plataformas en soluciones IoT específicas.

Orquestaciones en IoT (IoT Orquestration)

Antes de diseñar tu solución IoT, debemos estudiar previamente el siguiente concepto: IoT Orchestration, que se refiere al proceso de gestionar, coordinar y optimizar la interacción entre dispositivos IoT, redes de comunicación, procesamiento de datos y servicios en la nube. Su objetivo es garantizar que los sistemas IoT funcionen de manera eficiente, segura y escalable, permitiendo que múltiples dispositivos y plataformas trabajen en conjunto sin problemas.

En un ecosistema IoT, donde miles o millones de dispositivos recopilan, procesan y transmiten datos en tiempo real, la orquestación se vuelve esencial para:

- Automatizar la comunicación entre dispositivos.
- Optimizar el uso de los recursos computacionales y de red.

- Implementar estrategias de escalabilidad y tolerancia a fallos.
- Integrar múltiples protocolos y arquitecturas en un solo sistema.

Algunos requerimientos son:

- Dispositivos: Interoperables, es decir, compatibles con múltiples protocolos de comunicación. Autónomos, con capacidad de actuar según reglas predefinidas. Energéticamente eficientes, para maximizar su operatividad en redes de baja potencia.
- Redes de comunicación: La orquestación debe gestionar eficientemente el tráfico de datos para evitar congestiones y mejorar la latencia en la comunicación.
- Procesamiento y análisis de datos: La orquestación en IoT permite determinar dónde y cuándo se deben procesar los datos, dependiendo de la necesidad de cada aplicación.
- Plataformas de gestión y seguridad: Los mecanismos de seguridad orquestados en IoT incluyen: **Autenticación y control de acceso** (TLS, OAuth, certificados digitales), **Encriptación de datos** en tránsito y en reposo y **Monitorización en tiempo real** para detectar y mitigar amenazas.

Modelos de Orquestaciones en IoT (IoT Orquestration)

Existen dos enfoques principales para la orquestación en IoT:

- Orquestación Centralizada: En este modelo, todos los dispositivos y procesos están controlados desde un único sistema o servidor central. Las decisiones, como el enrutamiento de datos y la asignación de recursos, son gestionadas por una plataforma centralizada, ver
- Figura 3.11 (izquierda).

  Ventajas:

  - Fácil gestión y monitoreo desde un solo punto de control.
  - Alta coherencia y sincronización de los datos.
  - Simplificación de la seguridad, ya que todo se administra desde un nodo central.

  Desventajas:

  - Mayor latencia en aplicaciones que requieren respuestas inmediatas.

- Riesgo de puntos únicos de fallo, si el servidor central deja de funcionar.
- Altos costos de infraestructura, debido a la demanda de procesamiento y almacenamiento en la nube.

Ejemplo de Uso:
Monitoreo y control de tráfico en una ciudad mediante una plataforma que centraliza todos los datos y genera patrones de movilidad en tiempo real.

- Orquestación Descentralizada: En este modelo, las decisiones y el procesamiento de datos ocurren de manera distribuida, en distintos nodos de la red (Edge o Fog Computing). Cada dispositivo tiene mayor autonomía y puede tomar decisiones localmente, ver
- Figura 3.11 (derecha).

Ventajas:

- Mayor escalabilidad, ya que los dispositivos pueden operar de manera independiente.
- Menor latencia, debido a que el procesamiento ocurre cerca de la fuente de datos.
- Mayor resiliencia, ya que un fallo en un nodo no afecta el resto del sistema.

Desventajas:

- Mayor complejidad en la gestión, ya que se requiere sincronización entre múltiples nodos.
- Mayor consumo energético en dispositivos Edge que procesan datos localmente.

Ejemplo de Uso:
Monitoreo ambiental en un bosque con sensores que procesan datos localmente y solo envían alertas a la nube cuando se detectan anomalías.

✍️ Respuesta corta**:** A partir de toda la información brindada hasta esta parte del libro, intenta definir la arquitectura, elementos y orquestación de los siguientes casos:

- Control de tráfico de una ciudad

- Control de iluminación en el hogar
- Monitoreo salud remoto
- Monitoreo de flotas de vehículos
- Control de riego parques y jardines
- Monitoreo temperatura y humedad de flores en contenedor
- Rastreo y gestión de activos
- Alerta temprana desastres naturales y Control calidad del agua

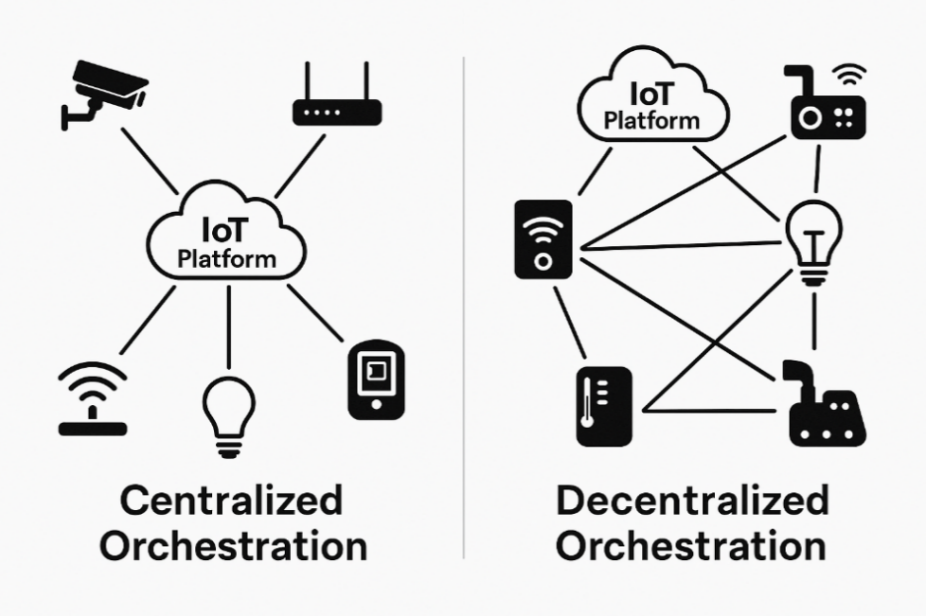


Figura 3.11 Modelos de Orquestación en IoT: Centralizada (izquierda) vs Descentralizada (derecha). Elaboración propia.

✍ ¿Qué criterios utilizarías para elegir entre una plataforma en la nube o una local?

Consejo Pedagógico:
Tip: Factores claves para seleccionar una plataforma IoT:

- Costos de implementación.
- Nivel de seguridad ofrecido.
- Escalabilidad y flexibilidad.
- Soporte de múltiples protocolos.

3.2 Desarrollo de Soluciones IoT con Node-RED

Node-RED, Node-RED. (2025, 10 de mayo), es una herramienta de programación visual basada en flujos, diseñada para la integración y automatización de sistemas IoT. Su flexibilidad y facilidad de uso la han convertido en una de las opciones más populares para desarrollar soluciones IoT sin necesidad de programación extensa.

Node-red en principio es un servidor web, por tanto, podemos invocarlo en una PC con Windows a través de una consola, como se aprecia en la Figura 3.12. Al ser un servidor web, solo requerimos de un browser como herramienta de desarrollo, pudiéndose acceder desde varios medios como una Tablet, un PC o un smartphone.

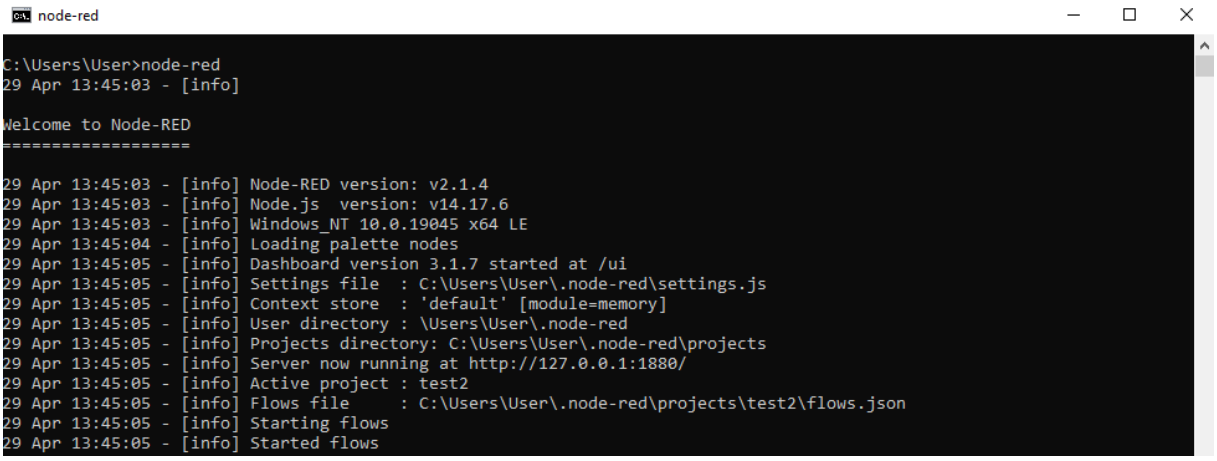


Figura 3.12 Consola en Windows, a través del comando node-red invocamos al servidor. Elaboración propia.

Luego que hemos editado nuestro código (flujo de nodos), debemos hacer un deploy para que dicha aplicación web quede corriendo en el servidor web que hemos levantado. En la Figura 3.13 se muestra el interfaz de usuario (back end) de nuestras aplicaciones web hechas en node-red.

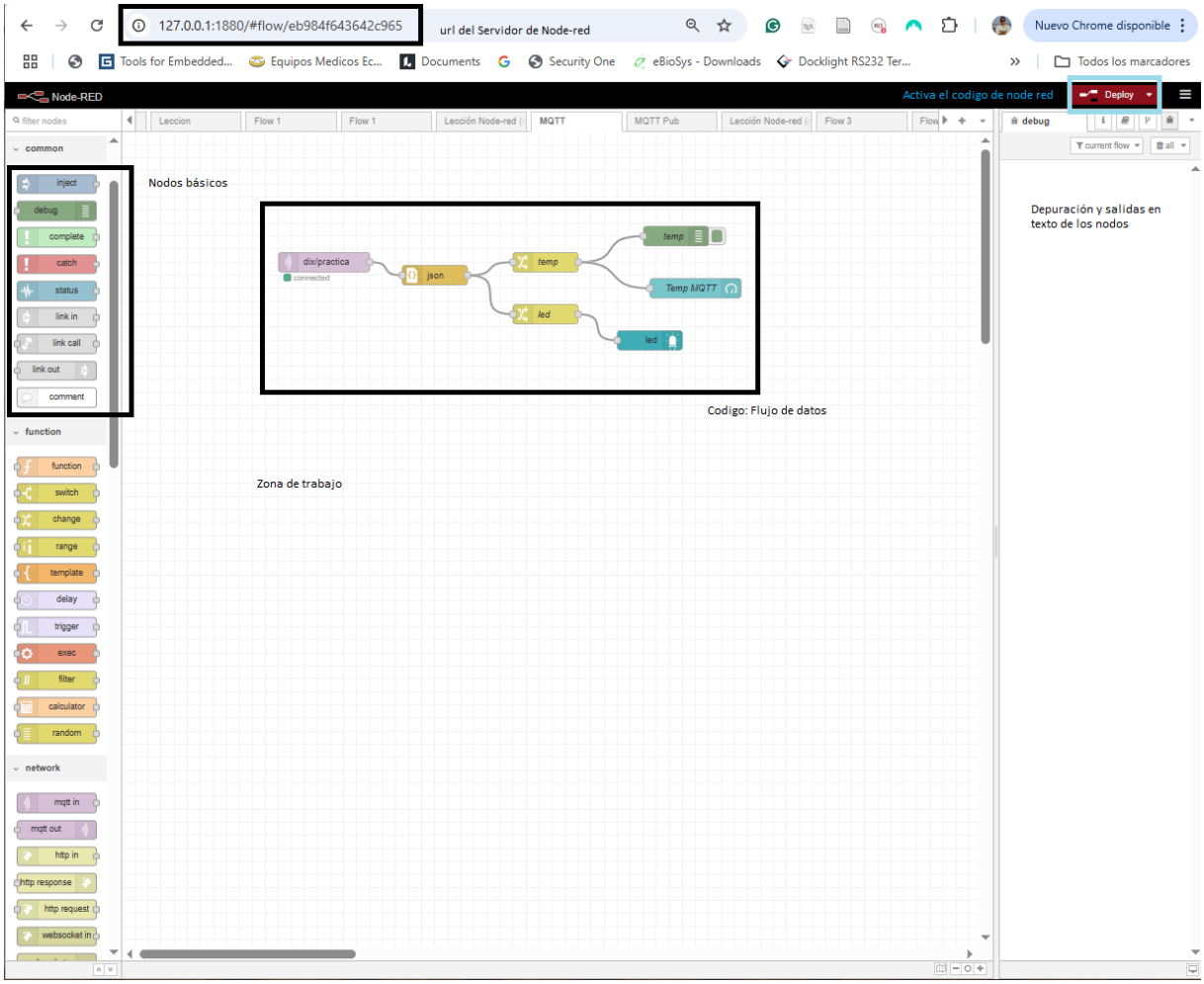


Figura 3.13 Área de desarrollo del código en node-red (back-end). Elaboración propia.

En la columna de la izquierda encontramos todos los nodos funcionales disponibles, no todos los nodos vienen instalados por default, por tanto, el programador tendrá un gran apoyo de la comunidad que ha desarrollado y libera una gran cantidad de nodos que debemos instalar. En la parte central, se encuentra el área de trabajo donde creamos nuestros flujos y estos pueden ser organizados en diferentes pestañas. Al lado derecho encontramos el área de configuración y depuración de todo node-red.

3.2.1 Ventajas y Desafíos de Node-RED en IoT

El uso de Node-RED en entornos IoT ofrece múltiples ventajas, pero también presenta ciertos desafíos que deben ser considerados que aparecen en la Figura 3.14.

Figura 3.14 Evaluación de los pros y contras de Node-RED en la implementación de soluciones IoT. Elaboración propia.

Ventajas de Node-RED

- Programación visual: Facilita el desarrollo de flujos sin necesidad de escribir código complejo.
- Integración fácil: Compatible con múltiples plataformas y protocolos como MQTT, HTTP y WebSockets.
- Soporte de la comunidad: Amplia base de usuarios y documentación disponible.
- Flexibilidad: Permite la creación de soluciones personalizadas para diversos casos de uso.
- Prototipado rápido: Ideal para la creación y prueba de flujos de datos en IoT.

Desafíos de Node-RED

- Curva de aprendizaje: Aunque es visual, se requiere conocimiento en programación para aprovechar todo su potencial.

- Problemas de rendimiento: Puede no ser la mejor opción para aplicaciones con requerimientos de alta velocidad y procesamiento intensivo.
- Escalabilidad limitada: Diseñado principalmente para soluciones locales o de pequeña escala.
- Dependencia de Node.js: Requiere la ejecución sobre un entorno basado en JavaScript.
- Preocupaciones de seguridad: Es fundamental aplicar medidas de seguridad adicionales en entornos críticos.

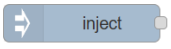
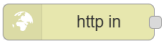
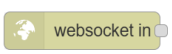
En los siguientes epígrafes se explorarán estrategias para la integración efectiva de Node-RED en plataformas IoT y ejemplos prácticos de su implementación.

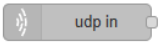
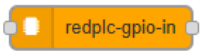
3.2.2 Integración de Dispositivos y Servicios en Node-RED

Node-RED permite la integración de diversos dispositivos IoT y servicios en la nube mediante una estructura modular basada en nodos. Cada nodo representa una función específica dentro del flujo de datos, facilitando la comunicación entre sensores, actuadores y plataformas de procesamiento. En Node-RED, los "nodos" son los bloques funcionales que se arrastran y conectan para crear flujos (flows). Cada nodo tiene una función específica, y pueden clasificarse en diferentes tipos, según su rol en el flujo de datos. A continuación, te presento una clasificación de los tipos principales de nodos en Node-RED:

- 1. Nodos de Entrada (Input nodes)

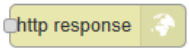
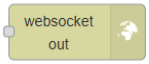
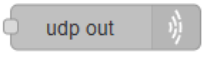
Estos nodos **reciben datos** desde fuentes externas y los inyectan en el flujo.

Nodo	**Descripción**
Inject:	Inyecta datos manualmente o en intervalos regulares 
mqtt in:	Recibe mensajes desde un broker MQTT 
http in:	Escucha peticiones HTTP entrantes 
websocket in:	Recibe datos a través de WebSocket 

Nodo	Descripción
serial in:	Recibe datos desde puertos serie 
tcp in:	Escucha conexiones TCP entrantes 
udp in:	Recibe datos desde conexiones UDP 
gpio in (rpi):	Lee entradas digitales en Raspberry Pi 

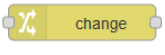
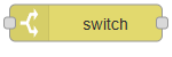
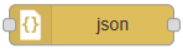
- 2. Nodos de Salida (Output nodes)

Estos **envían datos** hacia dispositivos, APIs o plataformas externas.

Nodo	Descripción
mqtt out:	Publica mensajes en un broker MQTT 
http response:	Devuelve una respuesta a una petición HTTP 
websocket out:	Envía datos por WebSocket 
tcp out:	Envía datos a través de TCP 
udp out:	Envía datos vía UDP 
gpio out (rpi):	Controla salidas digitales en Raspberry Pi 

- 3. Nodos de Procesamiento / Función (Function nodes)

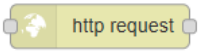
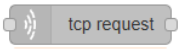
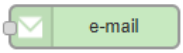
Manipulan y procesan los datos que fluyen entre nodos.

Nodo	Descripción
Function:	Permite escribir código JavaScript personalizado 
Change:	Cambia valores del mensaje (msg) sin código 
Switch:	Ruta condicional de datos, como un if lógico 
Template:	Inserta datos en plantillas de texto o HTML 
Json:	Convierte entre objetos JSON y texto plano 

Nodo	Descripción
Range:	Reescala un valor numérico (por ejemplo, 0-100 a 0-5) 
Delay:	Añade retardo o limita la frecuencia de los mensajes 

- 4. Nodos de Red (Network nodes)

Conectan con servicios o APIs externas a través de protocolos estándar.

Nodo	Descripción
http request:	Realiza llamadas HTTP (GET, POST, etc.) 
tcp request:	Se comunica vía sockets TCP 
Email:	Envío y recepción de correos electrónicos 

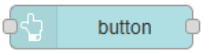
- 5. Nodos de Almacenamiento / Contexto

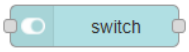
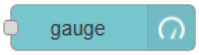
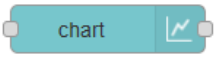
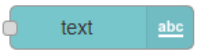
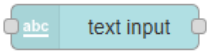
Usan bases de datos o memoria local para guardar datos.

Nodo	Descripción
file in/out:	Leer o escribir archivos 
Sqlite:	Acceso a bases de datos SQLite 
Mongodb:	Acceso a bases MongoDB 
Context:	Guarda variables en memoria temporal o persistente 

- 6. Nodos Dashboard (UI)

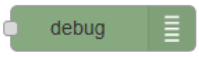
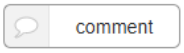
Permiten crear interfaces gráficas de usuario (dashboards) en tiempo real.

Nodo	Descripción
ui_button:	Botón interactivo 

Nodo	Descripción
ui_switch:	Interruptor tipo palanca 
ui_slider:	Control deslizante 
ui_gauge:	Medidor tipo velocímetro 
ui_chart:	Gráfica de líneas o barras 
ui_text:	Mostrar texto dinámico 
ui_text_input:	Campo de texto para ingreso de datos 

- 7. Nodos Especiales / Otros

Complementan funciones de control, depuración y lógica de flujo.

Nodo	Descripción
Debug:	Muestra mensajes en la consola de desarrollo 
Catch:	Captura errores en otros nodos 
Status:	Monitoriza el estado de nodos 
link in/out:	Conecta flujos de forma modular 
Comment:	Añade anotaciones al flujo 

Beneficios de la Integración en Node-RED

- Flexibilidad: Permite conectar una amplia variedad de dispositivos y protocolos de comunicación.
- Eficiencia: Facilita la implementación de flujos de datos optimizados sin necesidad de programación avanzada.
- Escalabilidad: Compatible con arquitecturas IoT en la nube, lo que permite expandir las soluciones fácilmente.

En el próximo epígrafe, se abordarán estrategias avanzadas para la optimización y automatización de flujos en Node-RED.

3.2.3 Integración de Node-RED con MQTT y REST-API

Node-RED se destaca por su capacidad de integrar múltiples protocolos de comunicación, siendo MQTT y REST-API dos de los más utilizados en soluciones IoT. Esta integración permite la interconectividad entre dispositivos, servidores y plataformas en la nube de manera eficiente y escalable. Ver Figura 3.15.

Elementos Claves en la Integración de Node-RED

1. Conectar Dispositivos: Permite establecer conexiones con sensores y actuadores IoT.
2. Integrar Servicios de Nube: Facilita la comunicación con plataformas de almacenamiento y procesamiento de datos en la nube.
3. Habilitar la Comunicación MQTT: Asegura la mensajería ligera y eficiente entre dispositivos mediante un modelo de publicación/suscripción.
4. Facilitar las Llamadas API REST: Permite la interacción basada en HTTP para el control y monitoreo remoto de dispositivos.
5. Lograr la Comunicación Fluida: Garantiza la interoperabilidad y el flujo continuo de datos entre todos los componentes de la red IoT.

Beneficios de Integrar Node-RED con MQTT y REST-API

- Interoperabilidad: Posibilita la comunicación entre diferentes dispositivos y plataformas.
- Escalabilidad: Permite diseñar soluciones flexibles y adaptables a diversas aplicaciones IoT.
- Eficiencia en la Transmisión de Datos: Optimiza el uso del ancho de banda y reduce la latencia en la comunicación.

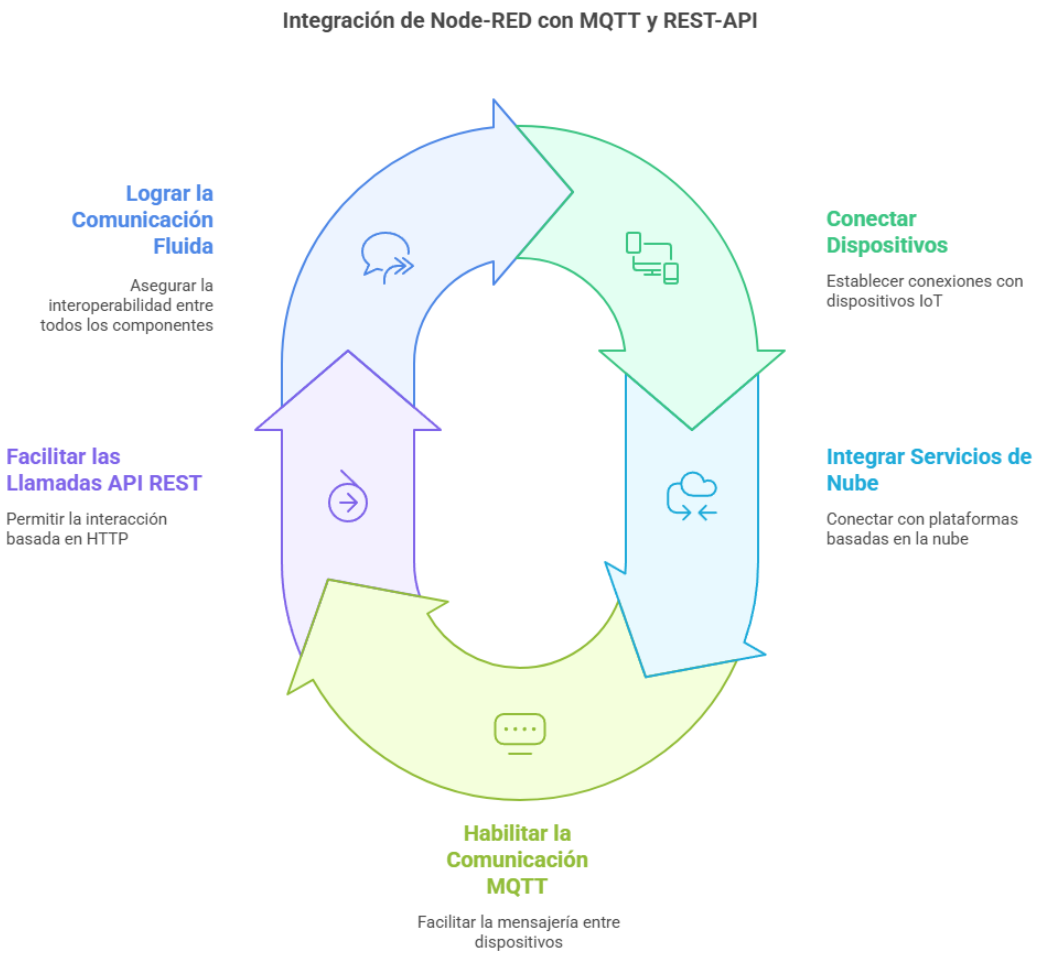


Figura 3.15 Esquema de integración de Node-RED con MQTT y REST-API, mostrando las conexiones clave en un entorno IoT. Elaboración propia.

En la Figura 3.16 aparece un flujo de node-red ejemplo que da solución a la integración en Node-red de los protocolos de comunicación MQTT y REST/API, típico de un gateway IoT. Este flujo permite:

1. **Conectar dispositivos IoT** (entradas y salidas),
2. **Habilitar la comunicación MQTT** entre esos dispositivos,
3. **Integrar servicios en la nube** (por ejemplo, enviar datos a una API externa),
4. **Facilitar llamadas API REST** para recibir comandos o consultar datos desde un cliente externo,
5. **Lograr una comunicación fluida** orquestando todo lo anterior en un flujo interoperable.

A continuación, te describo el flujo, seguido de un esquema visual simplificado en texto. Puedes implementarlo directamente o exportarlo como plantilla:

🔷 1. Entrada de Dispositivo (simulado o real)

[inject] → [function: simularSensor] → [mqtt out]

- inject: genera datos (temperatura, humedad, etc.).
- function: agrega un *msg.payload* en JSON.
- mqtt out: publica en un tópico (ej. *iot/sensor/temp*).

🔶 2. Comunicación MQTT

[mqtt in (iot/sensor/temp)] → [debug]

↓

[function: filtrar o transformar]

↓

[http request (API nube)]

- mqtt in: recibe datos de los sensores.
- function: transforma o filtra según lógica.
- http request: envía a una plataforma como ThingSpeak, Ubidots o una REST API propia.

🔷 3. API REST de Entrada

[http in (GET /estado)] → [function: generarRespuesta] → [http response]

- Permite que un cliente HTTP consulte el estado o datos actuales.

🔶 4. Visualización o Alerta

[mqtt in (iot/sensor/temp)] → [ui_chart / ui_text / ui_gauge]

- Integra el nodo dashboard para mostrar el valor en tiempo real.

🔧 Resumen Modular del Flujo

(1) Dispositivo → MQTT Out → MQTT Broker → MQTT In → HTTP Request (API externa)

→ UI Dashboard

→ REST API (GET /estado)

(2) Cliente REST → HTTP In → Función → HTTP Response

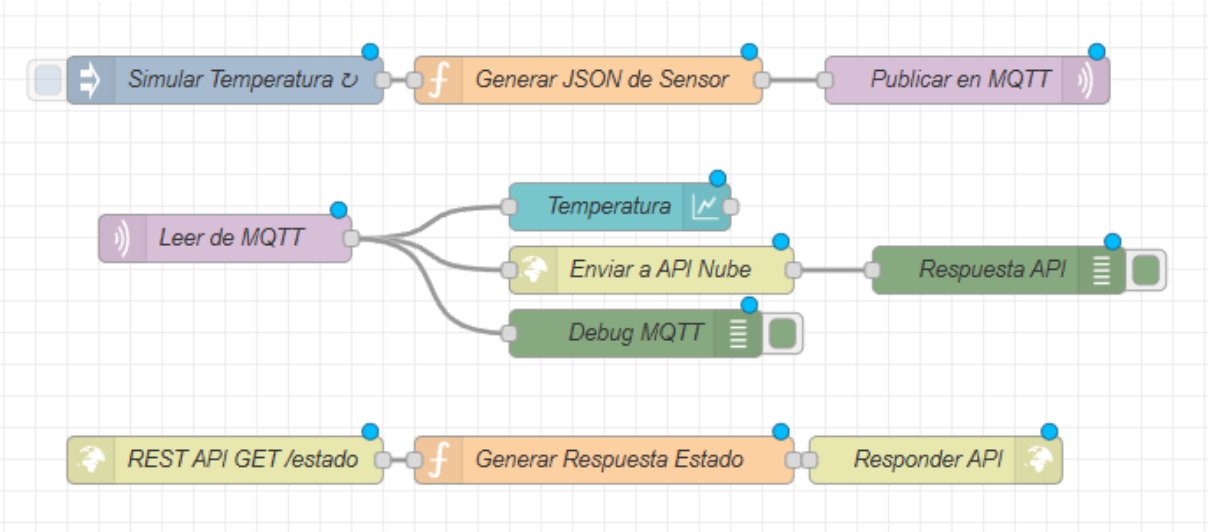


Figura 3.16 Flujo de Node-RED, basado en el esquema de integración de la Figura 3.15. Elaboración propia.

✅ ¿Qué incluye este flujo?

- Simulación de sensor con *inject* y *function*.
- Publicación y recepción vía MQTT (*iot/sensor/temp*).
- Envío a un servicio REST externo.
- Exposición de un endpoint API REST (*/estado*) que puedes consultar desde Postman o un navegador.
- Visualización de temperatura en un dashboard.
- Nodo *debug* para monitorear datos en tiempo real.

3.3 Programación de Dispositivos IoT

En el Capítulo 1 abordamos los fundamentos de IoT, específicamente hablando de arquitecturas IoT, en todas las arquitecturas encontramos en la

capa más baja del stack, la capa de dispositivos, donde los procesadores en diversos formatos dan vida e inteligencia a dicha capa. Esta inteligencia es gracias a que estos procesadores pueden ser programados, es decir, hay un software denominado "firmware" que corre sobre dicho CPU y se encarga de gestionar las entradas y salidas de todos los sensores y actuadores, asi como la comunicación con capas superiores.

El firmware en los procesadores o microcontroladores, mejor dicho, el código que corre sobre el CPU es un código digital de unos y ceros, tambien conocido como código de máquina. A lo largo del tiempo este código ha sido generado utilizando diferentes herramientas de desarrollo (IDE) y diversos lenguajes de programación que han convertido la lógica de programación que da solución a un problema en dicho código de máquina. Entre otros, encontramos lenguajes como: Ensamblador, C, C++, Node.js o micropython, incluso node-red puede correr sobre plataformas más robustas como un Raspberry Pi y ser el lenguaje de programación de un robusto dispositivo IoT. Obviamente, no es la idea, ya que se desea tener en campo (capa de dispositivos) una gran cantidad de dispositivos y por tanto la economía de recursos es vital.

Uno de los lenguajes más usados es el C, gracias a que se dispone de compiladores que traducen el código de C en lenguaje de máquina de forma muy eficiente. Un ejemplo de ello es el compilador de Arduino, que dispone de un arsenal de códigos disponibles para desarrolladoras noveles como profesionales.

3.3.1 Proceso de Desarrollo en C para Microcontroladores

Como se ha mencionado para desarrollar el firmware, requerimos de una plataforma de desarrollo, digamos que entienda el lenguaje C y a su vez, integrar un compilador. Hay muchas plataformas de desarrollo para C, aunque últimamente la plataforma de VSCode con el complemento de PlatformIO han tomado la tutela al respecto. Si necesitamos desarrollar el firmware se requieren los siguientes pasos:

Pasos Claves en el Desarrollo:

1. Iniciar Entorno de Desarrollo: Configurar entornos como VSCode y PlatformIO para facilitar la compilación y carga del código.
2. Escribir Código en C: Desarrollar algoritmos y funciones que interactúen con el hardware del microcontrolador.
3. Compilar Código: Verificar y depurar el código para asegurar que esté libre de errores antes de su ejecución.

4. Cargar Código en el Microcontrolador: Transferir el código compilado al dispositivo para su ejecución.

Beneficios de Programar en C para Microcontroladores, ver Figura 3.17:

- Eficiencia y Control: Permite un uso optimizado de la memoria y los recursos del hardware.
- Compatibilidad Extensa: Compatible con una gran variedad de microcontroladores.
- Flexibilidad en el Desarrollo: Facilita la integración con periféricos y sensores en entornos IoT.

Figura 3.17 Flujo del desarrollo de software en C para microcontroladores, desde la configuración del entorno hasta la carga del código en el dispositivo. Elaboración propia.

Figura 3.17: Flujo del desarrollo de software en C para microcontroladores, desde la configuración del entorno hasta la carga del código en el dispositivo.

En los siguientes epígrafes, se abordarán estrategias avanzadas para optimizar el desarrollo y depuración del código en sistemas embebidos, incluso sin dominar los lenguajes de programación antes mencionados, incluyendo el C. Este es el caso de ESPHome, que permite hacer una abstracción del lenguaje

de programación, programando el firmware en muy alto nivel, prácticamente configurando ciertos parámetros.

3.3.2 Simplificación de Implementaciones IoT con ESPHome

ESPHome, ESPHome (2023), es una solución diseñada para simplificar la programación y configuración de dispositivos IoT basados en ESP8266 y ESP32, ver Figura 3.18. Se enfoca en la creación de firmware mediante archivos YAML, reduciendo la necesidad de programación en C y facilitando la integración con plataformas como Home Assistant.

Elementos Claves en la Simplificación de IoT con ESPHome

1. Código YAML: Permite definir comportamientos y configuraciones sin necesidad de escribir código en C.
2. Integración con Home Assistant: Facilita la comunicación con esta plataforma para una automatización eficiente.
3. Abstracción de Código en C: Reduce la complejidad en el desarrollo de firmware mediante la automatización de tareas repetitivas.

Beneficios de Usar ESPHome en IoT

- Menor Complejidad: Simplifica el desarrollo al eliminar la necesidad de escribir código en bajo nivel.
- Automatización Eficiente: Facilita la gestión de dispositivos IoT con configuraciones predefinidas.
- Flexibilidad en la Configuración: Permite ajustes rápidos y sencillos a través de YAML.

Figura 3.18 Esquema de simplificación de IoT con ESPHome, mostrando sus principales elementos y su integración con Home Assistant. Elaboración propia.

Figura 3.18: Esquema de simplificación de IoT con ESPHome, mostrando sus principales elementos y su integración con Home Assistant.

Elección del Compilador en ESPHome

ESPHome permite compilar el firmware utilizando diferentes entornos, siendo los dos más comunes los que aparecen en la Figura 3.19:

- Compilador de Arduino: Ofrece mayor compatibilidad y facilidad de uso para principiantes.
- Compilador ESP-IDF: Proporciona mayor optimización y acceso a características avanzadas del hardware.

Figura 3.19 Comparación entre los compiladores de Arduino y ESP-IDF para la implementación de ESPHome. Elaboración propia.

3.3.3 Casos de Estudio en IoT

El Internet de las Cosas (IoT) ha transformado múltiples sectores, permitiendo la optimización de procesos, el análisis de datos en tiempo real y la automatización de tareas. En este epígrafe, se presentan estudios de caso que ilustran la aplicación de IoT en distintos ámbitos.

IoT en la Atención Médica

La implementación de IoT en el sector salud mostrado en la Figura 3.20, ha permitido mejorar la atención médica mediante dispositivos de monitoreo remoto, integración de sistemas hospitalarios y optimización de la gestión de datos. Sin embargo, enfrenta desafíos relacionados con la privacidad de los pacientes y la interoperabilidad de los sistemas.

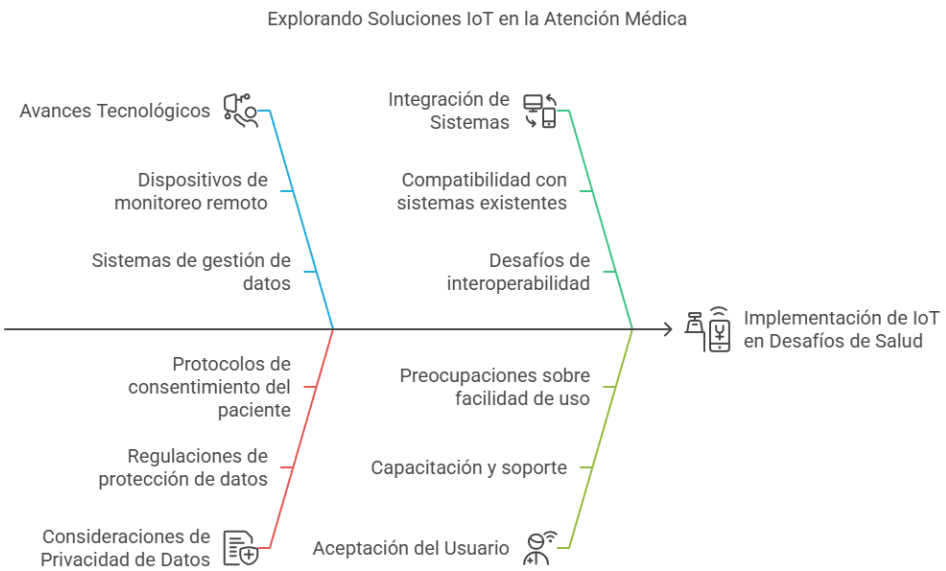


Figura 3.20 Representación de los principales factores que influyen en la implementación de IoT en el sector salud, destacando avances tecnológicos y desafíos de privacidad y aceptación del usuario. Elaboración propia.

IoT en Agricultura

El uso de sensores IoT en la agricultura ha optimizado la recolección de datos meteorológicos y del suelo, mejorando la toma de decisiones basada en datos como se indica en la Figura 3.21. Conjuntamente, ha permitido la optimización del uso de agua y fertilizantes, promoviendo la sostenibilidad y la reducción del desperdicio.

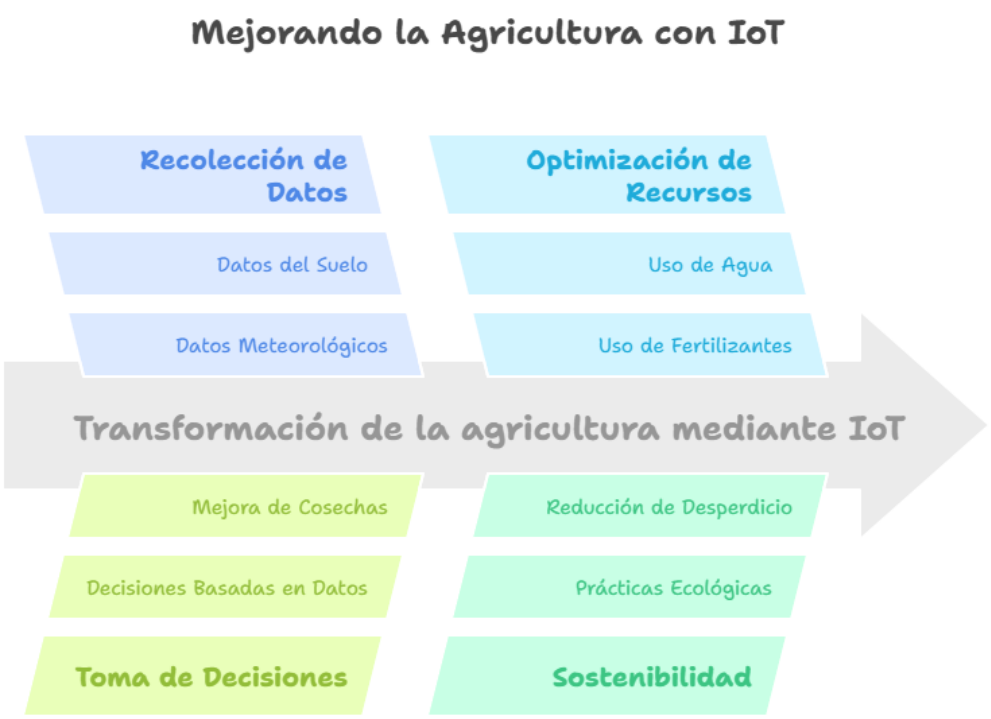


Figura 3.21 Diagrama de la transformación de la agricultura mediante IoT, destacando la optimización de recursos y la toma de decisiones basada en datos. Elaboración propia.

IoT en Domótica

Los sistemas de automatización del hogar basados en IoT, mostrados en la Figura 3.22, han permitido conectar dispositivos inteligentes, mejorar la eficiencia energética y aumentar la comodidad del usuario. El uso de plataformas como Home Assistant facilita la integración y el control remoto de dispositivos conectados.

Figura 3.22 Esquema del ciclo de interacción de IoT en domótica, mostrando cómo los dispositivos inteligentes mejoran la eficiencia y automatizan procesos. Elaboración propia.

IoT en Manufactura

En el ámbito industrial, IoT ha transformado los procesos de producción mediante la interconexión de dispositivos, el análisis de datos en tiempo real y la optimización del mantenimiento predictivo. Esto ha reducido costos operativos y aumentado la competitividad en el mercado. La Figura 3.23 muestra la cadena de transformación en la industria con el uso del IoT.

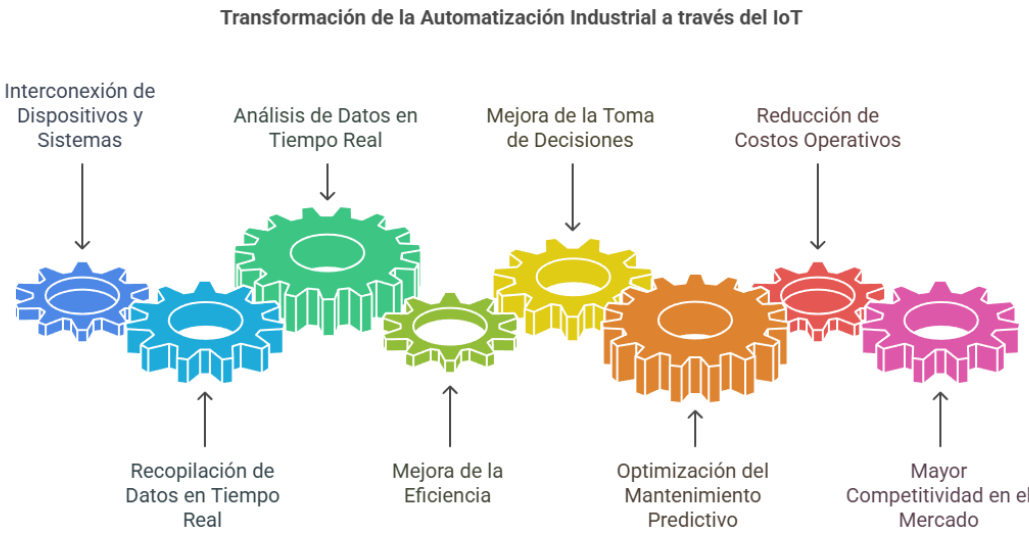


Figura 3.23 Representación de la transformación de la automatización industrial a través de IoT, destacando la interconexión de dispositivos y la mejora de la toma de decisiones. Elaboración propia.

Figura 3.23: Representación de la transformación de la automatización industrial a través de IoT, destacando la interconexión de dispositivos y la mejora de la toma de decisiones.

IoT en Ciudades Inteligentes

Las ciudades inteligentes han integrado IoT para optimizar el tráfico, mejorar los servicios públicos y aumentar la seguridad. Sensores conectados permiten monitorear variables ambientales, gestionar la movilidad urbana y mejorar la calidad de vida de los ciudadanos.

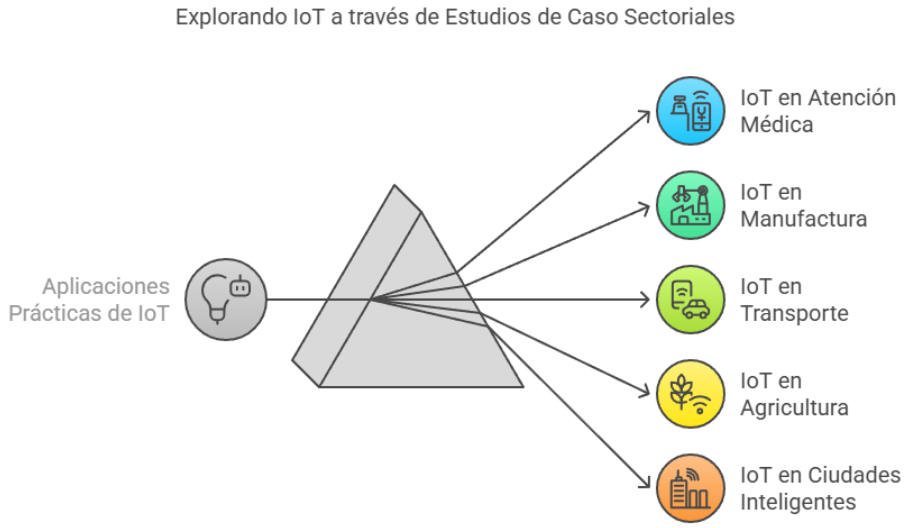


Figura 3.24 Esquema de la exploración de IoT a través de estudios de caso sectoriales, mostrando su aplicación en salud, manufactura, transporte, agricultura y ciudades inteligentes. Elaboración propia.

En conclusión, IoT ha revolucionado múltiples sectores mostrados en la Figura 3.24, facilitando la toma de decisiones basada en datos, mejorando la eficiencia operativa y promoviendo la automatización de procesos. Cada sector enfrenta desafíos específicos, pero los beneficios de IoT continúan impulsando su adopción a nivel global.

Consejo Pedagógico:
Tip: Como aplicar IoT en diferentes dominios de aplicación:

- Salud: Sensores de monitoreo de signos vitales.
- Agricultura: Sensores de humedad del suelo con control de riego.
- Domótica: Automatización de persianas y luces basadas en horarios y sensores de presencia.
- Industria: Sistemas de mantenimiento predictivo usando acelerómetros.

Elementos de Cierre

Resumen del Capítulo

Este capítulo abordó las plataformas IoT locales y en la nube, el desarrollo de flujos de automatización con Node-RED, y la programación de dispositivos

usando C y ESPHome, culminando en ejemplos prácticos aplicados a diferentes sectores.

Preguntas de Revisión

❓ Opción múltiple:
¿Cuál de las siguientes plataformas es local y facilita la automatización sin necesidad de conexión a Internet?
a) AWS IoT
b) Google Cloud IoT
c) Home Assistant
d) Azure IoT

✅ Verdadero/Falso:
Node-RED requiere habilidades avanzadas de programación para su uso básico.
(Verdadero / Falso)

✍️ Respuesta corta:
Menciona dos ventajas de usar ESPHome para desarrollar dispositivos IoT.

Clave de Respuestas

- ❓ Opción múltiple: c) Home Assistant
- ✅ Verdadero/Falso: Falso (Node-RED se basa en programación visual de flujos).
- ✍️ Respuesta corta: Configuración sencilla usando YAML; integración nativa con Home Assistant.

Referencias bibliográficas.

Azure IoT. (2025, 10 de mayo). *Soluciones de Internet de las cosas (IoT)*. Recuperado el 10 de mayo de 2025, de https://azure.microsoft.com/es-es/solutions/iot

AWS IoT. (2025, 10 de mayo). *Servicios de Internet de las cosas (IoT)*. Recuperado el 10 de mayo de 2025, de https://aws.amazon.com/es/iot/

Blynk. (2025, 10 de mayo). *Internet of Things platform for businesses and developers*. Recuperado el 10 de mayo de 2025, de https://blynk.io/

ESPHome. (2023). ESPHome Documentation. Recuperado el 10 de mayo de 2025, de https://esphome.io/

Google Cloud. (2025, 10 de mayo). *IoT platform product architecture*. https://cloud.google.com/architecture/connected-devices/iot-platform-product-architecture?hl=es-419

Home Assistant. (2025, 10 de mayo). *Open source home automation that puts local control and privacy first*. https://www.home-assistant.io/

Node-RED. (2025, 10 de mayo). Node-RED Official Documentation. https://nodered.org/

openHAB. (2025, 10 de mayo). *Empowering the smart home*. Recuperado el 10 de mayo de 2025, de https://www.openhab.org/

4 Glosario

Término	Definición
AIoT	Artificial Intelligence of Things: integración de inteligencia artificial con IoT para dotar de autonomía y aprendizaje a los dispositivos conectados.
AWS	Amazon Web Services: plataforma de servicios en la nube que incluye soluciones para conectar, gestionar y analizar dispositivos IoT.
Actuador	Dispositivo que convierte señales eléctricas en acciones físicas.
Arduino	Plataforma de desarrollo abierta para la creación de proyectos interactivos.
Broker MQTT	Servidor intermediario en sistemas de mensajería MQTT.
Compilador	Programa que traduce código fuente a lenguaje máquina ejecutable.
Convertidor ADC	Componente que convierte señales analógicas a digitales para procesamiento.
Domótica	Aplicación de tecnologías IoT para la automatización de viviendas, mejorando confort, eficiencia y seguridad.
ESP-IDF	Framework oficial de desarrollo para microcontroladores ESP32.
ESPHome	Framework que permite definir firmware para dispositivos IoT mediante YAML.
Edge Computing	Procesamiento de datos cercano al origen para reducir latencia.

Firmware	Software embebido en hardware para controlar sus operaciones.
Fog Computing	Modelo de computación distribuida entre la nube y el borde de la red.
HTTP	HyperText Transfer Protocol: protocolo de comunicación que permite la transferencia de información en la web.
Home Assistant	Plataforma de código abierto para automatización del hogar.
IIoT	Industrial Internet of Things: aplicación del IoT en entornos industriales para mejorar la eficiencia y productividad.
Interfaz	Punto de conexión entre dos sistemas para el intercambio de información.
Interoperabilidad	Capacidad de distintos sistemas para comunicarse y trabajar en conjunto.
IoE	Internet of Everything: concepto que amplía el IoT incorporando personas, procesos, datos y cosas.
IoT	Red de objetos físicos conectados que recopilan y comparten datos.
JSON	JavaScript Object Notation: formato de texto ligero para el intercambio de datos, muy usado en APIs REST.
MQTT	Protocolo de mensajería ligero basado en publicación/suscripción.
Microcontrolador	Circuito integrado con CPU, memoria y periféricos para controlar sistemas embebidos.

Microprocesador	Unidad central de procesamiento integrada en computadoras y SBC.
Mote	Nodo sensor o dispositivo inalámbrico autónomo que integra sensores, microcontrolador y conectividad en redes IoT.
Node-RED	Herramienta de programación visual para integración de flujos de datos.
Nube	Infraestructura remota para almacenamiento y procesamiento de datos.
Placa de desarrollo	Tarjeta electrónica utilizada para diseñar y probar circuitos.
Plataforma IoT	Infraestructura para gestionar, visualizar y procesar datos de dispositivos IoT.
Platformio	Entorno de desarrollo moderno y multiplataforma para programación de sistemas embebidos.
Protocolo de comunicación	Conjunto de reglas que definen cómo se transmiten los datos.
REST-API	Interfaz basada en HTTP que permite acceder y controlar recursos web.
RTOS	Sistema operativo de tiempo real para dispositivos con requerimientos críticos.
SBC	Single Board Computer: computadora de placa única con todos los componentes necesarios integrados en una sola tarjeta.
SOA	Service-Oriented Architecture: estilo de arquitectura que organiza el software como una colección de servicios interoperables.
Sensor	Dispositivo que detecta condiciones del entorno y genera señales eléctricas.

Servidor	Sistema que proporciona servicios o datos a otros dispositivos en la red.
Sistema embebido	Sistema computacional dedicado a una función específica dentro de un dispositivo mayor.
Smart City	Ciudad inteligente que utiliza tecnologías digitales e IoT para mejorar los servicios urbanos y la calidad de vida ciudadana.
SoC	System on Chip: circuito integrado que incluye procesador, memoria y periféricos en un solo chip.
VSCode	Editor de código extensible ampliamente usado en desarrollo de software embebido.
WebSocket	Protocolo que permite comunicación bidireccional en tiempo real entre cliente y servidor a través de una única conexión TCP.
WoT	Web of Things: evolución del IoT que permite que los dispositivos se integren a través de la web mediante estándares abiertos.
XML	Extensible Markup Language: lenguaje de marcado flexible para estructurar datos de forma jerárquica.
YAML	Lenguaje de marcado legible por humanos utilizado en configuración de sistemas.

Ing. Dixys Leonardo Hernández Rojas, PhD.
ORCID: https://orcid.org/0000-0002-2116-6531
Email: dhernandez@utmachala.edu.ec
Ingeniero electrónico y Máster en Electrónica por la Universidad Central de las Villas - Cuba. Obtuvo su PhD en Tecnología de la Información de las Comunicaciones Radio Móviles (TIC-RM) interuniversitario, otorgado por 6 universidades de España, Universidad de la Coruña, Zaragoza, País vasco, entre otras. Con 33 años de experiencia laboral, ha trabajado en la docencia universitaria de pregrado y postgrado y en la industria en varios países como Cuba, México, USA, Colombia, Chile y Ecuador, desarrollando proyectos y p roductos con sistemas embebidos full stack, telecomunicaciones y automatización industrial. Su línea de investigación actual es el Internet de las Cosas, Redes de sensores inteligentes, Redes inalámbricas con BLE, Zigbee, LoRA, Domótica, Realidad aumentada, Blockchain y Seguridad IoT. Ha dictado conferencias en numerosos congresos nacionales e internacionales en varios países. Ha publicado varios libros, ponencias y artículos científicos, muchos de ellos en revistas de alto impacto. Ha dirigido varios proyectos de investigación, fundó el Grupo IotMach y actualmente dirige el grupo de investigación AutoMathTIC y el semillero estudiantil Redes de Sensores inalámbricos de la UTMACH, dictando también la materia de Internet de las cosas en la carrera de pregrado de Tecnología de la Información en la UTMACH desde el 2021. Realizó su tesis de PhD en el ámbito de IoT, en esta área ha publicado 15 artículos desde 2015 a la fecha, 3 de ellos en revistas alto impacto Q1/Q2, un (1) capítulo de libro y ha dirigido cuatro (4) proyectos de investigación universitaria.

ISBN: 978-9942-53-020-2

Compás
capacitación e investigación