

INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL
INTERDISCIPLINARIA EN
INGENIERÍA Y TECNOLOGÍAS
AVANZADAS

UPIITA

*“Arquitectura IoT para el Monitoreo
y Detección de Anomalías en
el Consumo Eléctrico Residencial”*

Que para obtener el título de
“Ingeniero en Telemática”

Presenta el alumno:

Cordero Montes de Oca Luis Alberto

Huerta Trujillo Iliac

Villordo Jimenez Iclia

*Agradecimientos *Resumen

*Abstract

Introducción Contexto y motivación Planteamiento del problema Objetivos Objetivo general Objetivos específicos Alcance y limitaciones Estructura del documento

Estado del Arte Arquitecturas IoT Las arquitecturas de Internet de las Cosas (IoT) proporcionan marcos de referencia para estructurar sistemas distribuidos que abarcan dispositivos físicos, comunicación, procesamiento y aplicaciones. Su correcta selección e implementación es de gran importancia para asegurar requisitos no funcionales como escalabilidad, resiliencia y seguridad, especialmente en escenarios de telemetría energética y detección de consumos anómalos en tiempo real.

Modelos de referencia y estandarización Existen diversos modelos de referencia que guían el diseño de soluciones IoT, cada uno con énfasis distinto en dominios industriales, interoperabilidad o gobernanza:

- **IoT-A (The IoT Architectural Reference Model)**: Proporciona abstracciones de alto nivel (dominio, información, comunicación, funcional, seguridad) y un metamodelo que promueve interoperabilidad semántica entre objetos y servicios [6]. Es útil para mantener consistencia conceptual en sistemas heterogéneos.
- **IIRA (Industrial Internet Reference Architecture)** del Industrial Internet Consortium: Estructura el sistema en *viewpoints* (business, usage, functional, implementation) y define dominios funcionales (control, operaciones, información, aplicación) con foco en integración OT/IT y requisitos industriales (seguridad, confiabilidad, disponibilidad) [5]. Adecuado para plantas industriales y energía.
- **RAMI 4.0 (Reference Architectural Model Industry 4.0)**: Modelo tridimensional que alinea capas (del activo físico a la comunicación y la integración), jerarquía de la ISA-95 y ciclo de vida del producto [1]. Útil para trazabilidad, gemelos digitales y conformidad en manufactura/energía.
- **NIST IoT-Reference**: Define funciones nucleares (sensing, communication, data processing, applications, y seguridad transversal) con énfasis en gestión de riesgos y conformidad [13], [30].
- **Arquitectura 5C (Connection, Conversion, Cyber, Cognition, Configuration)**: Centrada en manufactura inteligente y mantenimiento predictivo, introduce una progresión desde adquisición hasta realimentación autónoma [17].

Para proyectos de monitoreo energético residencial o industrial ligero, IoT-A y NIST facilitan la interoperabilidad y la gestión del riesgo; para entornos industriales con integración de SCADA/PLC, IIRA y RAMI 4.0 aportan patrones robustos y vocabularios alineados a estándares.

Capas funcionales y patrón Edge–Fog–Cloud Un patrón ampliamente adoptado organiza la solución en capas: *device/edge*, *fog/gateway* y *cloud*, lo que permite distribuir cómputo, almacenamiento y control según latencia y costo [22], [25].

1. **Capa de Dispositivo/Percepción**: Sensores/actuadores, medición de potencia/energía, muestreo, preprocesamiento ligero (filtro, compresión, extracción de rasgos básicos). Requisitos: bajo consumo, fiabilidad, sincronización temporal.
2. **Capa de Borde (Edge/Fog/Gateway)**: Agregación de flujos, normalización, seguridad (terminación TLS), control de calidad de servicio, inferencia de baja latencia

(detección temprana de anomalías) y *store-and-forward*. Protocolos típicos: MQTT, CoAP, OPC UA [18], [21], [24].

3. **Capa de Nube:** Ingesta a gran escala, almacenamiento *time-series*, procesamiento por lotes/streaming, entrenamiento de modelos y visualización. Aquí se prioriza elasticidad, gestión de datos históricos y analítica avanzada.

Este desacoplamiento reduce latencia, delimita dominios de fallo y permite optimizar costos (cómputo intensivo en la nube; decisiones inmediatas en el borde). En detección de consumos anómalos, ejecutar reglas de primer nivel (umbrales, z-scores, modelos ligeros) en el edge mitiga eventos falsos y mejora tiempos de reacción, mientras que la nube refina detecciones con modelos estacionales o de *deep learning*.

Planos transversales: seguridad, gestión y observabilidad Independientemente del particionado en capas, tres planos transversales sostienen la operación:

- **Seguridad** (*security by design*): identidad del dispositivo (X.509), autenticación/autorización por tópicos MQTT, cifrado extremo a extremo (TLS 1.2+), rotación de credenciales y *secure boot* [13], [21]. El principio de privilegio mínimo debe reflejarse en ACLs y particionamiento de redes.
- **Gestión del ciclo de vida** (device management): provisión, configuración remota, actualización OTA, inventario y retirada. Estándares como LwM2M/OMA facilitan interoperabilidad [27].
- **Observabilidad**: métricas, logs y trazas distribuidas desde edge a cloud; retención diferenciada para series temporales; paneles de control y alertamiento con umbrales adaptativos.

Mensajería y acoplamiento Los buses de mensajería desacoplados permiten elasticidad y resiliencia. **MQTT** es dominante por su ligereza y soporte de QoS, **CoAP** para dispositivos extremadamente restringidos, y **OPC UA** en entornos industriales que requieren modelado semántico y seguridad integrada [18], [21], [24]. La selección debe considerar:

1. *Patrones de comunicación*: publish/subscribe vs. request/response.
2. *QoS y latencia*: QoS 1/2 para eventos críticos; retención para últimas lecturas; *backpressure*.
3. *Esquemas de tópicos y contratos*: nombres jerárquicos ('site/area/device/metric') y formatos auto-descriptivos (JSON/CBOR) con versión.

Datos y analítica para detección de anomalías Para consumos anómalos, el diseño de datos es determinante: granularidad temporal, agregaciones por ventana, manejo de estacionalidad, metadatos de dispositivo y normalización. En el borde, modelos ligeros (EWMA, ARIMA básico, bosque aleatorio compacto) pueden filtrar anomalías evidentes; en la nube, enfoques más pesados (LSTM, autoencoders, Prophet) capturan estacionalidad diaria/semanal [16], [19], [28]. Es recomendable separar:

- *Plano de características* (extracción/ingeniería de variables).
- *Plano de inferencia* (serving de modelos con control de versiones).
- *Plano de explicación* (importancias, SHAP, reglas).

Buenas prácticas y decisiones de diseño

- **Criterio de colocación:** colocar lógica donde minimice latencia y costo y maximice confiabilidad (p. ej., cortes de red → lógica esencial en edge).
- **Desacoplar por contratos:** usar contratos versionados de mensajes y pruebas de compatibilidad.
- **Seguridad desde el inicio:** identidades por dispositivo, TLS obligatorio, ACL por tópico, rotación periódica.
- **Observabilidad end-to-end:** trazabilidad desde sensor hasta alerta; KPIs por etapa.

Protocolos y mensajería La selección de protocolos de mensajería en IoT condiciona latencia, consumo energético, confiabilidad y seguridad del sistema. En monitoreo de consumo eléctrico y detección de anomalías, predominan protocolos ligeros orientados a telemetría (**MQTT**, **CoAP**) junto con **HTTP/REST** y **WebSockets** para integración de aplicaciones, y **OPC UA** en escenarios industriales con fuerte semántica y requisitos de conformidad [7], [18], [21], [24].

MQTT: telemetría ligera con publish/subscribe **MQTT** es un protocolo de mensajería ligero sobre TCP con patrón *publish/subscribe* y un *broker* central (por ejemplo, Mosquitto, EMQX). Ofrece niveles de calidad de servicio (QoS 0, 1 y 2), retención de mensajes y sesiones persistentes, lo que lo hace idóneo para enlaces inestables y dispositivos con recursos limitados [21].

- *Topologías y contratos:* la semántica de tópicos suele ser jerárquica (`site/area/device/metric`) y conviene versionar (`v1/`) para evitar acoplamiento rígido.
- *QoS:* QoS 1/2 reducen pérdida de datos a costa de mayor latencia y tráfico; en energía, métricas críticas (alarmas) suelen usar QoS 1 y lecturas de alta frecuencia, QoS 0.
- *Retained & LWT:* el mensaje retenido entrega el último estado a suscriptores tardíos; el *Last Will and Testament* notifica desconexiones inesperadas para mejorar observabilidad.
- *Seguridad:* TLS 1.2+ con autenticación por certificados X.509 y ACL por tópico en el broker; para clientes muy restringidos, credenciales rotadas y políticas de mínimo privilegio [13].

Variantes como **MQTT-SN** operan sobre UDP y reducen cabeceras para redes muy restringidas (LPWAN).

CoAP: request/response ligero sobre UDP **CoAP** implementa un estilo *RESTful* para dispositivos restringidos sobre UDP, con confirmación opcional (CON/NON), *observe* para notificaciones tipo *server push* y soporte de **DTLS** para seguridad [3], [24]. Su modelo *resource-oriented* y payloads compactos (p. ej., CBOR) son útiles en redes LLN (6LoWPAN).

- *Modos confiables:* mensajes confirmables (CON) ofrecen fiabilidad similar a QoS 1; *block-wise transfer* soporta cargas mayores.

- **Seguridad:** DTLS 1.3 o OSCORE para cifrado extremo a extremo a nivel de objeto en entornos con proxies.

HTTP/REST y WebSockets: integración y streaming a aplicaciones **HTTP/REST** sigue dominando la integración con sistemas TI y paneles de control por su ubicuidad y semántica clara [7]. Para telemetría en tiempo casi real a dashboards, **WebSockets** permiten canal bidireccional y estable reduciendo la sobrecarga de sondeo.

- **Patrones:** IoT ingiere vía MQTT/CoAP y expone APIs REST para consulta histórica y gestión; WebSockets para actualizaciones en vivo.
- **Seguridad:** TLS obligatorio (HTTPS/WSS), OAuth2/OpenID Connect para autenticación de usuarios y *API keys* o mTLS para *machine-to-machine*.

OPC UA: semántica y seguridad integradas en OT **OPC UA** combina modelo de información rico, servicios de descubrimiento, suscripción y seguridad integrada (cifrado, firma, políticas de usuario) [18]. En energía industrial, facilita interoperabilidad con PLC/SCADA y modelado semántico de activos. Para dispositivos muy restringidos, su costo puede ser elevado; es común integrarlo en gateway/edge que publica hacia la nube vía MQTT (puente MQTT–OPC UA).

Formatos de payload y serialización La carga útil debe equilibrar compacidad, legibilidad y costos de serialización:

- **JSON:** legible y ampliamente soportado; mayor tamaño. Útil para integración rápida.
- **CBOR:** binario compacto y eficiente para dispositivos y CoAP [4].
- **Protocol Buffers:** esquemas versionados y eficientes, ventajoso para pipelines de alto volumen y control de contratos [12].

Se recomienda versionar esquemas, incluir marcas de tiempo normalizadas (UTC, ISO 8601) y metadatos (ID de dispositivo, calidad, unidad) para reproducibilidad y detección de anomalías.

Patrones de diseño de mensajería

- **Topic design:** jerarquías por ubicación/activo/métrica, evitando comodines en producción para limitar exposición y costos.
- **Backpressure:** uso de colas/búferes en edge (*store-and-forward*), límites de *in-flight* y gestión de reconexiones exponenciales.
- **Idempotencia y orden:** claves de deduplicación y *sequencing* para series temporales; en MQTT, no hay garantía de orden global con múltiples publicadores.
- **QoS-aware routing:** enrutar mensajes críticos por canales con QoS superior; agrupar telemetría de baja prioridad en lotes para eficiencia.

Seguridad de transporte y extremos

- **TLS/DTLS:** cifrado en tránsito, PFS y suites modernas (AES-GCM/ChaCha20-Poly1305) [3].

- **mTLS**: autenticación mutua para dispositivos y gateways; gestión de PKI con rotación y revocación.
- **Autorización por tópico/recurso**: ACL por patrón de tópico (MQTT) o ruta (CoAP/HTTP); principio de mínimo privilegio.
- **E2E en entornos con proxy**: OSCORE (CoAP) y cifrado a nivel de payload cuando el proxy debe operar solo con metadatos.

Criterios de selección para consumo energético y anomalías Para detección de consumos anómalos:

- **MQTT** con QoS 1 para eventos/alarmas y QoS 0 para telemetría de alta frecuencia, retenidos para estados; TLS y ACL estrictas en broker.
- **CoAP** con confirmables (CON) y CBOR cuando los dispositivos sean muy restringidos o operen sobre 6LoWPAN/LPWAN.
- **HTTP/REST** para APIs de consulta histórica y administración; **WebSockets** para dashboards en vivo.
- **Puentes OPC UA–MQTT** cuando se integra OT/SCADA con analítica en la nube.

La decisión final debe balancear consumo de ancho de banda, latencia requerida para alertas, capacidad del dispositivo y requisitos de seguridad y conformidad. Plataformas y servicios Las plataformas para IoT integran la cadena de valor desde la ingesta de telemetría hasta la analítica y visualización. En escenarios de consumo energético y detección de anomalías, los componentes clave incluyen: *brokers* de mensajería, *ingestors*, almacenamiento de series temporales, procesamiento de flujos y lotes, orquestación y tableros. La selección debe balancear latencia, costo, elasticidad, gobernanza y seguridad.

Brokers de mensajería y gateways Los **brokers MQTT** (Eclipse Mosquitto, EMQX, HiveMQ) son la columna vertebral de la telemetría [21]. Ofrecen *topic ACLs*, persistencia, retención, *bridges* y plugins para extender autenticación/autorización y observabilidad. **CoAP** suele desplegarse con *proxies* y *resource directories* para descubrimiento [24]. En entornos industriales, **OPC UA** se integra en *gateways* que traducen a MQTT para la nube [18].

- *Criterios*: soporte de TLS/mTLS, control fino de ACL por tópico, cuotas, gestión de sesiones, *bridging multi-tenant*, métricas nativas.
- *Operación*: alta disponibilidad con clúster activo-activo y *retained stores* replicados; backpressure y límites *in-flight*.

Ingesta y canalización La ingesta desacopla dispositivos de servicios de almacenamiento/analítica. Patrones comunes:

- **Puentes MQTT→TSDB/Stream**: conectores nativos o *functions* del broker para enrutar a bases de series de tiempo o buses de streaming.
- **Buses de datos** (Kafka/Pulsar): proporcionan *durability*, particionado y *replay* para flujos de alta tasa [9], [10]. Útiles cuando múltiples consumidores (detección online, archivado, aprendizaje) comparten el mismo *topic*.

- **ETL/ELT**: normalización, validación de esquemas (JSON Schema/Protobuf), enriquecimiento (metadatos de dispositivo) y control de versiones de contratos.

Almacenamiento de series temporales (TSDB) Las **TSDB** optimizan escrituras en ráfaga, compresión y *downsampling* con retención por políticas. Alternativas populares incluyen **InfluxDB**, **TimescaleDB** y **Prometheus** (sobre todo para métricas operativas) [2], [14], [29]. Para consumo energético:

- *Requisitos*: escrituras sostenidas, agregaciones por ventana, alineación temporal, etiquetado por `device/site/metric`, y consultas por rango.
- *Esquemas*: diseño basado en *measurement + tags* (InfluxDB) o tablas particionadas por tiempo (TimescaleDB); compresión *delta-of-delta* y *gorilla* para eficiencia.
- *Retención*: políticas multi-tier (crudo por días, agregados por meses) para optimizar costos.

Criterio	InfluxDB	TimescaleDB	Prometheus
Modelo de datos	Measurements + tags/fields; series identificadas por tags	PostgreSQL + extensiones; hypertables/particiones por tiempo	Series etiquetadas (metrics + labels); enfoque en métricas operativas
Ingesta	Alta tasa nativa; Line Protocol; conectores MQTT/Kafka	Ingesta vía Pull (scrape) y SQL/FDW/ingestor/push vía gateway; buena integración con ecosistema SQL	
Consultas	Flux/SQL (según edición); agregaciones y down-sampling	SQL estándar + funciones de time-series; facilidad de joins	PromQL; consultas operativas para monitoreo/alertas
Compresión	Delta-of-delta, Gorilla, TSM/TSI	Compresión de PostgreSQL + compresión por chunks	Compresión específica por bloques en TSDB
Retención [H]	Políticas por bucket; downsampling automático	Políticas por partición/retención; administración vía SQL	Retención por tiempo/espacio; pensado para métricas
Escalabilidad	OSS single-node; Enterprise/Cloud con clustering	Escala vertical + particionado; replicación nativa de Postgres	Federación y sharding vía proyectos complementarios
Casos de uso típicos	Telemetría IoT, sensores, analítica temporal general	IoT + analítica relacional; integración con datos de negocio	Observabilidad (métricas app/infra), alertas, paneles
Licencia	OSS + ofertas comerciales (Cloud/Enterprise)	OSS (Timescale License para algunas features)	OSS (Apache 2.0)
Ecosistema	Conectores a Grafana, Telegraf, Kapacitor	Ecosistema Postgres, extensiones, Grafana	Stack de observabilidad, Grafana, Alertmanager
Adecuación a consumo energético	Muy buena para series densas y downsampling	Excelente si necesita joins con datos de negocio y SQL	Útil para métricas operativas; menos óptimo para series crudas densas

Procesamiento de flujos y lotes La detección de anomalías requiere tanto **stream processing** (detección temprana) como **batch** (modelado y recalibración):

- **Stream:** Apache Flink, Spark Structured Streaming, Kafka Streams permiten ventanas deslizantes/tumbling, *watermarks* y estado gestionado para reglas y modelos online [8], [26].
- **Batch:** Spark/Beam para entrenamiento y análisis histórico; orquestado por Airflow/Luigi con *feature stores*.
- **Model serving:** *microservicios* con REST/gRPC o *serverless* para inferencia; control de versiones de modelos y *shadow deployments*.

Orquestación y contenedores La estandarización de despliegues facilita portabilidad del borde a la nube:

- **Contenedores** (Docker/OCI) y **Kubernetes** para elasticidad y gestión declarativa [11]. En el borde, **K3s** y **Azure IoT Edge/AWS IoT Greengrass** facilitan módulos, *twin state* y despliegues OTA [20], [23].
- **IaC:** Terraform/Ansible para reproducibilidad; *secrets management* (Vault/KMS) y políticas de seguridad.

Dashboards, APIs y alertamiento La plataforma debe exponer:

- **APIs** REST/GraphQL para consulta histórica y administración; **WebSockets** para actualizaciones en vivo.
- **Dashboards** (Grafana, Kibana, superset) con paneles por dispositivo, métricas agregadas y eventos [15].
- **Alertas:** umbrales dinámicos, detecciones por modelo y correlación de eventos; integración con email, webhooks y mensajería.

Seguridad, gobernanza y cumplimiento Las plataformas deben incorporar *security-by-default* y gobernanza de datos:

- **Identidad y acceso:** mTLS para dispositivos/gateways; IAM centralizado; ACL por tópico/recurso; rotación de credenciales y revocación.
- **Data governance:** clasificación de datos, *data lineage*, retención y anonimización donde aplique.
- **Cumplimiento:** lineamientos NIST para capacidad base de ciberseguridad en IoT [13].

Criterios de selección para IoT energético

- **Broker** con TLS/mTLS, ACL por tópico, sesiones persistentes y métricas exportables.
- **TSDB** con retención por niveles y consultas eficientes por *tags*.
- **Stream processor** con ventanas y estado para reglas de primer nivel; **batch** para recalibración de modelos.
- **Dashboards** con soportes de anotaciones y correlaciones; APIs bien definidas para integración externa.

Trabajos relacionados

Marco Teórico Fundamentos de medición y energía Protocolos IoT y seguridad Modelos y técnicas de detección de anomalías

Análisis del Sistema Requerimientos funcionales Requerimientos no funcionales Casos de uso y actores Riesgos y supuestos

Diseño del Sistema Arquitectura propuesta Diseño de datos y tópicos Diseño de servicios e interfaces Diagramas de componentes y secuencia

Implementación del Sistema Entorno y herramientas Módulos implementados Configuración y despliegue Ejemplos de código

Pruebas y Resultados Plan y metodología de pruebas Pruebas funcionales e integración Rendimiento y escalabilidad Resultados y discusión

Conclusiones y Trabajo Futuro Conclusiones Limitaciones Líneas de trabajo futuro

Guía de despliegue Datasets e instrumentación Documentación adicional

*Referencias

- [1] P. I. 4.0, *Reference Architectural Model Industrie 4.0 (RAMI 4.0)*, Accessed 2025-11-06, 2015. dirección: <https://www.plattform-i40.de/>
- [2] T. P. Authors, *Prometheus — Monitoring system & time series database*, Accessed 2025-11-06, 2024. dirección: <https://prometheus.io/>
- [3] D. Benjamin et al., *The Datagram Transport Layer Security (DTLS) Protocol Version 1.3, RFC 9147*, IETF RFC, Accessed 2025-11-06, 2022. dirección: <https://www.rfc-editor.org/rfc/rfc9147>
- [4] C. Bormann y P. Hoffman, *CBOR — Concise Binary Object Representation, RFC 8949*, IETF RFC, Accessed 2025-11-06, 2020. dirección: <https://www.rfc-editor.org/rfc/rfc8949>
- [5] I. I. Consortium, *The Industrial Internet of Things, Volume G1: Reference Architecture v1.9*, Accessed 2025-11-06, 2019. dirección: <https://www.iiconsortium.org/IIRA.htm>
- [6] T. I.-A. Consortium, *Internet of Things - Architectural Reference Model (IoT-A ARM)*, Project Deliverables, Accessed 2025-11-06, 2013. dirección: <https://www.riot-a.eu/>
- [7] R. T. Fielding, “Architectural Styles and the Design of Network-based Software Architectures”, en *Doctoral dissertation, University of California, Irvine*, 2000. dirección: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- [8] A. S. Foundation, *Apache Flink — Stateful Computations over Data Streams*, Accessed 2025-11-06, 2024. dirección: <https://flink.apache.org/>
- [9] A. S. Foundation, *Apache Pulsar*, Accessed 2025-11-06, 2024. dirección: <https://pulsar.apache.org/>
- [10] A. S. Foundation, *Kafka: A Distributed Streaming Platform*, Accessed 2025-11-06, 2024. dirección: <https://kafka.apache.org/>
- [11] C. N. C. Foundation, *Kubernetes Documentation*, Accessed 2025-11-06, 2024. dirección: <https://kubernetes.io/docs/home/>
- [12] Google, *Protocol Buffers*, Accessed 2025-11-06, 2024. dirección: <https://developers.google.com/protocol-buffers>
- [13] P. A. Grassi, M. Fagan, J. Voas et al., *NISTIR 8259A: Core Device Cybersecurity Capability Baseline for Securable IoT Devices*, Accessed 2025-11-06, 2020. dirección: <https://csrc.nist.gov/publications/detail/nistir/8259a/final>
- [14] InfluxData, *InfluxDB: Open Source Time Series Platform*, Accessed 2025-11-06, 2024. dirección: <https://www.influxdata.com/products/influxdb/>
- [15] G. Labs, *Grafana Documentation*, Accessed 2025-11-06, 2024. dirección: <https://grafana.com/docs/>
- [16] N. Laptev, S. Amizadeh e I. Flint, “Time-series Extreme Event Forecasting with Neural Networks at Uber”, en *ICML 2015 Workshop on Deep Learning*, 2015. dirección: <https://arxiv.org/abs/1710.10917>

- [17] J. Lee, B. Bagheri y H.-A. Kao, “A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems”, *Manufacturing Letters*, vol. 3, págs. 18-23, 2015. DOI: [10.1016/j.mfglet.2014.12.001](https://doi.org/10.1016/j.mfglet.2014.12.001)
- [18] W. Mahnke, S.-H. Leitner y M. Damm, *OPC Unified Architecture*. Springer, 2009. DOI: [10.1007/978-3-540-68899-0](https://doi.org/10.1007/978-3-540-68899-0)
- [19] P. Malhotra, L. Vig, G. Shroff y P. Agarwal, “Long Short Term Memory Networks for Anomaly Detection in Time Series”, en *ESANN*, 2015. dirección: <https://arxiv.org/abs/1607.00148>
- [20] Microsoft, *Azure IoT Edge documentation*, Accessed 2025-11-06, 2024. dirección: <https://learn.microsoft.com/azure/iot-edge/>
- [21] OASIS, *MQTT Version 3.1.1 Plus Errata 01*, Accessed 2025-11-06, 2015. dirección: <https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/>
- [22] M. Satyanarayanan, “The Emergence of Edge Computing”, *Computer*, vol. 50, n.º 1, págs. 30-39, 2017. DOI: [10.1109/MC.2017.9](https://doi.org/10.1109/MC.2017.9)
- [23] A. W. Services, *AWS IoT Greengrass*, Accessed 2025-11-06, 2024. dirección: <https://docs.aws.amazon.com/greengrass/>
- [24] Z. Shelby, K. Hartke y C. Bormann, *The Constrained Application Protocol (CoAP)*, *RFC 7252*, IETF RFC, Accessed 2025-11-06, 2014. dirección: <https://www.rfc-editor.org/rfc/rfc7252>
- [25] W. Shi, J. Cao, Q. Zhang, Y. Li y L. Xu, “Edge Computing: Vision and Challenges”, en *IEEE Internet of Things Journal*, vol. 3, 2016, págs. 637-646. DOI: [10.1109/JIOT.2016.2579198](https://doi.org/10.1109/JIOT.2016.2579198)
- [26] A. Spark, *Structured Streaming Programming Guide*, Accessed 2025-11-06, 2024. dirección: <https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html>
- [27] O. SpecWorks, *Lightweight M2M (LwM2M) Technical Specification*, Accessed 2025-11-06, 2020. dirección: <https://www.omaspecworks.org/what-is-oma-specworks/iot/lightweight-m2m-lwm2m/>
- [28] S. J. Taylor y B. Letham, “Forecasting at scale”, *The American Statistician*, vol. 72, n.º 1, págs. 37-45, 2018. DOI: [10.1080/00031305.2017.1380080](https://doi.org/10.1080/00031305.2017.1380080)
- [29] I. Timescale, *TimescaleDB: Time-series Database for PostgreSQL*, Accessed 2025-11-06, 2024. dirección: <https://www.timescale.com/>
- [30] J. Voas, *NIST SP 800-183: Networks of 'Things'*, Accessed 2025-11-06, 2016. dirección: <https://csrc.nist.gov/publications/detail/sp/800-183/final>