UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENTO DE COMUNICACIONES

ARQUITECTURA DE INTEROPERABILIDAD DE DISPOSITIVOS FÍSICOS PARA EL INTERNET DE LAS COSAS (IOT)

DIANA CECILIA YACCHIREMA VARGAS

PhD Thesis

DIRECTOR: CARLOS E. PALAU

SEPTIEMBRE 2019

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
DEPARTAMENTO DE COMUNICACIONES

ARQUITECTURA DE INTEROPERABILIDAD DE DISPOSITIVOS FÍSICOS PARA EL INTERNET DE LAS COSAS (IOT)

TESIS DOCTORAL

Diana Cecilia Yacchirema Vargas

Director:

Dr. Carlos Enrique Palau Salvador

Valencia, España

Septiembre 2019

Tesis Doctoral presentada por la doctoranda Diana Cecilia Yacchirema Vargas en el Departamento de Comunicaciones de la Universitat Politècnica de València para la obtención del grado de Doctor en Telecomunicaciones.

Título:

Arquitectura de Interoperabilidad de Dispositivos Físicos para el Internet de las Cosas (IoT)

Doctoranda:

Diana Cecilia Yacchirema Vargas (diayac1@doctor.upv.es)

Director:

Dr. Carlos Enrique Palau Salvador (cpalau@dcom.upv.es**)**

Tribunal evaluador:

Dr. Pedro Salvador Rodríguez Hernández

Dr. Francisco José Martínez Zaldívar

Dr. David Griol Barres

Esta tesis doctoral ha sido realizada dentro del grupo de investigación SATRD (Sistemas y Aplicaciones de Tiempo Real Distribuido) como parte de las actividades del proyecto de Investigación INTER-IoT financiado por la Comisión Europea dentro del Programa Marco de Investigación e Innovación Horizonte 2020 en virtud del acuerdo de subvención nº 687283. Asimismo, el presente trabajo ha sido financiado por la Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación SENESCYT - Convocatoria Abierta de Becas (Quito, Ecuador) y por la Universidad Escuela Politécnica Nacional del Ecuador.

El futuro tiene muchos nombres. Para los débiles es lo inalcanzable.
Para los temerosos lo desconocido. Para los valientes es la oportunidad.
~Victor Marie Hugo

A mi esposo y a los hijos que un día tendremos, quienes desde ya son mi vida entera.

A mis padres, Eduardo y Cecilia, a mis hermanos Eduardo y Paola y a mi sobrina Luisa Monserrate, quienes siempre me apoyaron y me han enseñado que solo con el corazón se puede ver bien.

Agradecimientos

Estos años de investigación han sido un viaje de aprendizaje y desafíos, que me han demostrado una vez más que con esfuerzo y perseverancia podemos llegar a donde queramos. Llegado este momento, en primer lugar, debo dar las gracias a *Dios*, por su amoroso cuidado; por darme las fuerzas necesarias, la capacidad y la sabiduría para cumplir esta gran meta. También quiero agradecer a todas las personas que de una u otra forma han contribuido desinteresadamente para la culminación de esta tesis doctoral, y apoyado este anhelo personal.

A mi director *Dr. Carlos Enrique Palau*, mi eterna gratitud, gracias por su paciencia, guía, consejos y apoyo continuo para el desarrollo de la presente tesis doctoral. Muchas gracias por abrirme la puerta a esta experiencia. Y, como dijo Benjamín Franklin, gracias por involucrarme.

Gracias también al *Dr. Benjamín Molina*, por esos breves pero certeros consejos cuando más lo necesitaba. Su inmenso conocimiento y la precisión al responder a todas mis preguntas supieron ser luz en los momentos difíciles, que me ayudó mucho y me animó a seguir adelante, y más que todo, gracias por su don de gente. Nunca olvidaré todo lo que ha hecho por mí.

Al Estado Ecuatoriano y en especialmente a la *Secretaría de Educación Superior, Ciencia, Tecnología e Innovación (SENESCYT)* y a la *Escuela Politécnica Nacional del Ecuador (EPN)* por la confianza que depositaron en mí para la realización del máster y posteriormente de esta tesis doctoral. Gracias por la inversión realizada durante todos estos años para la materialización de este logro que me ha brindado la oportunidad de ampliar mis horizontes, no solamente a nivel profesional, sino también a nivel personal; siendo una de las experiencias más enriquecedoras desde el primer día que elegí la *Universitat Politècnica de València* como una nueva experiencia académica y de vida.

A *Cristian*, mi compañero de vida, aventuras, alegrías y sueños; gracias por su comprensión, esfuerzo y sobre todo por el cariño brindado día a día, sin duda han sido determinantes para culminar este arduo trabajo. Usted es quién me ha

visto luchando durante todos estos años y ha compartido muchas de mis angustias y logros. Gracias Amor. *Dios* nos bendiga siempre.

A mis padres *Eduardo* & *Cecilia*, en cuyo estímulo constante y amor he confiado a lo largo de mis años, gracias por su apoyo incondicional y creencia absoluta en cada una de mis metas propuestas, con ustedes he aprendido a conseguirlas con tesón y ahínco. Este y todos los logros alcanzados a lo largo de mi vida se los debo a ustedes. Mi agradecimiento profundo a mi madre, *Cecilia*, por las lecciones de superación, perseverancia y esperanza y por las enseñanzas de comprensión y perdón que me da cada día.

A mis hermanos *Eduardo* & *Paola*, y a mi querida sobrina, *Luisa Monserrate*, gracias por todo su amor incondicional, ustedes son el mayor impulso que tengo en la vida. Por ustedes todos los días de mi vida son una nueva oportunidad para ser mejor persona y alcanzar nuevas metas.

"El agradecimiento es la memoria del corazón."

Lao Tsé

Abstract

The vision of the Internet of Things (IoT) implies a hyper-connected global ecosystem in which all devices with communication capacity are ubiquitously connected to the Internet. However, to reach the full potential of IoT, it is not enough that the devices are connected to Internet, they also need to communicate and interact with each other. Unfortunately, building a global ecosystem of devices that connect with each other without problems is practically impossible nowadays. The reason is that IoT is constituted by a plethora of heterogeneous devices in terms of the data format and communication components that make it up, such as hardware, technologies and communication protocols. This heterogeneity inevitably leads to the appearance of "vertical silos" that are isolated from the rest of the IoT (e.g., we still need to install 5 applications to interact with 5 different devices due to the incompatibility between these devices), further exacerbated by the fact that billions of next-generation devices will depend on their ability to connect with each other to get the most benefit.

Therefore, the interoperability of devices is one of the main challenges to face in the field of IoT research. In this sense, the abstraction of the underlying hardware and software heterogeneity of the devices and the conversion of protocols for the exchange of information between them presents a key strategy. This thesis has specified an interoperability architecture to enable communication between devices in IoT and its integration with standard platforms. The architecture is based on recent research trends and best practices such as the architectural reference model (IoT-A) and the M2M functional architecture but adapted to some requirements that make this solution be used for the implementation of IoT applications in different environments.

The design of the architecture has led to first prototype implementation, called smart IoT gateway, which aims to enable the technical and syntactic interoperability of heterogeneous devices. Analogously, a second implementation is planned, called an interconnection architecture which extends the functionalities of the smart IoT gateway through the integration of a proxy interconnection

entity that allows the integration of heterogeneous devices with standard IoT platforms.

Consistent with the current pragmatic IoT approach, the utility and feasibility of architecture implementations have been demonstrated by means of testbeds applied to two IoT use cases. These use cases are derived from the European INTER-IoT project financed by the European Union through the Horizon H2020 program. The first of these is INTER-LogP, which aims to improve transport and logistics management processes in port environments, through the exchange of information between the different heterogeneous IoT platforms involved. The second, INTER-Health, pretend to monitor the environment and lifestyle of people in a decentralized manner and with mobility, to prevent health problems.

Finally, the experience acquired in the deployment of these cases of use has motivated the development of new proposals and IoT application studies that represent an additional contribution to the present doctoral thesis.

Resumen

La visión del Internet de las cosas (IoT) implica un ecosistema global hiperconectado en el que todos los dispositivos con capacidad de comunicación se conecten de manera ubicua al Internet. Sin embargo, para alcanzar todo el potencial de IoT, no es suficiente que los dispositivos estén conectados a Internet, también necesitan comunicarse e interactúan entre sí. Desafortunadamente, construir un ecosistema global de dispositivos que se conecten entre sí sin problemas es prácticamente imposible hoy en día. La razón es que IoT está constituida por una plétora de dispositivos heterogéneos en términos del formato de datos y componentes de comunicación que lo forman, como por ejemplo hardware, tecnologías y protocolos de comunicación. Esta heterogeneidad lleva inevitablemente a la aparición de "silos verticales" que están aislados al resto de IoT (p. ej., todavía necesitamos instalar 5 aplicaciones para interactuar con 5 dispositivos diferentes debido a la incompatibilidad entre estos dispositivos), exacerbada aún más por el hecho de que miles de millones de dispositivos de próxima generación dependerán de su capacidad de conectarse entre sí para obtener el mayor beneficio.

Por lo tanto, la interoperabilidad de dispositivos es uno de los principales desafíos a enfrentar en al ámbito de investigación de IoT. En tal sentido, la abstracción de la heterogeneidad hardware y software subyacentes de los dispositivos y la conversión de protocolos para el intercambio de información entre los mismos presenta una estrategia clave. En esta tesis se ha especificado una arquitectura de interoperabilidad para habilitar la comunicación entre dispositivos en IoT y su integración con plataformas IoT estándar. La arquitectura está fundamentada en tendencias de investigación recientes y mejores prácticas como el modelo de referencia arquitectónico (IoT-A) y la arquitectura funcional M2M, pero adaptada a unos requerimientos que hacen que esta solución pueda utilizarse para la implementación de aplicaciones IoT en diferentes entornos.

El diseño de la arquitectura, se ha llevado a una primera implementación prototipo, denominada *smart IoT gateway,* que tiene como objetivo habilitar la interoperabilidad técnica y sintáctica de dispositivos heterogéneos. De forma análoga, se proyecta una segunda implementación, denominada arquitectura de interconexión la cual extiende las funcionalidades del *smart IoT gateway* a través de la integración de una entidad *proxy* de interconexión que permite la integración de dispositivos heterogéneos con plataformas IoT estándar.

En forma consecuente con el actual enfoque pragmático de IoT, la utilidad y viabilidad de las implementaciones de la arquitectura se ha demostrado mediante *testbeds* aplicados a dos casos de uso IoT. Dichos casos de uso se derivan del proyecto Europeo INTER-IoT financiado por la Unión Europea a través del programa Horizonte H2020. El primero de ellos, es INTER-LogP, que tiene por objetivo mejorar los procesos de gestión de transporte y logística en entornos portuarios, por medio del intercambio de información entre las distintas plataformas IoT heterogéneas involucradas. El segundo, INTER-Health, pretende monitorizar el entorno y el estilo de vida de las personas de forma descentralizada y con movilidad, para prevenir problemas de salud.

Finalmente, la experiencia adquirida en el despliegue de estos casos de uso ha motivado el desarrollo de nuevas propuestas y estudios de aplicación de IoT que representan una contribución adicional de la presente tesis doctoral.

Resum

La visió de la Internet de les coses (IoT) implica un ecosistema global hiperconectado en el qual tots els dispositius amb capacitat de comunicació es connecten de manera ubiqua a la Internet. No obstant això, per a aconseguir tot el potencial de IoT, no és suficient que els dispositius estiguen connectats a Inter-net, també necessiten comunicar-se i interactuen entre si. Desafortunadamen-et, construir un ecosistema global de dispositius que es connecten entre si sense problemes és pràcticament impossible hui dia. El raó és que IoT està constituïda per una plètora de dispositius heterogenis en termes del for-mate de dades i components de comunicació que ho formen, com per exemple maquinari, tecnologies i protocols de comunicació. Aquesta heteroge-neidad porta inevitablement a l'aparició de "sitges verticals" que estan aisla-dues a la resta de IoT (p. ex., encara necessitem instal·lar 5 aplicacions per a in-teractuar amb 5 dispositius diferents a causa de la incompatibilitat entre aquests dispositius), exacerbada encara més pel fet que milers de milions de dispositius de pròxima generació dependran de la seua capacitat de connectar-se entre si per a obtindre el major benefici.

Per tant, la interoperabilitat de dispositius és un dels principals desa-fíos a enfrontar en a l'àmbit d'investigació de IoT. En tal sentit, la abstrac-ción de l'heterogeneïtat maquinari i programari subjacent dels dispositius i la conversió de protocols per a l'intercanvi d'informació entre els mateixos presenta una estratègia clau. En aquesta tesi s'ha especificat una ar-quitectura d'interoperabilitat per a habilitar la comunicació entre dispositius en IoT i la seua integració amb plataformes estàndard. L'arquitectura està fonga-esmentada en tendències d'investigació recents i millors pràctiques com el model de referència arquitectònic IoT-A i l'arquitectura funcional M2M, però adaptada a un requeriments que fan que aquesta solució puga utili-zarse per a la implementació d'aplicacions IoT en diferents entorns.

El disseny de l'arquitectura, s'ha portat a una primera implementació proto-tipus, denominada smart IoT gateway, que té com a objectiu habilitar la inter-operabilitat tècnica i sintàctica de dispositius heterogenis. De forma análo-ga, es projecta una segona implementació, denominada arquitectura de in-terconexión la qual estén les funcionalitats del smart IoT gateway a través de la integració d'una entitat proxy d'interconnexió que permet la integració de dispositius heterogenis amb plataformes IoT estàndard.

En forma conseqüent amb l'actual enfocament pragmàtic de IoT, la utilitat i viabilitat de les implementacions de l'arquitectura s'ha demostrat medien-et testbeds aplicats a dos casos d'ús IoT derivats del projecte Europeu INTER-IoT finançat per la Unió Europea a través del programa Horitzó H2020. El primer d'ells, és INTER-LogP, que té per objectiu millorar els processos de gestió de transport i logística en entorns portuaris, per mitjà de l'intercanvi d'informació entre les diferents plataformes IoT hete-rogéneas involucrades. El segon, INTER-Health, pretén monitorar l'entorn i l'estil de vida de les persones de forma descentralitzada i amb mo-vilidad, per a previndre problemes de salut.

Finalment, l'experiència adquirida en el desplegament d'aquests casos d'ús ha motivat el desenvolupament de noves propostes i estudis d'aplicació de IoT que representen una contribució addicional de la present tesi doctoral.

Contenido

Abstract ..... **x**

Contenido ..... **xvi**

Lista de Figuras ..... **xx**

Lista de Tablas ..... **xxiii**

Acrónimos ..... **xxiv**

1. Introducción **1**

1.1 Motivación 1
1.2 Objetivos 5
1.3 Metodología de investigación 6
1.4 Principales contribuciones 8
1.4.1 Artículos en revistas 8
1.4.2 Artículos en congresos internacionales 8
1.4.3 Capítulos de libro 9
1.4.4 Proyectos y contexto de investigación 10
1.5 Estructura de la tesis 10

2. Estado del arte **13**

2.1 Introducción 13
2.2 Internet de las Cosas (IoT) 14
2.2.1 Evolución desde una visión 14
2.2.2 Definiciones de IoT 16
2.2.3 Características de IoT 19
2.2.4 Componentes de IoT 21
2.2.5 Arquitecturas de referencia en IoT 25
2.2.5.1 Arquitectura de tres capas 26
2.2.5.2 Arquitectura de cuatro capas 26
2.2.5.3 Arquitectura de cinco capas 27
2.2.5.4 Arquitectura de red jerárquica para la conectividad escalable M2M 27
2.2.6 Dominios de aplicación de IoT 28
2.2.6.1 Dominio industrial - Transporte y logística 29
2.2.6.2 Dominio del bienestar sanitario - Cuidado de la Salud 30
2.2.6.3 Dominio de Ciudad Inteligente 31
2.2.7 Desafíos en el desarrollo de IoT 33

2.3 Interoperabilidad en el ámbito de IoT 34
2.4 Interoperabilidad de dispositivos en IoT 37
2.4.1 Heterogeneidad de los dispositivos 38
2.4.2 Patrones de comunicación de dispositivos 40
2.4.2.1 Comunicación de dispositivo a dispositivo (D2D) 40
2.4.2.2 Comunicación de dispositivo al cloud (D2C) 41
2.4.2.3 Comunicación de dispositivo a gateway (D2G) 43
2.4.3 Niveles de Interoperabilidad 45
2.4.3.1 Interoperabilidad Técnica 47
2.4.3.1.1 Tecnologías y protocolos de comunicación relacionados 48
2.4.3.1.1.1 Capa física y de enlace 48
2.4.3.1.1.2 Capa de red y de transporte 59
2.4.3.1.1.3 Capa de aplicación 62
2.4.3.2 Interoperabilidad sintáctica 73
2.4.3.3 Interoperabilidad semántica 77
2.5 Conclusiones 81

3. Especificación de la arquitectura 83

3.1 Introducción 83
3.2 Análisis de requerimientos 84
3.3 Visión general de la arquitectura 87
3.3.1 Vista de Contexto 88
3.3.2 Vista Funcional 90
3.3.2.1 Dominio Dispositivo 92
3.3.2.2 Dominio Gateway 93
3.3.2.2.1 Adaptador de protocolo 94
3.3.2.2.1.1 Controlador de dispositivo 95
3.3.2.2.1.2 Bridge 96
3.3.2.2.2 Administración de Datos 97
3.3.2.2.2.1 Transformación de datos 97
3.3.2.2.2.2 Almacenamiento local de datos 100
3.3.2.2.2.3 Procesamiento de Datos 101
3.3.2.2.2.4 Análisis de datos 103
3.3.2.2.2.5 Manejo de eventos 105
3.3.2.2.2.1 Broker message 107
3.3.2.2.3 Proxy de Interconexión 110
3.3.2.3 Dominio Cloud 110
3.3.2.3.1 Plataformas IoT 110
3.3.2.3.2 Análisis de datos *cloud* 111
3.3.2.4 Dominio de aplicación 112
3.4 Conclusiones 113

4. Smart IoT gateway para la interoperabilidad de dispositivos heterogéneos 114

4.1 Introducción ... 114
4.2 Arquitectura smart IoT gateway ... 115
4.3 Relación del smart IoT gateway con la arquitectura global ... 117
4.4 Caso de estudio AAL (Ambient assisted living) ... 120
4.5 Implementación del caso de estudio ... 122
4.5.1 Redes de sensores inalámbricas ... 122
4.5.2 Smart IoT gateway ... 125
4.5.2.1 Plataforma de desarrollo ... 125
4.5.2.2 Lenguaje de programación ... 126
4.5.2.3 Adaptador de dispositivos ... 127
4.5.2.3.1 Adaptador 6LowPAN ... 127
4.5.2.3.2 Adaptador ZigBee ... 129
4.5.2.3.3 Adaptador LoRa ... 130
4.5.2.3.4 Adaptador Wi-Fi ... 131
4.5.2.3.5 Adaptador Bluetooth ... 132
4.5.2.4 Message broker ... 132
4.5.2.5 Transformación de datos ... 135
4.5.2.6 Procesamiento de datos ... 137
4.5.2.7 Manejo de eventos ... 139
4.5.2.8 Almacenamiento de datos ... 140
4.5.2.9 Interfaz gráfica de usuario (GUI) ... 142
4.6 Evaluación y resultados ... 143
4.6.1 Escenarios de prueba ... 144
4.6.2 Uso de CPU ... 145
4.6.3 Uso de memoria RAM ... 146
4.6.4 Consumo de energía ... 147
4.7 Trabajos relacionados ... 148
4.8 Conclusiones ... 155

5. Arquitectura de interconexión de dispositivos heterogéneos con plataformas IoT ... 156

5.1 Introducción ... 156
5.2 Relación de la arquitectura de interconexión con la arquitectura global ... 158
5.3 Trabajos relacionados ... 158
5.4 OneM2M y especificaciones de interconexión ... 159
5.4.1 Estándar oneM2M ... 159
5.4.2 Interconexión a través del IPE ... 161
5.5 Arquitectura de interconexión ... 162
5.6 Verificación de la arquitectura de interconexión ... 165
5.6.1 Caso de estudio: INTER-LogP ... 165

5.6.2 Procedimientos de interconexión entre los dispositivos y la plataforma oneM2M ... 168
5.6.3 Implementación y servicios proporcionados a partir de la interconexión. 173
5.6.3.1 Implementación ... 173
5.6.3.2 Servicios ... 177
5.6.3.3 Resultados preliminares ... 179
5.7 Extensibilidad de la arquitectura de interconexión ... 180
5.7.1 FIWARE ... 180
5.7.2 Procedimientos de interconexión entre los dispositivos y la plataforma FIWARE ... 183
5.8 Conclusiones ... 185

6. Sistema de detección de caídas para personas mayores utilizando IoT y Big data ... 187

6.1 Introducción ... 187
6.2 Estado del arte y trabajos relacionados ... 189
6.3 Relación de la arquitectura del sistema con la arquitectura global ... 194
6.4 Sistema de detección de caídas utilizando IoT y Big Data ... 195
6.4.1 Extracción de características ... 196
6.4.1.1 Conocimiento histórico de SisFall ... 196
6.4.1.2 Ventanas deslizantes junto con SMA ... 197
6.4.2 Selección del algoritmo de aprendizaje supervisado ... 198
6.4.2.1 Algoritmos de aprendizaje supervisado ... 198
6.4.2.2 Selección del clasificador ... 200
6.4.3 Sistema de detección de caídas ... 203
6.4.3.1 Dispositivo Portátil ... 204
6.4.3.2 Red de comunicaciones inalámbrica ... 205
6.4.3.3 Smart IoT gateway y servicios *fog* ... 205
6.4.3.4 Servidor cloud ... 211
6.4.4 Resultados y evaluaciones ... 211
6.4.4.1 Resultados de la detección de caídas ... 211
6.4.4.2 Comparación con otros sistemas ... 213
6.5 Conclusiones ... 216

7. Conclusiones y líneas de trabajo futuras ... 218

7.1 Conclusiones generales ... 218
7.2 Trabajos futuros ... 222

Bibliografía ... 225

Lista de Figuras

Figura 1.1: Relación del número de personas de la población mundial respecto a la cantidad de dispositivos conectados a Internet desde 2003 y su previsión en 2020.....2
Figura 1.2: Representación de aplicaciones vistas como silos verticales de información en IoT 3
Figura 1.3: Metodología de investigación utilizada en el desarrollo de la tesis.....6
Figura 2.1: Representación de IoT 14
Figura 2.2: IoT en la evolución de Internet 15
Figura 2.3: Dimensiones de la comunicación en IoT..... 19
Figura 2.4: Elementos principales de IoT 22
Figura 2.5: Arquitecturas de IoT..... 25
Figura 2.6: Arquitectura M2M ETSI..... 27
Figura 2.7: Ejemplos de dominios de aplicaciones de IoT..... 32
Figura 2.8: Ejemplo del patrón de comunicación D2D..... 41
Figura 2.9: Ejemplo del patrón de comunicación D2C. 42
Figura 2.10: Ejemplo del patrón de comunicación D2G..... 44
Figura 2.11: Niveles de interoperabilidad asociados a los dispositivos IoT..... 45
Figura-2.12: Ejemplo de dos dispositivos no interoperables que utilizan el mismo estándar..... 47
Figura 2.13: Arquitectura ZigBee (Capa de Red y Aplicación) 51
Figura-2.14: Topologías, nodos lógicos y dispositivos soportadas por la tecnología ZigBee..... 52
Figura 2.15: Componentes de una red LoRaWAN..... 54
Figura 2.16: Red y pila de protocolos 6lowPAN..... 60
Figura 2.17: Funcionalidad de CoAP y HTTP (Adaptado de (Bormann *et al.,* 2012))63
Figura-2.18 Tipos de Mensaje (a) Confirmable con respuesta *piggybacked; (b)* Confirmable con respuesta *separate (c)* No-confirmable. 65
Figura 2.19 Funcionamiento del Protocolo MQTT..... 67
Figura 2.20 Ejemplos de mensajes MQTT utilizando diferentes tipos de QoS..... 70
Figura 2.21 Ejemplo de representación de datos en XML 74
Figura 2.22 Ejemplo de representación de datos en JSON..... 76
Figura 2.23: Dispositivos no interoperable semánticamente. 77
Figura 2.24: Una visión general del patrón SSO de la ontología SSN. 79
Figura 3.1: Vista de contexto de la Arquitectura. 89
Figura 3.2: Arquitectura de interoperabilidad..... 91
Figura 3.3: Componentes de un dispositivo IoT 93

Figura 3.4: Grupo Funcional Dominio de Gateway 94
Figura 3.5: Ejemplos de pilas de protocolos de comunicación admitidas por el controlador de dispositivo 95
Figura 3.6: Proceso de transformación de datos 98
Figura 3.7: Estructura del formato de datos interno 99
Figura 3.8: Estructura de los atributos y metadatos 99
Figura 3.9: Componente funcional almacenamiento de datos 100
Figura 3.10: Componente funcional procesamiento de datos 101
Figura 3.11: Componente funcional análisis de datos 105
Figura 3.12: Flujo de datos de la arquitectura utilizando el mecanismo *publish/subscribe* 109
Figura 3.13: Proceso del aprendizaje supervisado ejecutado en el dominio Cloud para crear el modelo ML (parte superior) e instancia local del modelo ML en el dominio del gateway (parte inferior) 112
Figura 4.1: Versión embrionaria "*Smart IoT gateway"* de la arquitectura 115
Figura 4.2: Arquitectura del caso de estudio (AAL-IoTSys) 121
Figura 4.3: Dispositivos IoT utilizados en el prototipo AAL 124
Figura 4.4: Plataforma de desarrollo del *smart IoT gateway* e interfaces integradas 125
Figura 4.5: Lista de recursos obtenida al ejecutar la petición CoAP GET al recurso CoAP /.well-known/core del dispositivo 6LoWPAN. 128
Figura 4.6: Ejemplo de petición GET CoAP para obtener los datos del sensor de temperatura 129
Figura 4.7: Opciones de inicio para establecer a comunicación RF en la red LoRa implementada 131
Figura 4.8: Ejemplo de la publicación del *raw data* del sensor *heartRate* por parte del adaptador Wi-Fi al Message *Broker* 134
Figura 4.9: Ejemplo de publicación de datos desde un adaptador al *Message Broker*. 134
Figura 4.10: Ejemplo de datos sin procesar *(raw data*) de los sensores de temperatura y localización (GPS) 135
Figura 4.11: Ejemplo de la representación de los datos de los sensores de temperatura (izquierda) y localización (derecha) 136
Figura 4.12: Ejemplo de la configuración de una regla en el CEP 138
Figura 4.13: Ejemplo de datos de los sensores almacenados en la base de datos local del *smart IoT gateway* 141
Figura 4.14: Aplicación Web para la monitorización de las personas mayores 142
Figura 4.15: CPU promedio del Smart IoT Gateway en los escenarios de carga baja y alta evaluados. 145
Figura 4.16: Uso de la memoria RAM en los escenarios de carga baja y alta evaluados. 145
Figura 4.17: Consumo de la batería en los escenarios de carga baja y alta evaluados. 147
Figura 5.1: Arquitectura funcional oneM2M 159

Figura 5.2: Arquitectura de *interworking* de dispositivos NoDN con oneM2M basada en IPE 162
Figura 5.3: *Testbed* de interconexión 166
Figura 5.4: Flujo de información de los procedimientos de interconexión 170
Figura 5.5: Transformación de datos en mensajes de solicitud primitiva oneM2M. 171
Figura 5.6: Diagrama funcional del *testbed* implementado 172
Figura 5.7: Aplicación Web "Smart truck" 175
Figura 5.8: Consumo medio adicional de combustible de un camión por viaje 179
Figura 5.9: Arquitectura IoT de FIWARE 181
Figura 5.10: Modelo de información de contexto FIWARE NGSI 182
Figura 5.11: Esquema de interconexión con la plataforma FIWARE 183
Figura 5.12: Transformación de datos en formato JSON-LD 183
Figura 6.1: Etapas del sistema IoTE-Fall 194
Figura 6.2: Validación cruzada (5-*fold*) aplicada a *Motion-DT* para crear y entrenar cada modelo 199
Figura 6.3: Curvas ROC y valores de AUC alcanzados para cada tipo de caída con un promedio de 0.96 (panel superior) y para cada tipo de ADL con un promedio de 0.95 (panel inferior). 201
Figura 6.4: Visión general del sistema IoTE-Fall 202
Figura 6.5: Dispositivo portátil (a) Diagrama esquemático; (b) Estructura de hardware 203
Figura 6.6: Clasificación para uno de los árboles de decisión que forman el *ensemble*: (izquierda) diagrama del árbol de decisión, (centro) grado de confianza alcanzado por el árbol de decisión al predecir la clase LF , (derecha) regla de decisión generada por el árbol de decisión para predecir la clase LF. 206
Figura 6.7: Ejemplo de notificaciones de detección de caídas 207
Figura 6.8: Ontología extendida (Device) de UniverSAAL para la representación semántica de los eventos de caídas y actividades 208

Lista de Tablas

Tabla 2.1: Visión general de la clasificación de constrained devices..... 38
Tabla 2.2: Principales características de las tecnologías de comunicación inalámbrica 58
Tabla 2.3: Comparativa entre los protocolos de capa de aplicación CoAP y MQTT 73
Tabla 3.1: Requerimientos de la arquitectura 85
Tabla 3.2: Estructura de un mensaje de suscripción..... 106
Tabla 4.1: Cumplimiento de los requerimientos de la arquitectura por parte del *smart IoT gateway*..... 117
Tabla 4.2: Dispositivos IoT utilizados en el caso de estudio (AAL-IoTSys) 123
Tabla 4.3: Reglas configuradas en el CEP para la detección de eventos. 137
Tabla 4.4: Dispositivos IoT conectados al *smart IoT gateway* a través de varias interfaces de comunicación 143
Tabla 4.5: Comparación del *smart IoT gateway* con propuestas alternativas..... 153
Tabla 5.1: Dispositivos IoT empleados en el caso de uso..... 173
Tabla 5.2: Consumo de combustible promedio adicional por camión por hora..... 178
Tabla 6.1: Caídas y actividades de SisFall utilizadas en este trabajo 196
Tabla 6.2: Estadísticas de la evaluación de los algoritmos de clasificación 202
Tabla 6.3: Ruta de acceso al recurso de aceleración del dispositivo portátil para obtener los datos de movimiento..... 204
Tabla 6.4: Resumen consolidado de los resultados alcanzados por el sistema IoTE-Fall en la detección de caídas 212
Tabla 6.5: Comparación de IoTE-Fall con otros trabajos relacionados..... 213

Acrónimos

6LowPAN	*IPv6 over Low power Wireless Personal Area Networks*
AAL	*Ambient Assisted Living*
ACK	*Acknowledge*
ADL	*Activities of Daily Living.*
AHA	*Active and Healthy Ageing*
AND	*Application Dedicated Nodes*
AP	*Access Point*
API	*Application Programming Interface*
APO	*Application Object*
APS	*Application Sub-Layer*
ASN	*Application Service Nodes*
BAN	*Body Area Network*
BW	*Bandwith*
CEP	*Complex Event Processing*
Che	*Context History Entrepot*
CoAP	*Constrained Application Protocoll*
CON	*Confirmable*
CPE	*Electronic Product Codes*
CR	*Coding Rate*
CSS	*Chirp Spread Spectrum*
CSV	*Comma-Separated Values*
D2C	*Device to Cloud*

D2D	*Device to Device*
D2G	*Device to Gateway*
DAG	*Directed Acyclic Graph*
DLS	*Dynamic Real-Time Lighting System*
DODAG	*Destination oriented DAG*
FS	*Fusion systems*
GE	*Generic Enabler*
GPRD	*General Data Protection Regulation*
GUI	*Graphical User Interface*
HAL	*High level Architecture*
INTER-Health	*Interconnection for Mobile Health*
INTER-LogP	*Interconnection for Transport and Logistics*
IoT	*Internet of Things*
IoT-A	*Internet of Things Architecture*
IoT-ARM	*IoT Architectural Reference Model*
IPE	*Interworking Proxy Entity*
JSON	*JavaScript Object Notation*
LLN	*Low-Power and Lossy Networks*
LR-WPAN	*Low-Rate Wireless Personal Area Networks*
M2M	*Machine-to-Machine*
ML	*Machine Learning*
MN	*Middle Node*
MQTT	*Message Queue Telemetry Transport*
NFC	*Near Field Communication*
NGSI	*Next Generation Service Interfaces*
NON	*No Confirmable*
NWS	*Non-Wearable-Based Systems*
OASC	*Open & Agile Smart Cities*

OF	*Objective Function*
OWL	*Web Ontology Language*
PAN	*Personal Area Network*
QoL	*Quality of life*
RA	*Router Advertisement*
RDF	*Resource Description Framework*
RFID	*Radio Frequency Identification*
RPL	*Routing Protocol for LLN*
RS	*Router Solicitations*
RST	*Reset*
SaaS	*Software as a Service*
SBC	*Single-Board Computer*
SF	*Spread Factor*
SoC	*System on Chip*
SSN	*Semantic Sensor Network*
SSO	*Stimulus-Sensor-Observation*
SSP	*Security Service Provider.*
SVM	*Support Vector Machine*
uCode	*Ubiquitous Code*
WLAN	*Wireless local Area Network*
WPAN	*Wireless Personal Area Networks*
WS	*Wearable-Based Systems*
WSN	*Wireless Sensor Networks*
WWAN	*Wireless Wide Area Network*
XML	*Extensible Markup Language*
ZDO	*ZigBee Device Object*

Capítulo 1
Introducción

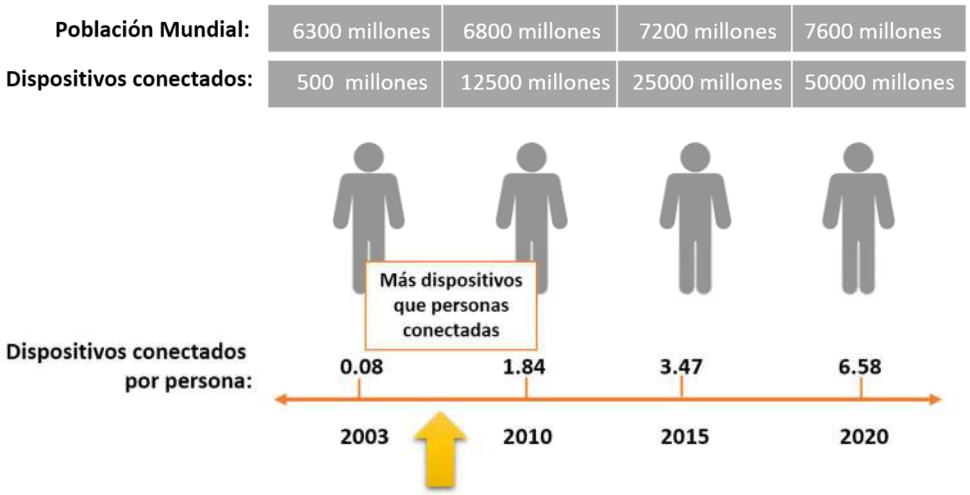
«La mente que se abre a una nueva idea
jamás volverá a su tamaño original»

Albert Einstein (1879-1955)

1.1 Motivación

En un futuro cercano, todos los objetos que nos rodean incluso los más simples se convertirán en "inteligentes". Estarán conectados a Internet para explotar todos los beneficios potenciales del paradigma de Internet de las cosas (IoT, por sus siglas en inglés, *Internet of Things*), allanando así el camino a nuevos servicios de aplicaciones, informática y escenarios de comunicación.

Un pronóstico reciente realizado por la Corporación Internacional de Datos (IDC) proyecta a IoT y al ecosistema asociado a ser un mercado de $ 7,1 billones de dólares en el 2020 (Lund & Morales, 2014), que incluirá 50 mil millones de dispositivos conectados (6,58 dispositivos por persona) en 2020 con un impacto diez veces mayor que Internet (Figura 1.1). Por su parte, Morgan Stanley, proyecta 75 mil millones de dispositivos conectados a la red para 2020. Mirando más lejos, la empresa china *Huawei* pronostica 100 mil millones de dispositivos conectados a Internet para 2025. Estas previsiones dejan ver un nuevo modo de interacción en el mundo físico, inspirado en la idea de ubicuidad, donde todos los objetos que nos rodean (p.ej. sensores, automóviles, electrodomésticos, receptores de audio/ video, detectores de humo, etc.) que habitualmente no se consideran computadoras se puedan conectar a Internet en cualquier momento y en cualquier lugar (D. C. Yacchirema & Palau, 2016). Los avances en electrónica, telecomunicaciones, tecnologías de sensores, redes inalámbricas de sensores, así como el costo cada vez menor de los dispositivos y conectividad de datos, habilitan esta posibilidad.



Fuente: (Evans, 2011)

Figura 1.1: Relación del número de personas de la población mundial respecto a la cantidad de dispositivos conectados a Internet desde 2003 y su previsión en 2020

La visión de IoT implica un ecosistema global hiperconectado en el que todos los dispositivos con capacidades de comunicación, detección/actuación puedan interconectarse e interactuar entre sí, siempre que sea necesario, (Atzori *et al.*, 2010) siendo capaces de recoger información, procesarla y compartirla. Esta visión brinda un amplio conjunto de oportunidades a consumidores, fabricantes, empresas, gobiernos y comunidades académicas, que resulta en aplicaciones y servicios altamente diversificados enfocados a diferentes contextos, incluyendo, la ciudad inteligente, salud móvil, hogar automatizado, transporte inteligente, entre otros. Sin embargo, uno de los principales obstáculos al que se enfrenta IoT es el alto grado de heterogeneidad de los dispositivos de diferentes tipos y perfiles técnicos, con diferentes formato de datos y componentes de comunicación que lo forman (protocolos, tecnologías y hardware), como se reconoce ampliamente en la literatura (Gonzalez-Usach *et al.*, 2019; Noura *et al.*, 2018; Yacchirema *et al.*, 2016; Yacchirema & Palau, 2016; Khan *et al.*, 2012; Miorandi *et al.*, 2012. IoT, por su propia naturaleza, es un entorno extremadamente heterogéneo basado en una amplia gama de dispositivos diferentes que recopilan información heterogénea del entorno. La gama de dispositivos puede ser variada, desde dispositivos con restricciones hasta dispositivos potentes; diferentes en términos de consumo de energía y disponibilidad de recursos (memoria, almacenamiento, procesamiento, etc.).

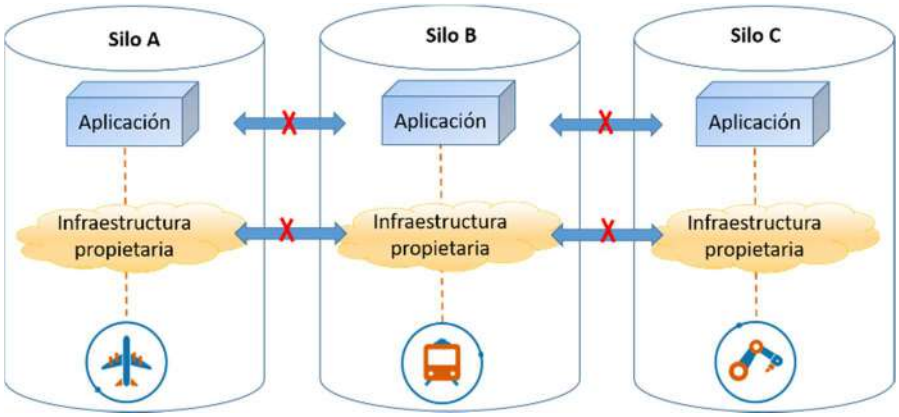


Figura 1.2: Representación de aplicaciones vistas como silos verticales de información en IoT

Estas diferencias están creando "silos verticales de información" (Figura 1.2) con un enfoque de aplicación limitado, imponiendo formatos de datos e interfaces específicas. Estos silos no pueden interactuar entre sí de manera fácil y eficiente, ya que normalmente conectan un tipo específico de dispositivo a una aplicación determinada (que a menudo utiliza tecnología patentada) a través de su propia infraestructura de TIC. Las aplicaciones similares no comparten ninguna información, lo que resulta en una redundancia innecesaria y en un aumento de los costos.

Si bien algunos fabricantes de dispositivos y proveedores de servicios proporcionan su propio ecosistema para garantizar la interoperabilidad total entre sus productos, otros optan por integrar sus dispositivos en los sistemas existentes. Como consecuencia de ello, los sistemas implementan sus propios protocolos de comunicación, modelos de datos, APIs (del inglés, *Application programming interface*) y servicios.

Debido a este diseño de sistema de "silos verticales", un entorno inteligente necesitaría implementar múltiples sistemas y servicios de IoT diferentes para satisfacer las múltiples necesidades de cada dominio de aplicación. De hecho, esta es una situación muy similar a la de los tiempos en que existían muchas tecnologías de red no interoperables (es decir, que no podían intercambiar información) antes de que Internet se popularizara. Los silos pueden convertirse en la barrera de IoT, y "*solo si podemos resolver el problema de interoperabilidad, podemos tener una verdadera Internet de las cosas*"(Tan, L., & Wang, 2010).

La falta de interoperabilidad entre dispositivos de IoT, es uno de los problemas más serios que causa la imposibilidad de desarrollar aplicaciones IoT que aprovechen la información proveniente de múltiples dispositivos e impide la adopción de IoT aún más rápida a gran escala. Mientras los dispositivos individuales de IoT no puedan comunicarse entre sí, la visión de un ecosistema global de IoT no se realizará. De hecho, la interoperabilidad de IoT es necesaria para alcanzar el 60% del valor potencial en las aplicaciones de IoT (Mckinsey Global Institute, 2015).

Así, la necesidad de una mejor integración horizontal e interoperabilidad entre dispositivos heterogéneos y su integración e inclusión perfecta con plataformas IoT estándar, es la clave para permitir el intercambio y uso de datos relevantes, y establecer sinergias significativas que permiten crear soluciones inteligentes que pueden habilitar diferentes escenarios de IoT.

Las diferentes propuestas para abordar la interoperabilidad en IoT provienen de iniciativas comerciales y soluciones basadas en los intentos de normalización de los diferentes organismos de estandarización, así como también del resultado de proyectos públicos de investigación, como el FP7 (Séptimo programa Marco) de la Unión Europea. Entre las investigaciones más relevantes podemos mencionar WSO2 (2015), FIESTA-IoT (2015), OpenIoT (2017), SymbIoTe (2015). Los principales esfuerzos de interoperabilidad de estas soluciones se centran en la semántica y/o en las interfaces de programación de alto nivel sin considerar los problemas de compatibilidad en las capas inferiores.

De este modo, en esta tesis se ha llevado a cabo la especificación de una arquitectura que permite la interoperabilidad entre dispositivos físicos en el ecosistema IoT atendiendo a diferentes niveles de interoperabilidad, contribuyendo de esta forma al despliegue de soluciones IoT horizontales. Para ello, de forma consecuente con el actual enfoque pragmático de IoT, la arquitectura ha sido validada a través de dos casos de uso IoT derivados del proyecto Europeo Inter-IoT INTER-IoT (InterIoT, 2016) financiado por la UE H2020.

- INTER-LogP, en el que se ha implementado una instancia de la arquitectura que permite compartir la información de dispositivos heterogéneos con diferentes plataformas IoT orientadas al transporte portuario y la logística a través de un *Interworking proxy entity* (IPE).

- INTER-Health, en el que se ha implementado dos instancias de la arquitectura. Una instancia para permitir la monitorización de adultos mayores en entornos AAL y una instancia para soportar el envejecimiento activo y saludable de personas mayores (AHA por sus siglas en inglés, *Active and healthy ageing*) que permite la detección de caídas.

Dejando abierta la posibilidad de extender la arquitectura definida y resultados obtenidos a otras áreas de aplicación y casos de estudios; monitorización de la calidad del aire, apnea del sueño, síndrome metabólico, etc.

1.2 Objetivos

A partir de los precedentes listados, la presente tesis atiende a la resolución del problema de la interoperabilidad de dispositivos físicos, centrándose en el nivel técnico y sintáctico, contribuyendo a la utilización masiva de los dispositivos IoT independiente de los protocolos y tecnologías subyacentes.

En tal contexto el objetivo principal del presente trabajo de tesis se enfoca en:

Definir una arquitectura de interoperabilidad que permita la comunicación sin fisuras entre dispositivos heterogéneos, y que sea fácilmente integrable con plataformas IoT estándar.

Para alcanzar el objetivo principal, los objetivos específicos y desafíos a considerar son:

O1. Identificar y enmarcar el problema de la interoperabilidad en el ecosistema de IoT con énfasis en la interoperabilidad de dispositivos.

O2. Analizar la heterogeneidad y los diferentes patrones utilizados para la comunicación de dispositivos en IoT. Así como también los diferentes niveles a través de los cuales se puede abordar el problema de interoperabilidad de dispositivos en IoT y las tecnologías habilitadoras en cada nivel.

O3. Diseñar y especificar una arquitectura que permita la interoperabilidad de dispositivos físicos y la integración con plataformas IoT estándar independiente de las tecnologías y protocolos de comunicación subyacentes utilizados.

O4. Desarrollar un prototipo de la arquitectura a través de la implementación de un *Smart IoT gateway* que se adapte a los requerimientos que se proponen en la tesis.

O5. Aplicar la arquitectura propuesta en entornos realistas de operación enfocados al transporte y logística, y la salud derivados del proyecto Europeo INTER-IoT.

1.3 Metodología de investigación

La metodología de investigación empleada para lograr los objetivos descritos anteriormente se planificó en tres fases principales de acuerdo con un enfoque de investigación constructiva, siguiendo un proceso dinámico e interactivo entre las diferentes fases como se muestra en la Figura 1.3.

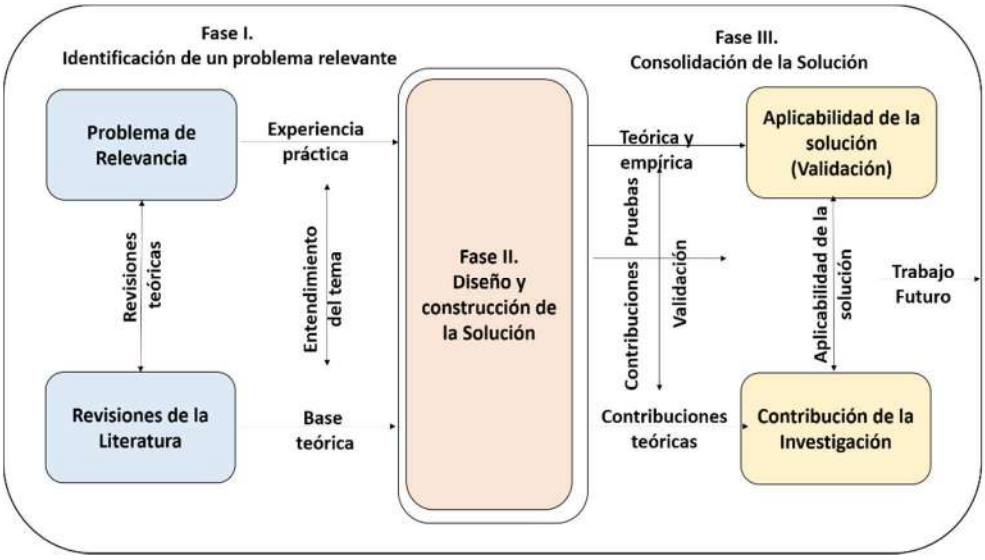


Figura 1.3: Metodología de investigación utilizada en el desarrollo de la tesis

La investigación constructiva se utiliza para definir y resolver problemas prácticos, con la implicación general de contribuir al cuerpo de conocimiento existente (Oyegoke, 2011).

La primera fase de este enfoque consiste en seleccionar un problema relevante desde un punto de vista práctico y obtener una comprensión profunda del área de estudio. Específicamente, la investigación que realizó esta tesis comenzó con un estudio exhaustivo del estado del arte de los trabajos de investigación de IoT tanto en el mundo académico como en la industria con el objetivo de identificar los desafíos no resueltos que impiden la adopción de esta tecnología, caracterizando el problema de la interoperabilidad como uno de los principales desafíos

a resolver. Las fuentes consultadas fueron desde publicaciones científicas (actas de congresos y artículos de revistas) hasta documentos de estándares y proyectos de investigación (publicaciones de grupos de trabajo de ingeniería (RFCs), estándares, propuestas de normas, etc.) que tratan diferentes temas relacionados con las principales líneas de investigación que se aportarán en esta tesis. Los temas de interés para el desarrollo de esta tesis fueron, componentes de IoT, arquitecturas de referencia para el desarrollo de soluciones IoT, desafíos en la adopción de IoT, interoperabilidad en el ámbito de IoT, patrones de comunicación de dispositivos, protocolos de comunicación (para permitir la conectividad de dispositivos) y formatos de datos (para serializar la información). La rigurosidad, la originalidad y la calidad fueron tres de los criterios principales al elegir las fuentes de conocimiento de los trabajos relacionados con las áreas de investigación de interés para esta tesis.

La segunda fase de la metodología de investigación constructiva implica el diseño de una o más soluciones aplicables al tema de interés que demuestre su viabilidad. Así, en esta fase, se realizó la especificación, diseño e implementación de una solución que contribuyen al estado de la técnica analizado en la primera fase. Se abordaron dos importantes contribuciones en las respectivas áreas de conocimiento. Por un lado, se especificó una arquitectura modular y flexible que puede impulsar la construcción de soluciones y servicios IoT, mediante la resolución de la interoperabilidad de dispositivos y la integración de estos dispositivos con plataformas IoT estándar. Por otro lado, se implementó un prototipo (*Smart IoT Gateway*) de acuerdo con esa arquitectura y se llevaron a cabo pruebas de rendimiento, para validar su factibilidad para implementaciones reales. Posteriormente se extendió la funcionalidad del *smart IoT gateway* y se definió una arquitectura de interconexión para la integración de dispositivos heterogéneos con plataformas IoT estándar.

Finalmente, la tercera fase de la metodología de investigación constructiva se centra en consolidar la contribución al estado de la técnica y observar la utilidad general de los resultados. En el contexto de la tesis, esta fase se centró en validar las contribuciones de esta tesis a través de casos de estudio (*testbeds*) derivados del proyecto Europeo, INTER-IoT. Los resultados obtenidos en esta fase se analizaron para evaluar cómo los trabajos de investigación específicos lograron los objetivos definidos para mejorar el estado de la técnica.

1.4 Principales contribuciones

En esta sección se presentan las publicaciones generadas en el marco de la tesis doctoral, las cuales se han clasificada en artículos en revistas, congresos internacionales y libros.

1.4.1 Artículos en revistas

- **Yacchirema, D.**, de Puga, J. S., Palau, C., *et al.* (2019) Fall detection system for elderly people using IoT and ensemble machine learning algorithm. *Personal and Ubiquitous Computing*. doi:10.1007/s00779-018-01196-8. Journal Q1.

- **Yacchirema, D.**, Sarabia-Jácome, D., Palau, C. E., *et al.* (2018) System for monitoring and supporting the treatment of sleep apnea using IoT and big data. *Pervasive and Mobile Computing*, 50, 25–40. doi:https://doi.org/10.1016/j.pmcj.2018.07.007. Journal Q1.

- **Yacchirema, D. C.**, Sarabia-Jácome, D., Palau, C. E., *et al.* (2018) A Smart System for sleep monitoring by integrating IoT with big data analytics. *IEEE Access*, 1. doi:10.1109/ACCESS.2018.2849822. Journal Q1.

- **Yacchirema, D. C.**, & Palau, C. E. (2016) Smart IoT Gateway For Heterogeneous Devices Interoperability. *IEEE Latin America Transactions*, 14, 3900–3906. doi:10.1109/TLA.2016.7786378. Journal Q2.

- **Yacchirema, D. C.**, Pradilla, J., Esteve M., and Palau, C., *et al.* (2019) "Architecture for Internet of Things Interoperability: A Case Study on SmartCities," in *IEEE Latin America Transactions. (Aceptado)*. Journal Q2.

1.4.2 Artículos en congresos internacionales

- **Yacchirema, D.**, Belsa-Pellicer, A., Palau, C., *et al.* (2018) OneM2M Based-Interworking Architecture for Heterogeneous Devices Interoperability in IoT. In: *2018 IEEE Conference on Standards for Communications and Networking (CSCN)*, pp. 1–6. doi:10.1109/CSCN.2018.8581740

- **Yacchirema, D.**, Suárez de Puga, J., Palau, C., *et al.* (2018) Fall detection system for elderly people using IoT and Big Data. In: *Procedia Computer Science*, Vol. 130, pp. 603–610. doi:https://doi.org/10.1016/j.procs.2018.04.110

- **Yacchirema, D.**, Gonzalez-Usach, R., Palau, C., et al. (2018) Interoperability of IoT Platforms applied to the transport and logistics domain. In:. Zenodo. doi:10.5281/zenodo.1451428

- Pace, P., Gravina, R., Aloi, G., Fortino, G., Fides-Valero, k., Ibanez-Sanchez., G., Traver, V., Palau.C., **Yacchirema, D**., et al. (2017) IoT platforms interoperability for active and assisted living healthcare services support. In: 2017 Global Internet of Things Summit (GIoTS), pp. 1–6. doi:10.1109/GIOTS.2017.8016250

- **Yacchirema, D.**, Palau, C., & Esteve, M. (2017) Enable IoT Interoperability in Ambient Assisted Living: Active and Healthy Aging Scenarios. In: 1st Edition of Globe-IoT 2017: Towards Global Interoperability among IoT Systems, p. 6. Las Vegas.

- **Yacchirema, D. C.**, Esteve, M., & Palau, C. E. (2016) Design and implementation of a Gateway for Pervasive Smart Environments. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 4454–4459. doi:10.1109/SMC.2016.7844933

1.4.3 Capítulos de libro

- **Yacchirema, D.**, Palau, C., & Esteve, M. (2019) Edge-of-Things Computing-Based Smart Healthcare System. In: *Handbook of Research on the IoT, Cloud Computing, and Wireless Network Optimization*, pp. 1–22. IGI Global.

- Gonzalez-Usach, R., **Yacchirema, D.**, Julian, M., *et al.* (2019) Interoperability in IoT. In: *Handbook of Research on Big Data and the IoT*, pp. 149–173. IGI Global.

- Gonzalez-Usach, R., **Yacchirema, D.**, Collado, V., *et al.* (2017) AmI Open Source System for the Intelligent Control of Residences for the Elderly. In: *Interoperability, Safety and Security in IoT*, pp. 46–52. Springer.

1.4.4 Proyectos y contexto de investigación

Esta tesis doctoral se ha desarrollado en el contexto del grupo de investigación SATRD (Sistemas y Aplicaciones de Tiempo Real Distribuido) del Departamento de Comunicaciones de la Universitat Politècnica de València (UPV).

Los trabajos que han hecho posible el desarrollo de esta tesis, se engloban en proyectos de I+D financiados con fondos públicos. En particular, esta tesis ha realizado colaboraciones en los siguientes proyectos de investigación, y algunas publicaciones han sido financiadas por los mismos:

- Proyecto INTER-IoT: "*Interoperability Internet of Things*" financiado por la Comisión Europea dentro del Programa Marco de Investigación e Innovación Horizonte 2020 en virtud del acuerdo de subvención nº 687283. (Responsable UPV). De Enero de 2016 a Diciembre de 2018.

- Proyecto ACTIVAGE: "*Supporting Active and Healthy Ageing through IoT technologies*" orientado a entornos de vida inteligente. Financiado por la Comisión Europea dentro del Programa Marco de Investigación e Innovación Horizonte 2020 en virtud del acuerdo de subvención nº 732679. (Responsable UPV). De Enero de 2017 a Diciembre de 2019.

Por otra parte, esta tesis también ha sido realizada gracias al apoyo del Estado Ecuatoriano a través de la Escuela Politécnica Nacional y el programa de becas de la Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación (SENESCYT) de Ecuador (mediante una beca de estudios doctorales concedida a la autora de esta investigación).

1.5 Estructura de la tesis

Para alcanzar los objetivos descritos en la sección 1.2, el presente documento está estructurado de la siguiente manera:

- Capítulo 1- Introducción

En este capítulo se presenta la motivación del problema detectado, la interoperabilidad de los dispositivos físicos en el Internet de las cosas, que es el objeto de estudio de esta tesis. Adicionalmente, se presentan los objetivos y el marco de trabajo que rige esta investigación. Finalmente, se hace una descripción de la estructura de la tesis y las principales contribuciones generadas y publicadas en el campo de investigación. El capítulo está relacionado con el objetivo **O1.**

- Capítulo 2- Estado del arte

En este capítulo se realiza una revisión global del estado del arte de los temas más importantes de IoT relacionados con este trabajo. Para ello se introduce en los aspectos fundamentales de la interoperabilidad de dispositivos IoT y las capas o niveles a través de la cual se enmarca dicha interoperabilidad. También se explica los patrones de comunicación para integrar los dispositivos a Internet y permitir la comunicación entre dispositivos. En la última parte del capítulo se realiza un análisis de las diferentes tecnologías y protocolos de comunicación, así como los formatos de datos utilizados para abordar la conectividad y serialización de los datos dentro del contexto de la interoperabilidad de dispositivos IoT. El capítulo está relacionado con el objetivo **O2.**

- Capítulo 3- Definición de la arquitectura de interoperabilidad de dispositivos físicos para IoT

En este capítulo se propone una arquitectura para la interoperabilidad de dispositivos IoT, la cual aplica principios de diseño del modelo de referencia arquitectónico (IoT-A), el patrón de comunicación D2G y la arquitectura de referencia M2M para permitir la interoperabilidad técnica y sintáctica de dispositivos en IoT. En particular, se presenta una descripción detallada de los grupos, módulos y componentes funcionales de la arquitectura. El capítulo está relacionado con el objetivo **O3.**

- Capítulo 4- *Smart IoT gateway* para la interoperabilidad de dispositivos heterogéneos

En este capítulo se presenta la implementación de un prototipo de la arquitectura, así como la evaluación de su funcionalidad a través de un escenario AAL (del inglés *Ambient Assisted Living*). La evaluación abarca la implementación de un *testbed* conformado por un conjunto de dispositivos heterogéneos (sensores y actuadores), el desarrollo de un sistema de monitorización de adultos mayores en entornos interiores para AAL basado en IoT, la implementación de la interoperabilidad entre dispositivos. Además, se presentan los resultados que evalúan el rendimiento del prototipo en escenarios de carga alta y baja. Finalmente se incluye un análisis del trabajo relacionado. El capítulo está relacionado con el objetivo **O4.**

- Capítulo 5- Arquitectura de interconexión para la integración de dispositivos IoT con plataformas IoT

En este capítulo, se propone una arquitectura de interconexión que extiende la arquitectura propuesta en el capítulo 3 para permitir la integración de los dispositivos heterogéneos con las plataformas estándar oneM2M y FIWARE. La nueva arquitectura aplica los principios de interconexión definidos por el ETSI, para la implementación de una entidad proxy de interconexión (IPE) que permite la interacción y comunicación bidireccional entre los dispositivos heterogéneos y las plataformas. En particular, se presenta una descripción detallada del funcionamiento de la arquitectura, así como su verificación a través de un caso de uso enfocado al dominio de transporte y logística. Para ello, se describe en profundidad el proceso de implementación de la arquitectura de acuerdo con el escenario control de acceso, tráfico y asistencia operacional perteneciente al caso de uso INTER-LogP. El capítulo está relacionado con el objetivo. **O5.**

- Capítulo 6- Sistema para la detección de caídas de personas mayores basada en IoT y Big Data

En este capítulo se evalúa la arquitectura a través de un segundo caso de estudio enfocado al dominio de la salud. Para ello se describe la implementación y puesta en marcha de un sistema que permite la monitorización en tiempo real de la detección de caídas de personas mayores en ambientes internos. El capítulo también contiene un análisis cualitativo y cuantitativo de la efectividad del sistema para la detección de caídas. Este capítulo concluye con un análisis del trabajo relacionado. El capítulo está relacionado con el objetivo **O5**.

- Capítulo 7- Conclusiones y líneas de trabajo futuro

En este capítulo se presentan las conclusiones principales extraídas en la elaboración de la tesis, así como el trabajo futuro que puede ser desarrollado.

Capítulo 2
Estado del arte

«Si tan sólo conociésemos la magnificencia del 3, 6 y 9,
entonces tendríamos una de las claves del universo. La vida es y
siempre seguirá siendo una ecuación imposible de ser resuelta,
pero tiene ciertos factores que podemos llegar a conocer»

Nikola Tesla (1856-1943)

2.1 Introducción

El *Internet of Things* (IoT) es un paso más en la evolución del Internet impulsado por la inclusión de objetos físicos combinados con la capacidad de proporcionar servicios más inteligentes al entorno a medida que se dispone de más datos. Varios dominios de aplicaciones que van desde el cuidado de la salud y ciudades inteligentes a transporte y logística ya están empezando a beneficiarse de los conceptos de IoT. Sin embargo, existen desafíos asociados con IoT, como la interoperabilidad que deben ser direccionados en el camino hacia la visión de las "cosas" que son capaces de comunicarse entre sí. Para resolver este desafío se necesita la comunicación e interconectividad de las "cosas" independientes de las tecnologías, protocolos de comunicación y formatos de datos subyacentes. Esto les permitirá participar activamente en Internet, intercambiando información sobre sí mismos y su entorno.

En este capítulo se proporciona un estado del arte exhaustivo del marco tecnológico en el cual se ha desarrollado este trabajo de investigación, los temas principales que se abordan son: Las características, componentes fundamentales de construcción, arquitecturas de referencia y aplicaciones potenciales de IoT, así como también los principales desafíos en el dominio de IoT. También se identifica y se discute la interoperabilidad en el ámbito de IoT, su definición desde diferentes perspectivas de las comunidades académicas y de la industria. Finalmente, se profundiza en la interoperabilidad de dispositivos en IoT, como uno de los principales desafíos para la adopción de esta tecnología. Se hace hincapié

en los niveles de interoperabilidad utilizados para abordar este desafío: técnico, sintáctico y semántico. Los lectores interesados encontrarán referencias a publicaciones técnicas para un mayor detalle.

2.2 Internet de las Cosas (IoT)

2.2.1 Evolución desde una visión

IoT es un concepto emergente que involucra un número cada vez mayor de objetos heterogéneos e inteligentes de la vida cotidiana (p.ej. vehículos, sensores de salud, ropa, equipos industriales, cerraduras inteligentes, luces inteligentes, teléfonos inteligentes, etc.) como se muestra en la Figura 2.1.

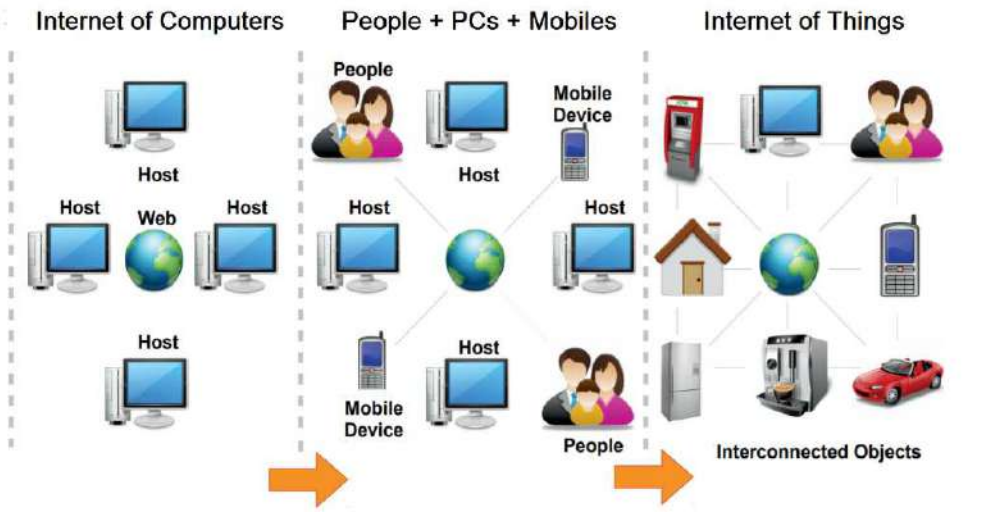
Figura 2.1: Representación de IoT

Este concepto fue acuñado por primera vez por Kevin Anston en el año 1999 centrándose principalmente en las tecnologías RFID (del inglés *Radio Frequency Identification*). Describió una visión en la que las computadoras serían capaces de recopilar datos sin ayuda humana y convertirlos en información útil, lo que sería posible con tecnologías como sensores y RFID, que permiten a las computadoras observar, identificar y comprender el mundo (Kevin Ashton, 2009).

Desde entonces, el concepto ha evolucionado, y hoy en día IoT abarca muchas otras tecnologías, incluidas redes inalámbricas de sensores(WSN, por sus siglas

en inglés *Wireless Sensor Networks*), comunicaciones de campo cercano (NFC, por sus siglas en inglés *Near Field Communication*), redes de área del cuerpo (BAN, por sus siglas en inglés *Body Area Network BAN*), comunicaciones de máquina a máquina (M2M, por sus siglas en inglés *machine-to-machine*) y otras redes de área personal (PAN, por sus siglas en inglés *Personal Area Network*), como Wi-Fi, Bluetooth, etc. (Al-Fuqaha *et al.*, 2015).

IoT es un paso más en la evolución de Internet, el cual comenzó como el *Internet of Computers*, una red global con servicios como la *World Wide Web* (WWW), lo que hizo que Internet fuera más popular y estimulara el rápido crecimiento de la *Web of Things* (WoT). En los últimos años, Internet se ha transformado en un *Internet of People* que crea conceptos como *Social Web* (o Web 2.0), donde el contenido es creado y consumido por personas conectadas. Esta tendencia se ha evidenciado por el crecimiento exponencial de los servicios de redes sociales como, por ejemplo, Facebook, Instagram, Twitter, Hotmail, entre otros, que junto con el crecimiento explosivo del acceso a *mobile Internet* y las aplicaciones móviles, evidencian que Internet ha madurado significativamente, hasta el siguiente paso, *Internet of Things*. La Figura 2.2, ilustra esta evolución.



Fuente: Adaptado de (Perera et al. 2014)

Figura 2.2: IoT en la evolución de Internet

Si bien el término Internet of Things es relativamente nuevo, los conceptos centrales que subyacen al IoT han existido durante décadas. A finales de la

década de 1950, por ejemplo, las tecnologías como RFID y las redes de sensores ya se utilizaban en contextos industriales y de fabricación para el seguimiento de artículos de gran tamaño tales como grúas y ganado. La idea de la comunicación directa M2M tampoco es nueva, ya que es básica para la idea de Internet en la que servidores, enrutadores y clientes se comunican entre sí. M2M se refiere a aquellas soluciones que permiten la comunicación entre dispositivos del mismo tipo y una aplicación específica. Sin embargo, generalmente no permiten el intercambio amplio de datos o la conexión de los dispositivos directamente a Internet (Holler *et al.*, 2014).

Lo que el IoT representa es una evolución del uso de estas tecnologías existentes en términos del número y tipo de dispositivos, así como de la interconexión de redes de estos dispositivos a través de Internet. La mayoría de los dispositivos actualmente en Internet (p.ej. servidores, computadoras de escritorio, computadoras portátiles, tabletas y teléfonos inteligentes) fueron diseñados originalmente para ser parte de esta red, por lo que incorporan capacidades integradas de procesamiento, almacenamiento y red. En particular, lo que el IoT propone es expandir la conectividad a internet y la capacidad de computo a dispositivos cotidianos (p.ej. sensores, automóviles, electrodomésticos, receptores de audio/ video, detectores de humo, etc.) que habitualmente no se consideran computadoras, permitiendo que estos dispositivos generen, intercambien y consuman datos con una mínima intervención humana (Perera et al. 2014).

2.2.2 Definiciones de IoT

A pesar de los constantes avances hasta la fecha, en la academia, industria, organizaciones de tecnología y organizaciones de estandarización todavía no existe una definición única y universal de IoT. Muchas de ellas han evolucionado con el tiempo debido a la aparición de nuevas tecnologías y a una mejora en las tecnologías existentes, entre las más relevantes y con mayor aceptación podemos mencionar:

- La comisión Europea, a través del *Cluster of European Research Project on the Internet of Things – CERP-IoT* (2009), en su definición preliminar afirma que IoT vincula a los objetos del mundo real con el mundo virtual, permitiendo así en cualquier momento y en cualquier lugar la conectividad para cualquier "cosa".
- Por otra parte, Vermesan *et al.* (2009) percibe a IoT como una extensión de la Internet existente y sostiene que IoT permite que las personas y las cosas estén conectadas en cualquier momento, desde cualquier lugar con

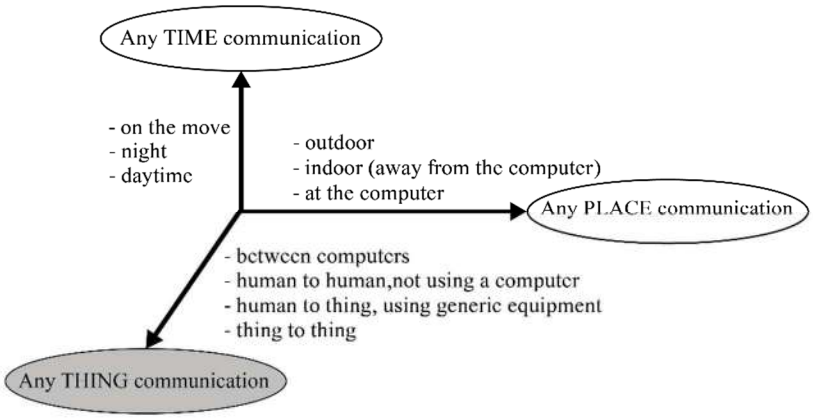
cualquier "cosa" y con cualquiera, idealmente utilizando cualquier ruta/red y cualquier servicio.

- Una definición diferente, centrada en la conectividad IP fue formulada por Zach Shelby & Carsten Bormann (2009), que afirman que IoT "*abarca todas las redes y los dispositivos integrados que están habilitados para IP y conectados a Internet de forma nativa, junto con los servicios de Internet que monitorizan y controlan esos dispositivos*".
- Otro punto de vista es el expuesto por el consorcio SAP (Haller, 2009) y hace referencia a IoT como un *"mundo donde los objetos físicos se conviertan en participantes activos de los procesos de negocios"*. En el cual, los dispositivos pueden interactuar y comunicarse entre sí mediante el intercambio de datos e información percibidos sobre el entorno, mientras reaccionan de manera autónoma a los eventos generados en el mismo – el entorno – ejecutando procesos que desencadenan acciones y crean servicios sin intervención humana.
- En el ámbito académico, una de las definiciones más aceptadas es la proporcionada por el consorcio CASAGRAS (2009), que afirma que IoT "*es una infraestructura de red global, que vincula objetos físicos y virtuales a través de la explotación de las capacidades de comunicación y captura de datos"*. Los autores resaltan la importancia de incluir en esta definición, los desarrollos de red y de Internet existentes y en evolución, que habilitarán la identificación de objetos, las capacidades de monitorización y conexión. En este sentido, IoT se convierte en una arquitectura para el despliegue de aplicaciones y servicios federados independientes, caracterizados por un alto grado de captura de datos autónomos, transferencia de eventos, conectividad de red e interoperabilidad.
- Por su parte, CERP-IoT (Vermesan *et al.*, 2010) amplía su primera definición y afirma que IoT es una infraestructura de red global dinámica con capacidades de autoconfiguración basadas en protocolos de comunicación estándar e interoperables. De acuerdo a esta definición, las "cosas" físicas y virtuales tienen identidades, atributos físicos y personalidades virtuales y utilizan interfaces inteligentes, y se integran perfectamente en la red de información.
- Una definición alternativa es provista por el *Internet Engineering Task Force – IETF*. (2012), donde define a IoT como: *"Una red de objetos interconectados únicamente direccionales, basada en protocolos de comunicación estándar"*.

- Desde la perspectiva de la normalización técnica, una de las definición de IoT ampliamente aceptadas es la propuesta por la *International Telecommunication Union* – ITU (2012), que define a IoT como una infraestructura global de la sociedad de la información, que permite ofrecer servicios avanzados mediante la interconexión de objetos (físicos y virtuales) gracias a la interoperabilidad de las tecnologías de la información y la comunicación (TICs) presentes y futuras.
- De igual manera, el *Institute of Electrical and Electronics Engineers -IEEE* propone dos definiciones de acuerdo al grado de complejidad de los escenarios de aplicación. En la primera definición centrada en los entornos y escenarios con un bajo nivel de complejidad enuncia que IoT es una red que conecta "cosas" identificables de forma única a Internet. Las "cosas" tienen capacidades de detección/actuación y programación. A través de la explotación de la identificación y detección únicas, la información sobre las "cosas" se pueden recopilar y el estado de las "cosas" se puede cambiar desde cualquier lugar, en cualquier momento, por cualquier "cosa" *(Minerva et al., 2015)*. La segunda definición centrada en escenarios más complejos combina las definiciones anteriores para describir a IoT como una red compleja, adaptable y de autoconfiguración que interconecta "cosas" a Internet mediante protocolos de comunicación estándar. Las "cosas" interconectadas tienen representación física o virtual en el mundo digital, capacidad de detección/actuación, una característica de programación y son identificables de manera única. La representación contiene información que incluye la identidad de las "cosas", el estado, la ubicación o cualquier otra información comercial, social o privada relevante. Las "cosas" ofrecen servicios, con o sin intervención humana, a través de la explotación de identificación única, captura de datos y comunicación, y capacidad de actuación. El servicio se explota mediante el uso de interfaces inteligentes y está disponible en cualquier lugar, en cualquier momento y para cualquiera que tenga en cuenta la seguridad (Minerva *et al.*, 2015) .

Definir el término IoT puede ser algo difícil porque puede tomar muchos enunciados dependiendo de las perspectivas tomadas (quién lo define, el dominio de aplicación y/o escenarios de implementación). Sin embargo, lo fundamental de IoT implica que los objetos que habitualmente no se consideran computadoras se pueden identificar de forma única (utilizando una IP o un identificador de recursos uniforme) en las representaciones virtuales. Además, los objetos en IoT

deben poder comunicarse e intercambiar datos entre sí (construyendo redes de objetos interconectados) en cualquier momento, desde cualquier lugar con cualquier dispositivo de forma autónoma generalmente utilizando cualquier ruta para proporcionar servicios avanzados a los usuarios finales. Esto es lo que se conoce como las tres dimensiones de la comunicación en IoT (Figura 2.3). Para ello, los estándares técnicos deben diseñarse en términos de las especificaciones de intercambio de datos, procesamiento y comunicaciones de la red.



Fuente:(Telecommunication Standarization Sector of ITU, 2012)

Figura 2.3: Dimensiones de la comunicación en IoT

2.2.3 Características de IoT

A partir de las definiciones expuestas, se puede establecer algunas características fundamentales que identifican a IoT, y que hacen de ella una infraestructura global para proporcionar servicios avanzados a los usuarios finales.

- **Objetos/*things*/dispositivos:** Un componente imperativo de IoT son los dispositivos denominados objetos o simplemente "cosas". Como ya se ha comentado, los objetos cotidianos, no solo las computadoras, son los principales actores de IoT. Estos tienen que ser identificables, reconocibles, localizables, direccionales y/o controlables. Estos atributos junto con las capacidades de detección/accionamiento (actuación), procesamiento, almacenamiento y de comunicación denotan a un *smart object- SO,* en adelante dispositivo IoT. Su importancia reside en las capacidades que tienen para hacer que los entornos físicos sean "inteligentes".

En un entorno inteligente (*smart environments*) los dispositivos IoT trabajan continuamente para hacer que la vida de los habitantes sea más cómoda (Youngblood *et al.*, 2005).

Los requisitos mínimos que han de cumplir los dispositivos IoT es que dispongan de capacidades de comunicación, detección y/o accionamiento (International Telecommunication Union, 2012). La capacidad de detección/accionamiento es la clave, y diferencia los dispositivos IoT de los dispositivos o entidades tradicionalmente considerados en los sistemas en red (p.ej., *router, hub, switch,* etc.); de hecho, estas capacidades abarcan las dos operaciones fundamentales (monitorización y actuación) que representan la interfaz y el acoplamiento entre el mundo físico y virtual.

- **Conectividad:** Una característica ampliamente aceptada es la presencia de una infraestructura de red o conectividad de red, que habilite la comunicación de los dispositivos IoT, su integración perfecta y un esquema de direccionamiento único. Esta infraestructura debe ser cualquier infraestructura de red que permita superar la idea de una Intranet de dispositivos separados. En IoT, el rango de opciones de conectividad aumentará exponencialmente y los desafíos de escalabilidad e interoperabilidad se mantendrán. En este contexto, las necesidades de comunicación cambiarán y se requerirán nuevas soluciones para satisfacer las demandas de conectividad de los dispositivos IoT emergentes.
- **Datos:** El verdadero valor de IoT radica en los datos asociados a cada dispositivo que se obtienen a partir de la interconexión entre ellos. Estos datos deben ser el primer paso hacia la acción y la inteligencia; sin embargo, solo la información no es suficiente, el análisis y el uso inteligente de estos datos, y la capacidad de vincularlos con otros datos, amplía su valor. En tal contexto, resulta adecuados mecanismos que habiliten dicha vinculación.
- **Autonomía:** Es una característica recurrente que caracteriza a los dispositivos IoT. Gracias a esta característica los dispositivos IoT deben tener la capacidad de realizar la mayoría de sus tareas sin la intervención humana o de otros dispositivos, y un grado de control sobre sus propias acciones y su propio estado interno según el contexto, las condiciones cambiantes o el entorno detectado (G Fortino *et al.*, 2012). Por ejemplo,

las cámaras de vigilancia pueden adaptar sus modos de operación (normales o infrarrojos) según sea de día o de noche.

- **Servicios:** Los servicios que pueden ser complejos o simples deben estar disponibles para interactuar con los dispositivos IoT, consultar su estado y cualquier información asociada a ellos, teniendo en cuenta las restricciones, como protección de la privacidad y coherencia semántica entre los dispositivos IoT y sus correspondientes objetos virtuales. Para ofrecer servicios dentro de las restricciones de los dispositivos IoT, las tecnologías en el mundo físico y en el virtual evolucionarán (International Telecommunication Union, 2012).
- **Heterogeneidad:** La heterogeneidad de los dispositivos IoT plantea problemas de interoperabilidad y es otra característica a menudo enfatizada y de primordial importancia a considerar para el desarrollo y adopción de esta tecnología. IoT aborda una gran diversidad de dispositivos, generalmente con capacidades de comunicación y computación muy básicas, que desafía la suposición de que cualquier dispositivo presenta una pila de protocolos completa. Si bien la visión de IoT requerirá avances sustanciales en varios campos su realización probablemente seguirá un proceso incremental, a partir de la interoperabilidad de dispositivos. Por lo tanto, se requiere el diseño de soluciones apropiadas que permitan la integración perfecta de dispositivos IoT a través de la coexistencia de los protocolos de comunicación, tecnologías subyacentes y el acceso a los recursos dentro de la solución de interconexión elegida. Esta característica es parte fundamental de esta tesis.

2.2.4 Componentes de IoT

Para entregar la funcionalidad de IoT, se requiere de varios componentes complementarios que en conjunto ayudan a reducir la brecha entre el mundo físico y virtual. Los principales componentes (Figura 2.4) que han sido identificados y descritos en diversos trabajos (Al-Fuqaha *et al.*, 2015; Gubbi *et al.*, 2013) se pueden resumir en:

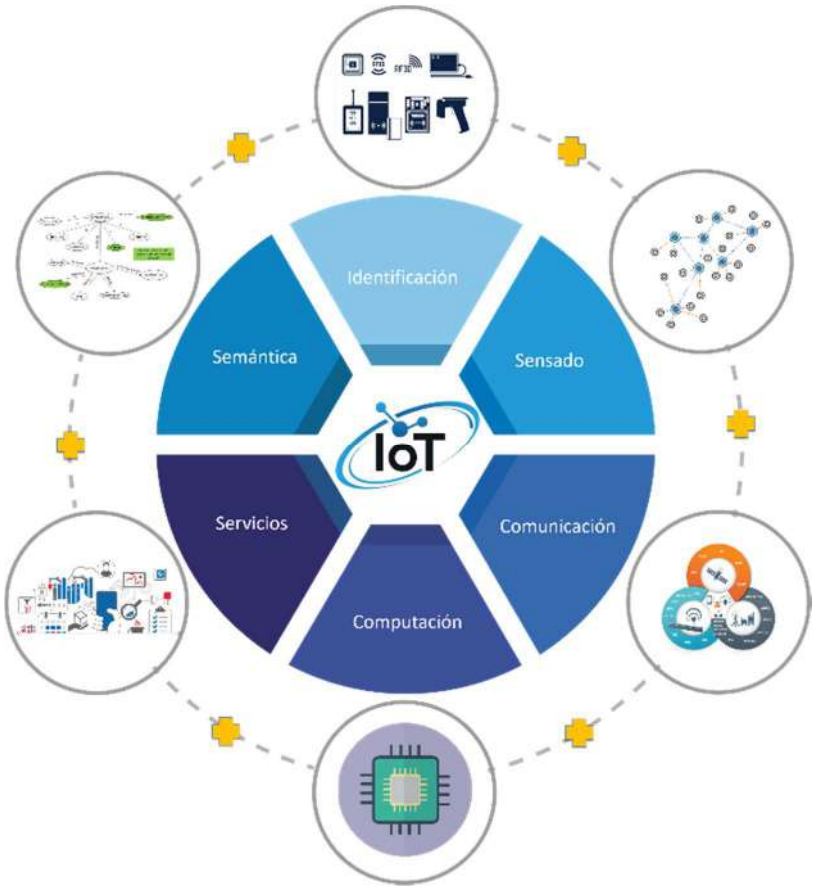


Figura 2.4: Elementos principales de IoT

- **Identificación:** La capacidad de identificar de forma única los dispositivos es fundamental para el éxito de IoT. Esto no solo permitirá identificar miles de millones de dispositivos, sino también controlar estos dispositivos remotamente a través de Internet. En la identificación es fundamental diferenciar entre el identificador del dispositivo y su dirección. El identificador del dispositivo hace referencia a su nombre, mientras la dirección se refiere a la dirección única del dispositivo dentro de una red de comunicaciones. Estos dos términos son muy diferentes entre sí porque dos o más dispositivos pueden tener el mismo nombre, pero siempre una dirección diferente y única.

Existen varios métodos que facilitan la identificación de dispositivos como los códigos de productos electrónicos (CPE, por sus siglas en inglés *Electronic Product Codes*) y los códigos ubicuos (uCode, por sus siglas en inglés *Ubiquitous Code*). El CPE está diseñado para ser almacenados en una etiqueta RFID, mientras que el uCode se puede utilizar con cualquier esquema de etiquetado, incluidos códigos de barras, código QR y etiquetas RFID (Prabhu, 2013). Otra posibilidad para la identificación es proporcionar la descripción de un dispositivo indicando sus características relevantes como por ejemplo si está equipado con comunicación inalámbrica, lo que significa que podría comunicar directamente su propia entidad. Por su parte, los métodos de direccionamiento de los objetos de IoT incluyen IPv6, IPv4 y 6LowPAN. En primer lugar, se utilizó IPv4 para asignar la dirección a los dispositivos. No obstante, la necesidad de direccionamiento de la gran cantidad de dispositivos de IoT proyecta a IPv6 como uno de los protocolos para soportar la operación de Internet en substitución de IPv4, debido a su espacio de direcciones de 128 bits.

- **Detección/actuación:** La detección de IoT consiste en recopilar datos de dispositivos relacionados dentro de la red. Los datos recopilados se envían a un almacén de datos, base de datos o al *cloud* para su almacenamiento y posterior procesamiento de manera que permitan tomar acciones específicas basadas en los servicios requeridos. Hay muchos dispositivos de detección como etiquetas RFID, sensores inteligentes, dispositivos de detección portátil, etc. Los dispositivos también pueden interactuar con el entorno de forma activa, es decir realizando acciones. Estos dispositivos se conocen como actuadores. En términos de tecnologías habilitadas, un tema clave para IoT es el desarrollo de medios apropiados para permitir interacciones con el entorno, en este sentido se espera que los componentes de detección estén también representados por las tecnologías de redes de sensores inalámbricas (WSN). De hecho, la capacidad de las WSN para detectar el entorno y auto organización en redes *ad hoc* representa una característica importante desde la perspectiva de IoT.
- **Comunicación:** Parte fundamental de IoT es lograr que dispositivos heterogéneos se comuniquen e interactúen entre sí. La comunicación permite que los dispositivos puedan enviar y recibir información. Generalmente los dispositivos IoT operan con un bajo consumo de energía y en entornos de enlaces de comunicación ruidosos y con altos niveles de

pérdidas de información. Por lo tanto, se requiere de protocolos de comunicación especializadas para realizar esta tarea de comunicación. Existen diversos protocolos de comunicación que facilitan la comunicación en IoT como ZigBee, LoRA (Alliance, 2015), Z-Wave, Wi-Fi, Bluetooth (McDermott-Wells, 2005), 6LowPAN y LTE-Advanced (Crosby & Vafa, 2013).

- **Computación:** La capacidad de cálculo de los datos recopilados de los dispositivos IoT está representado por las unidades hardware de procesamiento (p.ej. microcontroladores, microprocesadores, *system on chip (*SoC)) y el software de procesamiento. Entre las plataformas de hardware que permiten realizar el procesamiento en aplicaciones de IoT se destacan Arduino, Intel Galileo, Raspberry Pi, BeagleBone, T-Mote Sky, entre otros. En cuanto al software de procesamiento, los sistemas operativos son vitales, para proporcionar el procesamiento, ya que se ejecutan durante todo el tiempo de activación de un dispositivo. Entre ellos se destacan ContiKi RTOS(Dunkels *et al.*, 2004), TinyOS(Levis *et al.*, 2005), LiteOS (Cao *et al.*, 2008) y Riot OS(Baccelli *et al.*, 2013).
- **Servicios:** Los servicios de IoT pueden clasificarse en cuatro clases (Gigli & Koo, 2011): servicios relacionados con la identidad, servicios de agregación de la información, servicios basados en la colaboración y servicios ubicuos. Para cada aplicación un servicio IoT puede ser aplicado.
  - *Servicios relacionados con la identidad*: son la base de todos los demás servicios, se encargan de identificar la identidad de los dispositivos.
  - *Servicios de agregación de información*: recopilan y resumen las medidas sensoriales sin procesar.
  - *Servicios de colaboración*: actúan sobre los servicios de agregación de información y utilizan los datos obtenidos para tomar acciones y reaccionar en consecuencia.
  - *Servicios ubicuos*: proporciona servicios de colaboración en cualquier momento, en cualquier lugar, para cualquiera que los necesite. El objetivo final de toda aplicación IoT es alcanzar el nivel de servicios ubicuos.
- **Semántica:** La semántica en IoT se refiere a la capacidad de extraer conocimiento de los datos a través del uso de recursos e información de modelado para dar sentido al servicio solicitado. Este componente es

compatible con las tecnologías de la Web semántica, como el *Resource Description Framework* (RDF), la ontología *Web Ontology Language* (OWL) y el formato *Efficient XML Interchange (EXI)* (W3C, 2014).

2.2.5 Arquitecturas de referencia en IoT

Varios desafíos se interponen entre la idea conceptual de IoT y su despliegue en entornos reales, dada la cantidad de dispositivos y condiciones necesarias para que funcionen. De hecho, un punto de partida para la implementación exitosa de IoT está estrechamente relacionado con el establecimiento de una arquitectura de referencia (M. Wu *et al.*, 2010). Dependiendo de los diferentes usos que vayan a tener los dispositivos IoT se requiere contar con funcionalidades concretas. Por lo tanto, es necesaria una arquitectura modular y escalable capaz de soportar la incorporación de distintas funcionalidades y hacer frente al complejo escenario de IoT.

A lo largo de los últimos años, se han propuesto un número cada vez mayor de arquitecturas basadas en una estructura de múltiples capas (Figura 2.5) cada una de la cuales cumple una función específica. La arquitectura de alto nivel de tres capas (Figura 2.5 (a)) se introdujo en las primeras etapas de la investigación en esta área. Esta arquitectura es comúnmente aceptada e incluye la capa de percepción, red y aplicación (M. Wu *et al.*, 2010); las otras arquitecturas están compuestas por cuatro capas orientada a servicios (Li, S., Xu, L.D. & Zhao, 2015)(Figura 2.5 (b)) y por cinco capas basada en *Middleware* (Khan *et al.*, 2012; Kraijak & Tuwanut, 2015) (Figura 2.5 (c)).

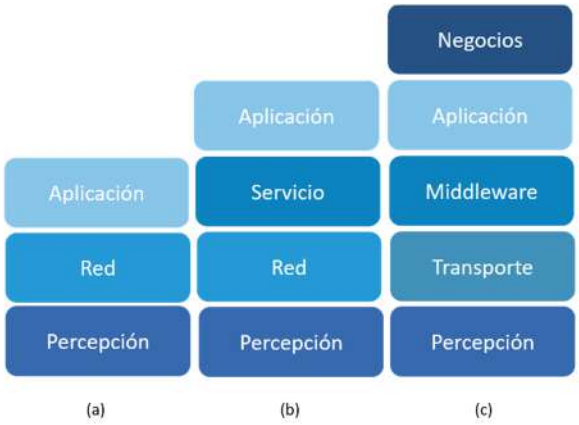


Figura 2.5: Arquitecturas de IoT

2.2.5.1 Arquitectura de tres capas

A continuación, se describe la funcionalidad de cada una de las capas de la arquitectura de tres capas. El rol de las capas de percepción, red y aplicación es el mismo en la arquitectura de tres y cuatro capas representadas en la Figura 2.5.

- **Capa de Percepción:** Esta capa inicial denominada percepción representa a los dispositivos físicos de IoT (sensores y actuadores). La principal responsabilidad de esta capa es la identificación de dispositivos, la recopilación de datos del entorno por parte de los dispositivos sensores y la transformación de estos datos en señales digitales. Esta capa también permite ejecutar acciones sobre el entorno a través de los dispositivos sensores.
- **Capa de Red:** La capa de red es responsable de conectarse a otros dispositivos físicos, dispositivos de red y servidores. Sus características también se utilizan para transmitir y procesar los datos de los sensores.
- **Capa de Aplicación:** En esta capa se encuadran las aplicaciones y servicios del usuario, que hacen un uso efectivo de la información procesada de los sensores. Adicionalmente, esta capa permite compartir datos con otras aplicaciones, servicios, sistemas y plataformas. Define varias aplicaciones en las que se puede implementar IoT, por ejemplo, transporte móvil, ciudad automatizada, casa inteligente, entre otras.

2.2.5.2 Arquitectura de cuatro capas

Considerando las características específicas de IoT, la arquitectura de cuatro capas además de las tres capas, incluye la capa de servicio que se corresponde a la capa *middleware* de la arquitectura de cincos capas. Esbozamos la función de esta capa restante.

- **Capa de Servicio:** La capa de servicio también denominada capa de *middleware* es responsable de almacenar los datos provenientes de la capa de transporte, extraer información de valor a partir del procesamiento y análisis de estos datos y tomar decisiones automáticas basadas en los resultados generados. Puede administrar y proporcionar un conjunto diverso de servicios a las capas inferiores.

2.2.5.3 Arquitectura de cinco capas

Finalmente, la arquitectura de cinco capas, incluye una capa adicional a la arquitectura de 4 capas, la capa de negocios.

- **Capa de Negocios:** Esta capa administra todo el sistema de IoT, apoya el proceso de toma de decisiones incluidas las aplicaciones, los modelos de negocio y la privacidad.

La tendencia de diseñar nuevas arquitecturas una y otra vez ha continuado. No obstante, definir una arquitectura de referencia de IoT que incorpore las mejores prácticas aceptadas del sector es un objetivo que ha encontrado un gran apoyo dentro de uno de los principales organismos de estándares de la comunidad de IoT, el Comité técnico del ETSI. A continuación, se proporciona una descripción general de la arquitectura propuesta por el ETSI, que está directamente relacionada con el patrón de comunicación D2G descrito en la sección 2.4.2.3.

2.2.5.4 Arquitectura de red jerárquica para la conectividad escalable M2M

El comité técnico del ETSI establecido en enero de 2009, ha definido una arquitectura de referencia denominada arquitectura de red jerárquica para la conectividad escalable M2M, que identifica las entidades funcionales y los puntos de referencia relacionados. Describe una arquitectura basada en recursos que se puede usar para el intercambio de datos y eventos entre dispositivos, que involucran comunicaciones a través de redes sin la intervención humana.

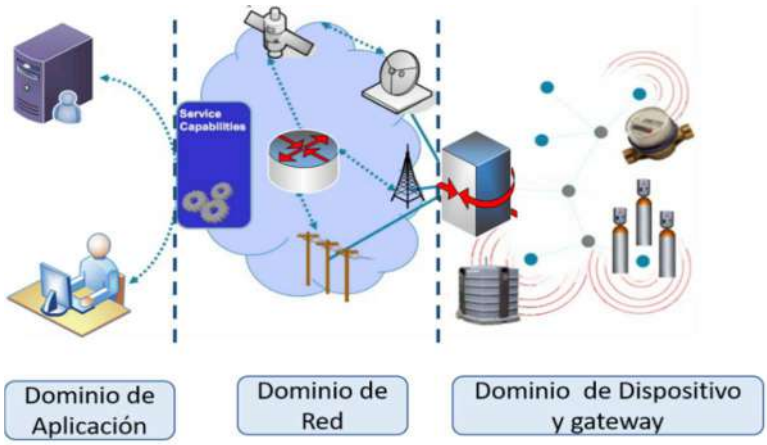


Figura 2.6: Arquitectura M2M ETSI.

La Figura 2.6 muestra la arquitectura de alto nivel del ETSI, la cual se compone de tres dominios: dominio de dispositivo y *gateway*, dominio de red y el dominio de aplicación.

- **Dominio de dispositivos y *gateway*:** consiste en varios dispositivos, una o más *gateways* M2M/IoT y una o más redes de área, que proporciona la conectividad a nivel de capa física y MAC entre dispositivos y *gateways*. Un dispositivo puede conectarse al servidor remoto de dos maneras: i) una conexión directa con la red de acceso o ii) pasando a través de la red de área y el *gateway*. Varias tecnologías de corto alcance como las descritas en la sección 2.4.3.1.1.1, pueden ser utilizadas dentro de la red de área. El *gateway* actúa como proxy entre los dispositivos y el dominio de la red al proporcionar rutas eficientes para enviar datos hacia y desde el servidor remoto. También puede ejecutar aplicaciones M2M y posee inteligencia local para recopilar y procesar datos.

- **Dominio de red:** consiste en redes de acceso y redes centrales, y proporciona conectividad entre los *gateways* y las aplicaciones. Las redes centrales proporcionar funciones como conectividad IP, control y gestión de la red, itinerantica e interconexión con otras redes. Los ejemplos incluyen, pero no se limitan a, xDSL, WLAN, satélite, GSM, WiMAX y LTE.

- **Dominio de aplicación:** consiste del servidor y de las capacidades de servicio M2M, las cuales garantiza las interacciones entre las aplicaciones y los dispositivos. El servidor es el punto de integración para almacenar todos los datos recopilados por los dispositivos, lo que hace que estos datos estén disponibles en tiempo real para las aplicaciones M2M.

2.2.6 Dominios de aplicación de IoT

Uno de los motores más importantes de la innovación y el desarrollo de IoT son sus aplicaciones (Vongsingthong & Smanchat, 2014). IoT tiene enormes potenciales para desarrollar nuevas aplicaciones inteligentes en casi todos los campos. Esto se debe principalmente a su capacidad para realizar la detección (p.ej. recopilar información sobre fenómenos naturales, parámetros médicos o hábitos de los usuarios) y ofrecer servicios personalizados. Estas aplicaciones apuntan a mejorar la calidad de vida y tendrán un impacto profundo en la economía y en

la sociedad. Las diversas aplicaciones se pueden agrupar en tres dominios principales (Borgia, 2014):

- **Dominio industrial***:* Actividades que involucran transacciones financieras o comerciales entre empresas, organizaciones y otras entidades.
- **Dominio del bienestar sanitario:** Actividades relacionadas con el desarrollo de servicios inteligentes para apoyar y mejorar las actividades de las personas y la sociedad.
- **Dominio de ciudad inteligente:** Actividades relacionadas con la sostenibilidad de nuestras ciudades y la calidad de vida de las personas.

Cada domino no está aislado de los otros, pero está parcialmente superpuesto ya que algunas aplicaciones se comparten. A continuación, se describen ejemplos indicativos de cada dominio.

2.2.6.1 Dominio industrial - Transporte y logística

El dominio de transporte y logística es un ejemplo claro de la incursión de IoT impulsado, en este caso, por el desarrollo de múltiples *Inteligent Transport Systems* (ITS) (Harden, 2017). Entre ellos se incluyen, sistemas de ciudades inteligentes, geolocalización de vehículos y mercancías, sistemas de análisis de congestión de carreteras, sistemas de asistencia en viajes, entre otros. Estos sistemas se basan en una gran variedad de sensores, tanto fijos como móviles, instalados en los automóviles, trenes, autobuses, bicicletas, carreteras y/o rieles de las ciudades, así como también en los productos transportados (R. Khan *et al.*, 2012).

En el dominio del transporte, estos sensores podrían enviar información relevante a los sitios de control de tráfico y vehículos de transporte para apoyar la gestión eficiente del tráfico, ayudar en la gestión de los depósitos, proporcionar al turista información de transporte adecuada, entre otros. En cuanto a la logística, tecnologías de procesamiento de información como RFID y NFC permitirán realizar una monitorización en tiempo real, casi de todos los eslabones de la cadena de suministro, desde el diseño de productos básicos hasta el servicio postventa, pasando por la compra de materias primas, la producción, el transporte, el almacenamiento, la distribución y la venta de productos. Así, los sensores pueden recopilar, por ejemplo, datos sobre la posición y el estado de los bienes y/o vehículos de transporte, tanto en ruta como dentro de centros logísticos o centros de carga. Los datos recopilados facilitan a las empresas de transporte la planificación y optimización, por ejemplo, de rutas más efectivas que

permitan el ahorro de energía durante las operaciones de los vehículos de transporte.

En cuanto a soluciones IoT aplicadas al dominio de transporte y logística, se resalta INTER-LogP, un caso de uso enfocado a la interoperabilidad de los ecosistemas logísticos portuarios derivado del proyecto Europeo INTER-IoT (INTER-IoT, 2016).

2.2.6.2 Dominio del bienestar sanitario - Cuidado de la Salud

Otra aplicación en la que el uso de las tecnologías de IoT se está volviendo cada vez más popular, es el sector sanitario, donde la monitorización continua y personalizada de la salud se ha vuelto esencial para mejorar la calidad de vida y el bienestar de los pacientes. De hecho, IoT desde una perspectiva de empoderamiento es una piedra angular de la transformación digital de la asistencia sanitaria. De acuerdo a estimaciones recientes de Aruba Networks, (2017), casi dos tercios (60%) de las organizaciones de atención de la salud en todo el mundo han comenzado a implementar la tecnología IoT y a conectar los dispositivos IoT a sus redes y para 2019, se espera que el 87% de estas organizaciones adopten esta tecnología ubicua. Además, se espera que para 2020, el valor del mercado de la salud inteligente global alcance los $ 169.30 mil millones.

El IoT a través de una amplia gama de sensores especializados, junto con otros dispositivos médicos interconectados generalmente mediante redes inalámbricas, proporcionan múltiples servicios innovadores que facilitan la toma de decisiones médicas, la prevención de enfermedades y el diagnóstico *ad hoc* en caso de emergencias (Hu *et al.*, 2013). Por ejemplo, se pueden utilizar para mejorar las *assisted living solutions* como apoyo a las personas ancianas y con capacidades especiales. Los pacientes llevaran sensores médicos para controlar parámetros como la temperatura corporal, la presión arterial y la actividad respiratoria. Se utilizarán otros sensores, ya sean portátiles (p.ej. acelerómetros, giroscopios, podómetros, etc.) o fijos (báscula, proximidad) para monitorizar y recopilar datos útiles de las actividades de los pacientes en sus entornos de vida.

El acceso generalizado y en tiempo real a la información alimentada por los sensores permitirán a los médicos tratantes, servicios de emergencia y cuidadores realizar una monitorización remota avanzada y podrán realizar acciones de respuesta rápida cuando sea necesario. En tal situación, los sensores también pueden proporcionar información de contexto de la ubicación geográfica de los pacientes.

La interconexión de estos sensores heterogéneos podría proporcionar una imagen completa de los parámetros de salud. Esto no solo permitirá reducir significativamente las admisiones al hospital o la tasa de visitas al departamento de emergencias, sino que también mejorará de forma masiva la optimización del flujo de trabajo. A la vez que también permite una recuperación más corta y mejor, la autonomía, la independencia, el compromiso y la satisfacción de los pacientes (Islam *et al.*, 2015). Además, estos datos se pueden combinar con otros datos distribuidos geográficamente y que pertenecen a múltiples proveedores involucrados en la atención de un paciente (hospitales, instituciones médicas especializadas, farmacias de guardia, centros de emergencia, centros ambulatorios) mediante la creación de vistas virtuales de datos compartidos (Shi *et al.*, 2016) que puede exponerse a los usuarios finales a través de una interfaz común, sobre la cual se pueden construir varios servicios complejos para apoyar el cuidado integral de la salud de los pacientes.

2.2.6.3 Dominio de Ciudad Inteligente

Otro sector de aplicaciones relevante se relaciona con las *SmartCities,* este término se utiliza para denotar el *cyber-physical system (CPS)* emergente mediante el despliegue de infraestructura moderna en un ciudad, que permite ofrecer servicios avanzados e innovadores a los ciudadanos(Zanella *et al.*, 2012; Ahlgren *et al.*, 2016). Las tecnologías de IoT pueden encontrar diversas aplicaciones en escenarios de *SmartCities.* Como caso de estudio, las tecnologías de IoT se pueden utilizar para proporcionar sistemas avanzados de monitorización ambiental. A través de una gran cantidad de dispositivos miniaturizados y auto gestionables desplegados en el ambiente de manera distribuida, será posible monitorizar los fenómenos y procesos naturales (p.ej. temperatura, viento, lluvia, etc.), así como también áreas críticas sin requerir la presencia de operadores humanos (p.ej. en áreas volcánicas). El procesamiento de la información en tiempo real, junto con la capacidad de los dispositivos sensores para comunicarse entre sí, proporciona una plataforma sólida para detectar y monitorizar anomalías que pueden poner en peligro la vida humana y animal. En tal sentido, IoT puede permitir el desarrollo de una nueva generación de sistemas de monitorización y soporte a la toma de decisiones, proporcionando una granularidad mejorada y capacidades en tiempo real sobre las soluciones existentes.

Otro caso en el que la capacidad de detección de los dispositivos IoT es compatible con la monitorización ambiental está representado por la detección de la calidad del aire (Kotsev *et al.*, 2016). Los sensores pueden detectar el nivel de contaminación del aire, recuperar información sobre el *smoke* (p.ej. El nivel de

dióxido de carbono, PM10, PM2.5, ozono, dióxido de nitrógeno, etc.) y entregar dicha información a las agencias de salud. Cuando un conjunto de sensores detecta la posible presencia de contaminación del aire, se envía una alarma directamente a los ayuntamientos de cada ciudad en poco tiempo, junto con otros parámetros que son útiles en la toma de decisiones y el apoyo, como la descripción de las zonas sujetas a la contaminación, la posible presencia de personas, etc. Muchos otros escenarios como, por ejemplo, los de salud pueden beneficiarse de la capacidad de acceder a datos de contaminación en tiempo real sobre áreas libre de contaminación para realizar recomendaciones de rutas saludables a los pacientes que padecen enfermedades causadas por la contaminación del aire como las enfermedades cardiovasculares y pulmonares.

En cuanto a experimentación y despliegue de soluciones aplicadas al cuidado de la salud, se resalta INTER-Health (Pace *et al.*, 2017), también derivado del proyecto Europeo INTER-IoT. INTER-Health tiene por objetivo desarrollar un sistema integrado de IoT para monitorizar el estilo de vida de las personas de manera descentralizada y en movilidad, mediante la integración de las plataformas existentes UniversAAL (Hanke *et al.*, 2011) and BodyCloud (Giancarlo Fortino *et al.*, 2014), principalmente para prevenir problemas de salud resultantes de trastornos de la conducta alimentaria y de la actividad física.

En general, el alcance de IoT es extremadamente amplio y en varios dominios de aplicación existen varios escenarios donde las tecnologías de IoT pueden ser aplicadas para el desarrollo de nuevos servicios o para mejorar las soluciones disponibles. Por lo tanto, se espera que la adopción de IoT esté fuertemente impulsada por las necesidades del mercado y por su dinámica. Un ejemplo representativo de las aplicaciones anteriormente descritas se muestra en la Figura 2.7.

Figura 2.7: Ejemplos de dominios de aplicaciones de IoT

2.2.7 Desafíos en el desarrollo de IoT

IoT puede ofrecer enormes beneficios económicos a través de la implementación de diferentes servicios y aplicaciones enfocados a varios aspectos de nuestra vida, pero también enfrenta muchos desafíos técnicos clave. A continuación, se describen brevemente algunos de ellos.

- **Conectividad:** La conectividad perfecta consumirá extremadamente poca energía, tendrá una amplia cobertura y será capaz de transmitir grandes cantidades de datos. Desafortunadamente, esta conectividad perfecta no existe. La heterogeneidad de los dispositivos involucrados lo hace aún más difícil, ya que se pueden esperar muchas interconexiones físicas diferentes. El desafío de IoT es abordar esta heterogeneidad en la fase de diseño identificando los posibles casos de uso en cada sector.

- **Arquitectura del sistema:** La naturaleza de múltiples dominios de aplicación de IoT dificulta la creación de una arquitectura y una aplicación única y excepcional. En pocas palabras, las soluciones para un determinado dominio no son aplicables a otros, ya sea debido a requisitos funcionales o debido a diferencias de hardware. El desafío en IoT es construir tales arquitecturas, teniendo en cuenta la portabilidad, la integración y la conectividad. Además, las arquitecturas deben ser abiertas y, utilizar tecnologías estándar, a fin de no restringir a los usuarios para que utilicen soluciones propietarias de extremo a extremo (Chen *et al.*, 2014).

- **Interoperabilidad e integración:** IoT se caracteriza por una gran heterogeneidad en cuanto a los dispositivos que participan en un sistema, que presentan capacidades muy diferentes en términos de los componentes de comunicación y computación; debido a que están construidos por diferentes fabricantes. Su integración perfecta solo puede ser posible si se construyen sobre estándares abiertos. Puede haber múltiples estándares para las mismas áreas (p.ej. diferentes estándares de redes inalámbricas), pero se debe establecer la interoperabilidad entre ellos (p.ej., a través de *gateways* entre diferentes redes físicas)

- **Complejidad computacional y de almacenamiento:** Los dispositivos IoT generarán cantidades masivas de datos. Estos datos pueden ser continuos o en ráfagas, y pueden estar estructurados o no estructurados.

Para extraer el máximo provecho de estos datos, deben ser transportados, almacenados, procesados y analizados (Lee & Lee, 2015). Estas operaciones ejercen una enorme presión sobre las redes, el almacenamiento y la infraestructura computacional. El desafío en IoT es desarrollar y mantener tales infraestructuras complejas.

- **Seguridad, confianza y privacidad:** La penetración de IoT en la vida diaria enfatiza la necesidad de soluciones seguras adecuadas. Por un lado, la gran cantidad de dispositivos involucrados dificulta el diseño de un sistema completamente seguro, ya que hay muchos puntos de ataque potencial. Entonces, cualquier solución debe poder manejar datos de una gran cantidad de dispositivos sin causar ninguna pérdida de los mismos, garantizar las medidas de seguridad adecuadas para los datos transmitidos y evitar el acceso externo no autorizado (R. Khan *et al.*, 2012). De acuerdo con el índice de privacidad de IoT TRUSTe. (2015), solo el 20% de los usuarios de Internet estuvieron de acuerdo en que los beneficios derivados de los dispositivos inteligentes superan cualquier inquietud de privacidad. Si bien IoT ha dado lugar a importantes avances tecnológicos, la confianza y aceptación de IoT dependerán del equilibrio adecuado entre promover esta innovación y garantizar la seguridad y privacidad de los usuarios.

De los desafíos enumerados, es obvio que el desarrollo del IoT depende del progreso de varias disciplinas, incluidas las redes inalámbricas de sensores, la computación en la nube, computación en la niebla y la seguridad informática. En esta tesis doctoral abordamos el desafío de la interoperabilidad entre dispositivos IoT que está directamente relacionado con la integración y conectividad.

2.3 Interoperabilidad en el ámbito de IoT

El potencial transformador de IoT permitirá alcanzar unos ingresos tan relevantes como los 11 mil millones de dólares anuales para el 2025 (Mckinsey Global Institute, 2015). Pero no será fácil, como ya se ha mencionado IoT es capaz de desarrollarse en muchas áreas y tiene tantos usos que su propia diversidad puede ser el principal obstáculo para su crecimiento.

En un entorno donde operan innumerables dispositivos heterogéneos de diferentes tipos y perfiles técnicos, desarrollar la capacidad de comunicarse entre ellos y hacerlos fácilmente accesibles es uno de los principales desafíos en IoT.

Por esta razón la interoperabilidad en IoT surge como una necesidad importante para el desarrollo y adopción de esta tecnología disruptiva. Para entender su relevancia, a continuación, profundizaremos en su concepto. Primero, desde un punto de vista amplio se introduce de manera general el concepto y niveles de interoperabilidad presentes en la literatura.

Posteriormente, se describe la interoperabilidad de los dispositivos físicos con énfasis en la heterogeneidad de los dispositivos, los patrones de comunicación para conectar dispositivos IoT y los niveles de interoperabilidad a través de los cuales se puede resolver el problema de la interoperabilidad de los dispositivos en IoT. Finalmente, se analizan las tecnologías y protocolos de comunicación, los formatos de serialización y estándares presentes en cada uno de estos niveles de interoperabilidad.

En general, el problema de la interoperabilidad es complejo y existe desde 1988 (Department of Defense of United States of America, 1988); y posiblemente incluso antes. Debido a su relevancia e importancia, se han establecido varias definiciones que tienen diferente significado según el contexto de aplicación, perspectiva y dominio en cuestión. Dentro de estas definiciones, destacan:

- El IEEE (Radatz J, Geraci A, 1990) argumenta que la interoperabilidad es "*la capacidad de los sistemas para intercambiar datos y usar la información"*. Esto implica que dos sistemas son interoperables si pueden entenderse entre sí y usar la información intercambiada de cada uno.
- Desde una visión más amplia la *International Organization for Standardization and the International Electrotechnical Commission- ISO/IEC* (2015), define la interoperabilidad como "*la capacidad de comunicar, ejecutar programas o transferir datos entre varias unidades funcionales de una manera que requiera que el usuario tenga poco o ningún conocimiento de las características únicas de esas unidades*". De acuerdo a esta definición la interoperabilidad debe garantizar la transparencia en la comunicación y transferencia de datos.
- El *Department of Defense*- DoD (ATIS, 2015), por su parte, hace una definición de la interoperabilidad desde la perspectiva de los sistemas de comunicaciones electrónicas y define la interoperabilidad como *"la condición lograda entre sistemas de comunicaciones electrónicas o los elementos de equipos de sistemas de comunicaciones electrónicas cuando la información o los servicios pueden intercambiarse directa y satisfactoriamente entre ellos y/o sus usuarios"*.

Aunque existen muchas definiciones, todas coinciden en los principios básicos, que destacan las condiciones necesarias y suficientes para lograr la interoperabilidad: ***intercambio*** y ***usabilidad*** de datos e información (Diallo *et al.*, 2011).

Con base en estas condiciones, la interoperabilidad en el ecosistema de IoT se puede definir como la capacidad de dos o más entidades (es decir, dispositivos, aplicaciones o sistemas) heterogéneos para comunicarse, intercambiar datos y usar la información entre sí. En otras palabras, la interoperabilidad significa relacionar dos entidades y eliminar cualquier incompatibilidad entre ellas. La incompatibilidad es un concepto fundamental utilizado en el dominio de IoT. Es el obstáculo para establecer una interoperación perfecta (*seamless*). Por lo tanto, la interoperabilidad no significa que las entidades involucrados en el intercambio de información deban estar completamente estandarizadas (menos que sean iguales). Este es, y probablemente será, un escenario algo improbable. Pero a pesar de la falta de estándares compartidos, las entidades sí necesitan colaborar.

En general, la capacidad de dos entidades para interoperar se puede presentar a través de diferentes niveles interoperabilidad. Una de las clasificaciones más importantes de los niveles de interoperabilidad para los sistemas técnicos se denomina modelo de interoperabilidad conceptual (Tolk *et al,* 2004). Este modelo define seis niveles de interoperabilidad: sin conexión (sin interoperabilidad entre sistemas), técnica (conectividad básica y conectividad de red), sintáctica (intercambio de datos con una estructura definida), semántica (comprensión del significado de los datos), pragmática/dinámica (aplicabilidad de la información) y conceptual (visión compartida del mundo). Pantsar *et al.* (2012) define una clasificación similar que incluye los siguientes niveles de interoperabilidad: conexión, comunicación, semántica, dinámica, conductual y conceptual, que son equivalentes a los niveles definidos por *Tolk*: técnico, sintáctico, semántico, pragmático/dinámico y conceptual, respectivamente. Con la finalidad de habilitar la interoperabilidad entre plataformas IoT, el proyecto Europeo INTER-IoT, (2018) también define una clasificación de seis niveles de interoperabilidad: dispositivo, redes, middleware, servicio de aplicaciones, datos y semántica. El trabajo de Lappeteläinen *et al.* (2008) por su parte, propone una clasificación más específica para la interoperabilidad entre sistemas que consta de tres niveles: dispositivo, servicio e información.

Como se puede apreciar, existe mucha investigación en cuanto a la definición de niveles de interoperabilidad; estas clasificaciones expresan un rango de niveles de abstracción de interoperabilidad, que van desde un nivel muy alto, conceptual

e información, hasta un nivel bajo, técnico y dispositivo. En la práctica los niveles altos tienden a tratarse tácitamente, asumiendo que lo que falta se ha acordado de alguna manera previamente o se describe en la documentación para el desarrollador. De la misma manera los niveles más bajos tienden a ser abordados empíricamente, recurriendo a alguna herramienta o tecnología que se ocupa de lo que es necesario para hacer que la interoperabilidad funcione.

IoT puede considerarse como un sistema en red altamente dinámico y distribuido. Desde una perspectiva a nivel de sistema, el problema de la interoperabilidad en el ecosistema IoT surge en cada capa de la pila de protocolos debido a la heterogeneidad de dispositivos, redes y aplicaciones. Existen diversas aplicaciones dentro de IoT. Además, existen diferentes tecnologías de comunicación, que tienen como objetivo proporcionar soluciones a un conjunto de requisitos específicos para diferentes escenarios de aplicaciones. Por lo tanto, no existe y puede que nunca haya una tecnología que cumpla con los escenarios de todas las aplicaciones. Además, el aumento del nivel de heterogeneidad debido a la inclusión de dispositivos con componentes de comunicación y computación limitadas desafía la suposición de que cualquier dispositivo presente una pila de protocolos completa. Por lo tanto, la interoperabilidad es un problema de múltiples capas que debe garantizarse comenzando desde las capas más bajas de la pila de redes hasta la capa de aplicación. El problema de la interoperabilidad requiere de un enfoque más pragmático e integral, cuyo primer objetivo es tener dispositivos inteligentes que puedan implementarse en cualquier entorno, sin limitaciones técnicas o comerciales, con una red troncal de interconexión interoperable que les permita comunicarse con otros dispositivos inteligentes a su alrededor (Gubbi *et al.*, 2013) e integrarlos en cualquier plataforma IoT. Siendo la base para la creación de sistemas interoperables.

2.4 Interoperabilidad de dispositivos en IoT

La interoperabilidad de dispositivos se puede definir como la capacidad de dos o más dispositivos para conectarse, intercambiar información e interactuarn entre sí. (D. C. Yacchirema & Palau, 2016).

La interoperabilidad como ya se ha comentado, es un aspecto clave en el desarrollo de sistemas IoT en el que se pretende que interactúen sin fisuras diferentes dispositivos heterogéneos. Si además nos centramos en el caso particular, y en el objetivo de implementar la interacción entre dispositivos y plataformas IoT, el reto es aún mayor ya que no es una tarea fácil debido a la heterogeneidad

inherente en las capacidades de memoria, procesamiento, comunicación y formato de datos admitido por cada dispositivo. Se comentará con detalle en la siguiente sección.

2.4.1 Heterogeneidad de los dispositivos

IoT está compuesto por una variedad de dispositivos, incluso más que el internet tradicional (Noura *et al.*, 2018). En los ecosistemas actuales de IoT, los distintos dispositivos y aplicaciones operan en sus propias plataformas sin la compatibilidad adecuada con dispositivos de diferentes proveedores. Por ejemplo, una pulsera inteligente no puede interactuar con una bombilla inteligente sin la aplicación privada relevante proporcionada por el mismo proveedor. Como resultado, se establece islas de funcionalidad IoT que conducen a una intranet de dispositivos en lugar de un internet de dispositivos. Estos problemas se presentan debido a la gran variedad y heterogeneidad de los dispositivos. Los dispositivos IoT son heterogéneos al estar basados en hardware muy variado de plataformas y redes que se traducen en distintas capacidades de memoria, procesamiento, comunicación y formato de datos.

- **Capacidades de memoria y procesamiento**

La mayoría de los dispositivos que conforman IoT presentan ciertas limitaciones en términos de recursos, que incluyen CPU y memoria.

Tabla 2.1: Visión general de la clasificación de *constrained devices*

Clase	Tamaño de datos (e.g., RAM)	Tamaño de código (e.g., Flash/ROM)	CPU	Ejemplo
Clase 0 - C0	<< 10 kB	<< 100 kB	≤ 10 MHz	Etiquetas RFID
Clase 1 - C1	~ 10 kB	~ 100 kB	≤ 50 MHz	SensorTag, Arduino
Clase 2 - C2	~ 50 kB	~ 250 kB	≤ 1000 MHz	RedBee EconoTAG, Raspberry PI, CrossBow Telos

(kB = 1024 bytes, MB =1024 kB)

A tal efecto, el término "*constrained devices*" (C Bormann *et al.*, 2014) se introdujo recientemente para definir una categoría de dispositivos conectados con restricciones de recursos estrictas en comparación con los dispositivos (p.ej., computadores de escritorio, *tablets*, teléfonos móviles, etc.) que fueron diseñados originalmente para ser parte de Internet.

La Tabla 2.1 presenta una visión general de la clasificación de *constrained devices* de acuerdo a las capacidades de sus recursos.

- **Dispositivos Clase 0:** Generalmente son etiquetas RFID, sensores de bajo costo y actuadores que poseen restricciones severas, tanto en memoria como en las capacidades de procesamiento, que la mayoría de los dispositivos. Estas restricciones impiden que se comuniquen directamente a Internet de una manera segura. Sin embargo, pueden comunicarse a Internet con la ayuda de dispositivos con mayor capacidad (p. ej. dispositivos de Clase 2) que actúan como servidores *proxy*, *gateways* o servidores. Los dispositivos clase 0 son configurados con un conjunto de datos muy pequeño para su funcionamiento.
- **Dispositivos Clase 1**: Están bastante limitados en el espacio de código y en las capacidades de procesamiento. No pueden comunicarse fácilmente con otros nodos de Internet que utilizan una pila de protocolos completa como HTTP, Transport Layer Security (TLS) y protocolos de seguridad relacionados y basados en representaciones de datos como XML (*Extensible Markup Language*). Sin embargo, pueden usar una pila de protocolos diseñada específicamente para nodos restringidos como el *Constrained Application Protocol* (CoAP) y participar en conversaciones significativas sin la ayuda de un nodo intermediario. Por lo general, estos protocolos son livianos, energéticamente eficientes y consumen menor ancho de banda en comparación con los protocolos tradicionales de Internet. Además, estos protocolos pueden proporcionar soporte para la seguridad requerida en una red. Por lo tanto, los dispositivos Clase 1 pueden integrarse en una red IP. No obstante, deben ahorrar memoria, espacio de código y, en muchos casos, también energía.
- **Dispositivos Clase 2:** Tienen más recursos, pero aún están muy limitados en comparación con los dispositivos tradicionales. Estos dispositivos se caracterizan por tener recursos "suficientes" para ejecutar software basado en sistemas operativos tradicionales (OS), como Linux, o BSD. Estos dispositivos incluyen computadoras de una sola placa (SBC, por sus siglas en inglés, *single-board computer).*

- **Capacidades de comunicación:** Actualmente los dispositivos pueden utilizar varias soluciones de comunicación desde comunicaciones ubi-

cuas tradicionales (p. ej., comunicaciones celulares 3/4/ 5G y las tecnologías Wi-Fi) hasta soluciones propietarias no estándar (p.ej., Sigfox, LoRa), pasando por tecnologías comerciales (p. ej., Bluetooth, NFC, ANT +) así como protocolos y mecanismos de comunicación tradicionales para sensores y actuadores que están abriendo nuevas posibilidades para IoT (p.ej., ZigBee, WirelessHart, IEEE802.11 ah, ZigBee, etc.). Adicionalmente, los dispositivos también pueden utilizar diferentes soluciones para el enrutamiento (p.ej. RPL, CORPL, etc.), y encapsulación (p. ej., 6LowPAN, 6Lo, Thread, etc.).

Desde una perspectiva del hardware, el estado tecnológico de la técnica también es muy heterogéneo, debido a que se han desarrollado varias soluciones comerciales (p.ej., Arduino, TelosB, pcDuino, Libelium waspmote, etc.).

- **Formato de datos:** El mundo de los datos provenientes de los dispositivos y su semántica también presentan una gran heterogeneidad debido a la presencia de diferentes formatos (p.ej., WSDL, JSON, XML, W3C *Semantic Sensor Network* XG, SWE, ASN.12, etc.)

2.4.2 Patrones de comunicación de dispositivos

Para conseguir que diferentes dispositivos IoT consigan intercambiar información y por lo tanto ser interoperables es imprescindible realizar un estudio de los patrones de comunicación utilizados en el ecosistema IoT. El estándar RFC 7452 definido por el *Internet Architecture Board (IAB)* (H. Tschofenig, J. Arkko, D. Thaler, 2015) describe tres patrones de comunicación para los dispositivos IoT.

2.4.2.1 Comunicación de dispositivo a dispositivo (D2D)

El patrón D2D permite la comunicación e intercambio de información entre dispositivos de manera directa, en lugar de hacerlo a través de un componente intermediario como un servidor de aplicación o un conmutador. La comunicación se puede establecer a través de diferentes tipos de redes, incluidas, entre otras, las redes IP o Internet. No obstante, los dispositivos también utilizan redes de corto alcance como Bluetooth 4.0, ZigBee o Z-Wave para establecer comunicaciones directas de dispositivo a dispositivo.

La principal característica de este patrón de comunicación es la adherencia de los dispositivos a un protocolo de comunicación particular para lograr el intercambio de datos con los requisitos de calidad de servicio (*Quality of Service* QoS)

requeridos. Se usa comúnmente en aplicaciones como sistemas de automatización del hogar, donde dispositivos con requisitos de velocidad de datos relativamente bajos se comunican entre sí mediante el envío de pequeños paquetes de datos. Los dispositivos residenciales de IoT, como las bombillas, los termostatos, los interruptores de luz, son ejemplos de dispositivos de IoT típicos de este tipo de aplicaciones. En la Figura 2.8 se presenta un ejemplo de este modelo de comunicación, en el cual un interruptor inteligente se comunica con una bombilla inteligente a través de Bluetooth 4.0.

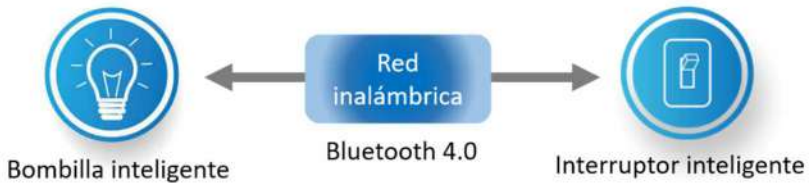


Figura 2.8: Ejemplo del patrón de comunicación D2D

Este patrón de comunicación plantea un conjunto de desafíos derivados de la falta de compatibilidad de protocolos asociados a los dispositivos y utilizados por los diferentes fabricantes. Por ejemplo, los dispositivos con protocolo ZigBee no pueden comunicarse entre sí con los dispositivos con protocolo Z-wave. En consecuencia, estos problemas limitan la elección de dispositivos y la experiencia de los usuarios.

2.4.2.2 Comunicación de dispositivo al cloud (D2C)

En el patrón de comunicación dispositivo al *cloud*, los dispositivos IoT se comunican directamente a un servicio *cloud* de Internet como por ejemplo un proveedor de servicios de aplicaciones para intercambiar datos y controlar el tráfico de mensajes. Este patrón normalmente aprovecha las comunicaciones preexistentes, como las conexiones convencionales por cable (*Ethernet*) o inalámbricas (*Wi-Fi*), para establecer la conexión entre los dispositivos y la red IP, que en última instancia se conecta al servicio *cloud*. Un ejemplo de este patrón de comunicación se ilustra en la Figura 2.9, en la cual los sensores de temperatura y monóxido de carbono (CO) transmiten los datos capturados en el entorno a una plataforma específica localizada en el *cloud*. Existen varios protocolos y estándares basados en IP dedicados a la comunicación entre dispositivos y plataformas de servicio como HTTP, REST y CoAP.

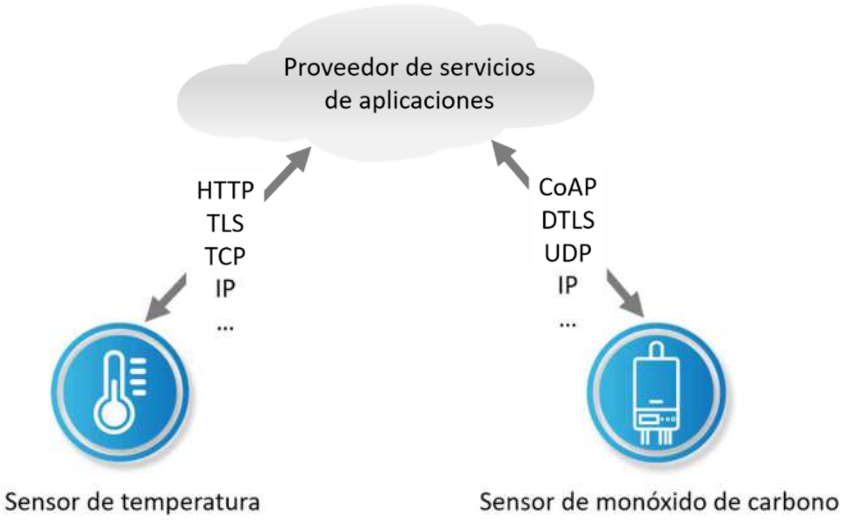


Figura 2.9: Ejemplo del patrón de comunicación D2C.

El termostato *Nest Learning* desarrollado por *Nest Labs* (Nest, 2018) y el televisor *Samsung SmartTV* son dos de los dispositivos IoT más populares que utilizan este patrón de comunicación. En el caso del termostato *Nest Learning*, el dispositivo transmite datos a una aplicación en el *cloud* encargada de almacenarlos y analizarlos. Este procesamiento permite al usuario obtener información detallada sobre el consumo de energía en su hogar. Además, esta conexión con el servicio *cloud* permite al usuario obtener acceso remoto a su termostato a través de un teléfono inteligente o una interfaz web. De igual manera, el televisor *Samsung SmartTV* utiliza una conexión a Internet para transmitir información referente a las visualizaciones del usuario a *Samsung* para su análisis y para habilitar las funciones de reconocimiento de voz interactivas del televisor.

Aunque la comunicación del dispositivo al *cloud* resuelve los problemas del patrón D2D, y agrega valor al usuario final al extender las capacidades del dispositivo más allá de sus características nativas, este patrón también plantea desafíos que deben ser considerados. Por una parte, al depender de la red tradicional, el ancho de banda y los recursos de la red limitan el rendimiento de este patrón de comunicación, por lo que es necesario optimizar la estructura de la red. Por otra parte, la mayoría de dispositivos IoT son dispositivos pequeños y restringidos; equipados con baterías de baja capacidad que transmiten a través de enlaces de radio de baja potencia. Estas restricciones impiden que estos dispositivos puedan conectarse a Internet y enviar los datos directamente al *cloud*. Además, las redes

compuestas por estos dispositivos son propensas a la pérdida de mensajes. Por lo que el uso de protocolos como TCP /IP y HTTP en las capas de transporte y aplicación, respectivamente, es ineficiente para los dispositivos, por lo que se necesitan protocolos optimizados para un buen rendimiento. El IETF ha especificado una alternativa a HTTP para dispositivos y redes restringidas, el protocolo CoAP, que se implementa en la parte superior de UDP como capa de transporte en lugar de TCP, como lo hace HTTP. El funcionamiento en detalle de este protocolo se describe en la sección 2.4.3.1.1.3. Adicionalmente, debido a que con frecuencia, el dispositivo y el servicio *cloud* son del mismo proveedor, este patrón de comunicación presenta problemas de interoperabilidad al intentar integrar dispositivos de diferentes fabricantes, lo que limita o impide el uso de un servicio alternativo de otro proveedor. Este bloqueo a su vez abarca otras facetas de la relación con el proveedor, como la propiedad y el acceso a los datos.

2.4.2.3 Comunicación de dispositivo a gateway (D2G)

El patrón de comunicación D2C descrito en la sección anterior es conveniente para dispositivos que explotan tecnologías de radio ampliamente disponibles en el mercado como Wi-Fi basado en IEEE 802.11. Sin embargo, para tecnologías de radio menos disponibles como IEEE 802.15.4 es necesario proporcionar una funcionalidad de capa de aplicación adicional. En tal sentido, a diferencia de los patrones de comunicación analizados al momento, el patrón de comunicación D2G se basa en un *middleware* o proxy operando en un *gateway*.

Un middleware es un software de aplicación que abstrae la complejidad y heterogeneidad de las redes de comunicación subyacentes, hardware, sistemas operativos, lenguajes de programación, etc., para permitir a una entidad interactuar o comunicarse con otras entidades, que no fueron diseñados originalmente para conectarse (Razzaque *et al.*, 2016).

En este patrón de comunicación, el dispositivo IoT se conecta a través de un *gateway* que actúa como un intermediario entre el dispositivo y el servicio *cloud* proporcionando funciones de traducción de datos y/o protocolos, así como también seguridad. Las cuales mejoran la flexibilidad y seguridad de la red de dispositivos IoT, a la vez que reducen el consumo de energía de estos dispositivos, debido a que se migra una parte del cálculo a la capa de aplicación. Un ejemplo de este patrón de comunicación se puede observar en la Figura 2.10.

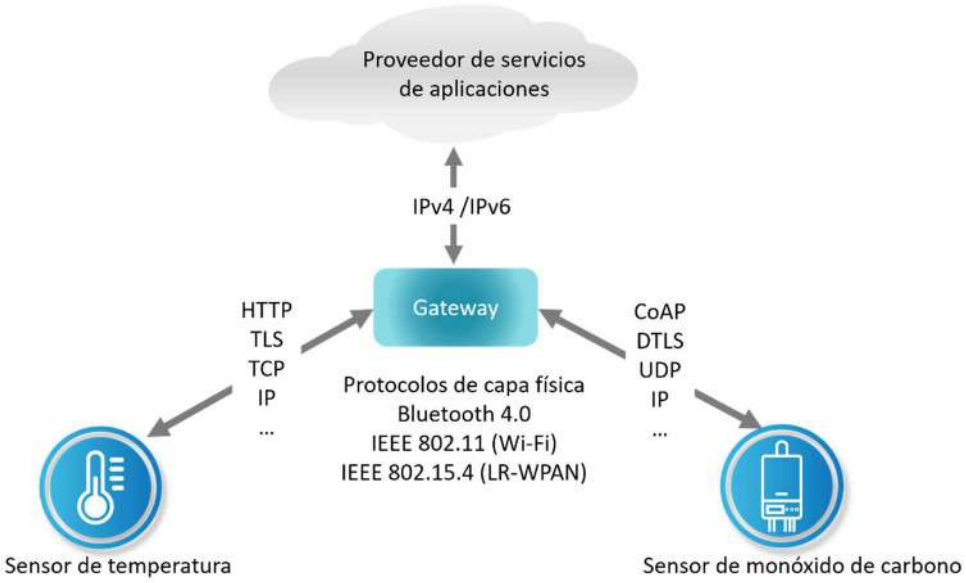


Figura 2.10: Ejemplo del patrón de comunicación D2G.

Este tipo de comunicación generalmente funciona gracias al uso de tecnologías de comunicación como Wi-Fi, que se usa ampliamente y puede cubrir varios metros. No obstante, recientemente, se ha lanzado el protocolo LoRaWAN dirigido a transmitir mensajes a largas distancias que pueden alcanzar los 20 kilómetros.

Este patrón, por ejemplo, se puede aplicar en el dominio de salud personal, donde un dispositivo móvil inteligente puede actuar como un *gateway* para comunicarse con una pulsera inteligente, cifrar y transmitir sus datos a su aplicación en el *cloud*. Otro ejemplo representativo de un *gateway,* es el concentrador *SmartThings* (SmartThings, 2018), que admite comunicaciones ZigBee y Z-Wave para comunicarse con ambas familias de dispositivos. Así los usuarios pueden acceder a los datos de los dispositivos mediante una aplicación de teléfono inteligente y una conexión a Internet. Además, un *gateway* permite soportar tanto el protocolo IPv6 como IPv4. Este último para mantener la compatibilidad con proveedores de aplicación heredados, mientras mantiene la conectividad con dispositivos restringidos que solo admiten el protocolo IPv6. En tal sentido, este patrón de comunicación permite la mayor flexibilidad, evitando la necesidad de actualizar los dispositivos heredados para admitir el protocolo IPv6.

Aunque la evolución de los protocolos de comunicación hace posible que los dispositivos puedan conectarse directamente a Internet, el uso de *gateways* puede implementar funcionalidades adicionales que los dispositivos incorporados no pueden implementar o proporcionar directamente, por ejemplo, agregación de datos, análisis y procesamiento de los datos, etc. Siendo esto, una ventaja más de este patrón de comunicación.

Adicionalmente, este patrón de comunicación se alinea a los cincos aspectos de interoperabilidad que caracterizan a los sistemas de IoT (físico, enlace, red, extremo a extremo y datos)(Bauer, Bui, *et al.*, 2013), dado que en lugar de centrarse en una pila de comunicación específica, proporciona un patrón transversal del cual se pueden derivar una o más pilas de comunicación. Debido a las ventajas inherentes que aporta este patrón de comunicación, el mismo será utilizado en la definición de la arquitectura propuesta en el capítulo 3.

2.4.3 Niveles de Interoperabilidad

Con base en las características heterogéneas mencionadas en la sección 2.4.1 y considerando el patrón D2G, el reto de la interoperabilidad entre dispositivos IoT se puede abordar a 3 niveles o capas; interoperabilidad técnica, sintáctica y semántica. (Figura 2.11).

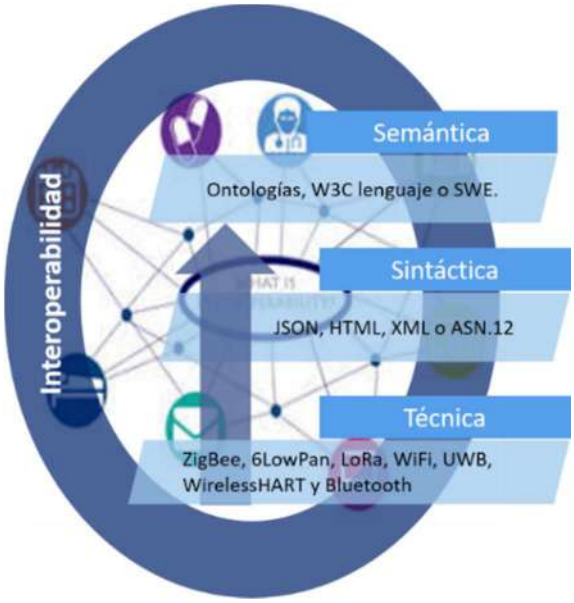


Figura 2.11: Niveles de interoperabilidad asociados a los dispositivos IoT

Estos niveles permiten hacer frente a la fragmentación actual en IoT al garantizar que los dispositivos i) compartan información y servicios; ii) proporcionen y reciban servicios de otros dispositivos; iii) interactúen entre sí u con otras aplicaciones para compartir información y iv) se integren en plataformas IoT.

- **Interoperabilidad técnica** (van der Veer & Wiles, 2008) **:** A menudo se centra en los **protocolos de comunicación y en la infraestructura necesaria (componentes de hardware y software)** para permitir la conectividad de dispositivos heterogéneos; sin conectividad no hay comunicación. Aquí los dispositivos solo pueden intercambiar mensajes sin interpretar su contenido interno.
- **Interoperabilidad sintáctica** (Molina, 2014) **:** se asocia con los **formatos de los datos** de los mensajes de los dispositivos, ya que estos deben tener una sintaxis y codificación bien definida. Aquí los dispositivos pueden interpretar el contenido de los datos dentro de la estructura del mensaje. Para evitar cualquier posible ambigüedad durante la interpretación de los datos, es necesario utilizar formatos estandarizados.
- **Interoperabilidad semántica** (van der Veer & Wiles, 2008; Ganzha *et al.*, 2017; IERC, 2011)**:** en este nivel, los dispositivos involucrados son capaces de **interpretar el contenido y el significado de los datos** intercambiados. Las ontologías y las tecnologías semánticas son medios para facilitar la interoperabilidad semántica. La falta de semántica explícita no ha sido un problema en los sistemas de comunicación tradicionales, como los teléfonos y las computadoras en general, porque la interoperabilidad a nivel semántico ha sido resuelta por usuarios humanos que se comunican entre sí mediante el uso de dispositivos. Sin embargo, en IoT, los dispositivos también se convierten en usuarios y, por lo tanto, necesitan comunicarse directamente entre sí e interpretar el significado de la información en tiempo de ejecución.

Si no se alcanza la interoperabilidad técnica y sintáctica, el intercambio de datos es imposible, a pesar de la posible interpretación de datos sin ambigüedad que se puede establecer a través de la interoperabilidad semántica. No obstante, la interoperabilidad entre dispositivos IoT puede ser un problema también en la representación semántica de los datos.

El desarrollo de la presente tesis doctoral, se centra en la interoperabilidad técnica y sintáctica, no obstante, resulta interesante exponer también la interoperabilidad semántica.

A continuación, se analiza cada uno de los tipos de interoperabilidad.

2.4.3.1 Interoperabilidad Técnica

Para formar una red interoperable, el primer requisito es la conectividad de los dispositivos ya sea de manera inalámbrica o cableada, es decir que los dispositivos tengan acceso a los medios compartidos, que pueden ser la misma frecuencia de radio o conectividad física a través de cable. En tal sentido, el uso de estándares en la capa física y MAC se plantea como una solución viable hacia la integración de los dispositivos en IoT. No obstante, usar los mismos estándares en diferentes implementaciones, no garantiza la comunicación de dispositivos.

Por ejemplo, en la Figura 2.12, el estándar IEEE 802.15.4 puede ser utilizado por diferentes dispositivos. Sin embargo, este estándar puede ser implementado de varias maneras, posiblemente no interoperables, debido a que este estándar solo ofrece la comunicación en las capas de red inferiores: física y MAC (Mainetti *et al.*, 2011). Mientras a nivel de capa de red varias tecnologías *Low Power Wireless Personal Area Networks* (LoWPAN) de la competencia, como ZigBee o 6LowPAN pueden ser utilizadas.

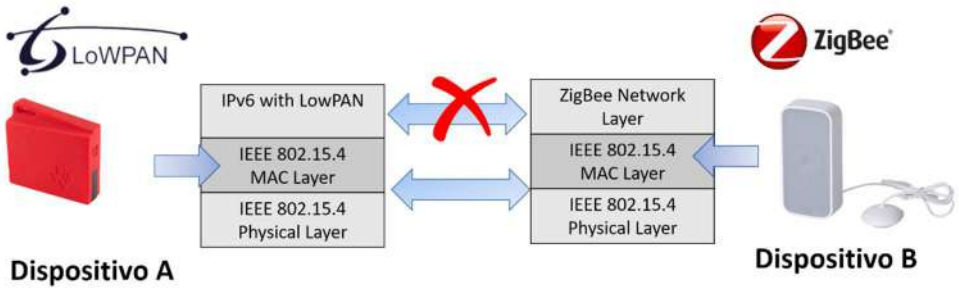


Figura 2.12: Ejemplo de dos dispositivos no interoperables que utilizan el mismo estándar.

Con base en el modelo de comunicación D2G los equipos de radio multimodo constituyen la principal solución técnica hacia la interoperabilidad de los diversos dispositivos heterogéneos que utilizan diferentes medios de comunicación y redes. Como se ha comentado, los teléfonos inteligentes, los concentradores domésticos (enrutador, *gateways*) son ejemplos representativos, debido a que implementan radios multimodo y admiten diversas tecnologías de comunicación (p.ej., 802.15.4, Wi-Fi, ZigBee, Bluetooth, LoRa, etc.). Los equipos radio multimodo actúan como puentes proporcionando la funcionalidad interoperable deseada. Una vez que los dispositivos están conectados, la mayor parte de la funcionalidad

de la interoperabilidad requerida puede implementarse en el software. Por ejemplo un túnel puede desarrollarse para establecer la interoperabilidad entre dispositivos 6LowPAN y ZigBee si los dispositivos son compatibles con la tecnología 802.15.4.

Varias tecnologías y protocolos de comunicación que tienen una amplia adopción en la actualidad son candidatos para la conectividad de dispositivos en IoT. Cada uno presenta ventajas y desventajas específicas.

2.4.3.1.1 Tecnologías y protocolos de comunicación relacionados

Esta sección analiza los principales protocolos utilizados para IoT de acuerdo a la capa donde operan: física/enlace, red/transporte y aplicación.

2.4.3.1.1.1 Capa física y de enlace

A continuación, se describen algunos protocolos ampliamente adoptados para la conectividad de los dispositivos en IoT. Presentamos las principales características necesarias para facilitar la comprensión de nuestro trabajo, incluyendo el rango, el ancho de banda, el uso de energía, la interoperabilidad y seguridad.

- **Wi-Fi:** se basa en el estándar IEEE 802.11 fue diseñado con la intención de reemplazar al estándar Ethernet IEEE 802.3 (ampliamente utilizado y basado en cable) mediante la comunicación inalámbrica a través de banda sin licencia. Su objetivo era proporcionar conectividad inalámbrica de corto alcance, fácil de implementar y fácil de usar, con interoperabilidad entre proveedores. Su rango se encuentra entre 30(interior)-100 metros (exterior)(Ray,2018). Actualmente, existen varias especificaciones: IEEE 802.11a/b/g/n. 802.11n ofrece el rendimiento de datos más alto, pero al costo del alto consumo de energía, por lo que los dispositivos IoT solo pueden usar 802.11 b o g por razones de ahorro de energía. Para reducir el consumo energético de los dispositivos IoT, Alianza Wi-Fi publicó el nuevo estándar para IoT, WiFi HaLow (802.11ah).
- **WiFi HaLow (802.11ah):** Se introdujo específicamente para abordar los problemas de alcance y potencia de IoT al combinar las ventajas de las tecnologías de comunicación de red de sensores de baja potencia y Wi-Fi (N. Ahmed *et al.*, 2016). WiFi HaLow cuenta con un nuevo diseño en las capas física y MAC que opera en la banda exenta de licencia ISM de 900 MHz para proporcionar un rango extendido (radio de 1 Km) con

bajos requisitos de energía, en comparación con las bandas actuales de 2,4 y 5 GHz (Akeela & Elziq, 2017).

En WiFi HaLow, el uso de energía se optimiza mediante la adopción de protocolos MAC que brindan menor potencia al utilizar formatos de trama más pequeños y al hacer que el tiempo de encendido de los sensores sea más corto, y eso se debe a la velocidad de datos mínima de 100kbps. Así, los dispositivos conectados pueden ser más pequeños y baratos, con posibilidad de funcionar durante meses o incluso años (~ 10 años), tras realizar una sola carga de su batería (Knyazev *et al.*, 2017). Sin embargo, 802.11ah requerirá puntos de acceso inalámbricos (o radios dentro de los puntos de acceso) y clientes especializados en comparación con Wi-Fi estándar.

- **Bluetooth:** Se basa en el estándar IEEE 802.15.1. Es una tecnología de comunicación inalámbrica de bajo consumo y bajo costo adecuada para la transmisión de datos entre dispositivos móviles en un rango corto (8–10 m). El estándar Bluetooth define una comunicación de red de área personal (PAN) y opera en la banda de 2.4 GHz. La velocidad de datos en varias versiones de Bluetooth varía de 1 Mbps a 32 Mbps (Bluetooth, 2003). La especificación de Bluetooth está supervisada por el *Bluetooth Special Interes Group (SIG)*. La versión de ultra bajo consumo y bajo costo de este estándar se denomina Bluetooth Low Energy (BLE o Bluetooth Smart) que surgió en la especificación Bluetooth 4.0 (y cuenta con las actualizaciones 4.1 y 4.2) y que ha sido diseñada para dispositivos con recursos limitados de baja potencia (SIG, 2018). Es decir es útil para dispositivos que transfieren una pequeña cantidad de datos a baja velocidad de datos dentro de rangos relativamente cortos(Bello *et al.*, 2017). Actualmente existe una adopción generalizada de esta tecnología debido a que la mayoría de los teléfonos inteligentes de gama alta son compatibles con BLE, lo que permite una integración sin problemas y sin infraestructura de los dispositivos BLE con Internet (Raza *et al.*, 2017). Esto convierte a BLE en una potencial tecnología disruptiva para IoT.

  Es importante destacar que la versión 4.2 permite que los sensores Bluetooth Smart accedan a Internet directamente a través de una conexión 6LoWPAN, lo que hace posible utilizar la infraestructura IP existente para administrar dispositivos BLE. Sin embargo, esta no es la ruta más fácil para los usuarios que deseen conectar sus dispositivos BLE, por lo

que el SIG de Bluetooth introdujo un kit de herramientas de acceso público que permite a los usuarios crear sus propias puertas de enlace de Internet para sus dispositivos (BLuetooth-SIG, 2018). Con la última versión 5.0, el SIG de Bluetooth reafirmó su posición dentro del escenario competitivo de IoT debido a que la nueva especificación cuadruplica el rango de alcance, duplica la velocidad y aumenta la capacidad de transmisión de datos con el mismo bajo consumo de energía (Bluetooth, 2019).

- **ZigBee:** También opera en el espectro de comunicación inalámbrica de 2.4 GHz, pero tiene un alcance más largo que BLE de hasta 100 metros. ZigBee se basa en el estándar de protocolo de comunicación IEEE 802.15.4 y se utiliza para redes de área personal o PAN, originalmente diseñado para entornos residenciales e industriales. IEEE 802.15.4 (Institute of Electrical and Electronics Engineers, Inc., 2003) es una colección de estándares que define las características de las capas físicas y MAC para redes de área personal inalámbricas de baja velocidad (LR-WPAN, por sus siglas en inglés, *Low-Rate Wireless Personal Area Networks)*. Estos estándares forman la base de especificaciones para protocolos de comunicaciones de alto nivel como ZigBee y 6LowPAN. Los estándares LR-WPAN proporcionan velocidades de datos desde 40 Kb/s a 250 Kb/s y funcionan a frecuencias de 868/915 MHz y 2.4 GHz a velocidades de datos altas y bajas, respectivamente(Baronti *et al.*, 2007).

  Zigbee proporciona comunicaciones inalámbricas bidireccionales y de bajo costo con muy bajo consumo de energía, y puede alcanzar un rendimiento de datos de hasta 250 kbps (a pesar de que la velocidad de datos tiende a ser mucho más baja en aplicaciones prácticas). ZigBee estandariza las capas superiores de la pila de protocolos: red y aplicación. Entre las funcionalidades proporcionadas por la capa de red se encuentran el enrutamiento *multihop*, el descubrimiento y mantenimiento de rutas, la seguridad y la conexión/salida de una red, con la consiguiente asignación de direcciones cortas (de 16 bits) a los dispositivos recientemente incorporados. Mientras que la capa de aplicación proporcionar un marco para el desarrollo y la comunicación de aplicaciones distribuidas (ZigBee Alliance, 2018).

  La capa de aplicación es el nivel más alto de la especificación ZigBee y consta de objetos de aplicación (APO, por sus siglas en inglés, *Application object*), subcapa de aplicación (APS, por sus siglas en inglés, *Application*

sub-layer) y un objeto de dispositivo Zigbee (ZDO, por sus siglas en inglés, *ZigBee Device Object*). Los servicios de administración de dispositivos y redes son proporcionados por el ZDO, que permite que los APO se descubran entre sí y se organicen en una aplicación distribuida (define el rol de cada uno de los dispositivos). Los servicios de transferencia de datos y seguridad son proporcionados por la APS a los APO y ZDO. Los APO se encargan de definir el perfil de la aplicación. Permite definir hasta 240 objetos para la comunicación o aplicaciones diferentes. Además, ZigBee incluye un proveedor de servicios de seguridad (SSP, por sus siglas en inglés, *Security Service Provider*). No obstante, cada capa (MAC, red y aplicación) puede protegerse y compartir claves de seguridad. El SSP se inicializa y configura a través del ZDO y requiere la implementación del AES (*Advanced Encryption Standard*).

La Figura 2.13 ilustra la arquitectura de ZigBee soportada sobre la pila 802.15.4.

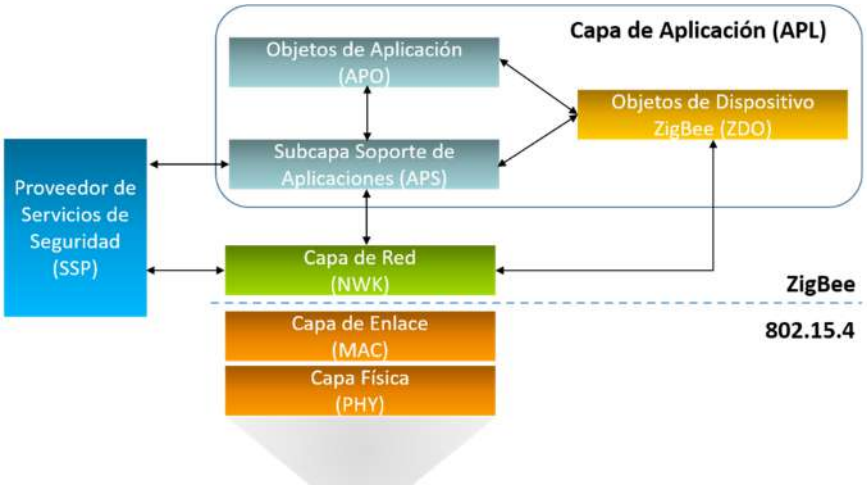


Figura 2.13: Arquitectura ZigBee (Capa de Red y Aplicación)

Zigbee es compatible con topologías de estrellas, árboles y mallas (Sun *et al.*, 2007). En las cuales pueden funcionar tres nodos lógicos (Figura 2.14):

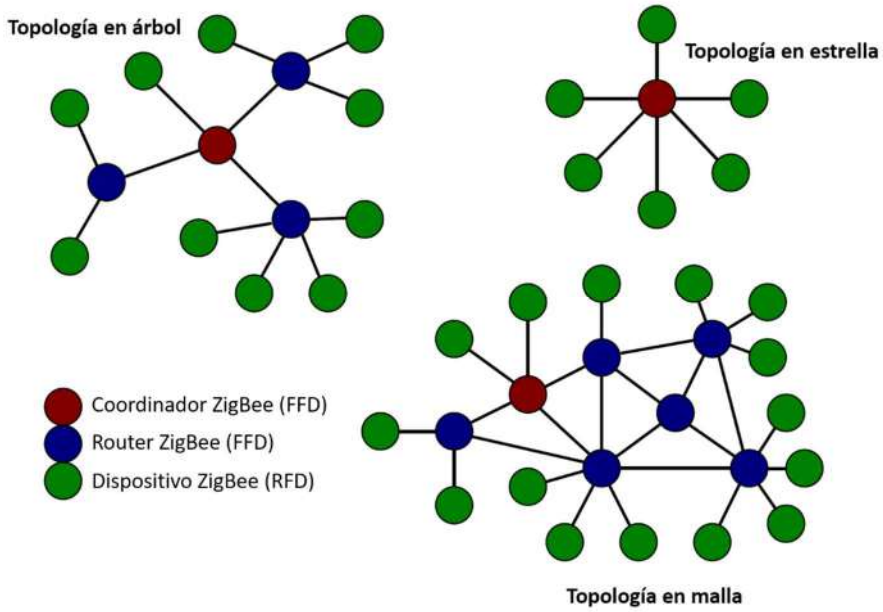


Figura 2.14: Topologías, nodos lógicos y dispositivos soportadas por la tecnología ZigBee

- *Router*: son nodos intermedios que permiten el intercambio de datos desde los dispositivos finales al coordinador y extienden la red más allá del alcance de radio de cobertura del coordinador. Debido a que este nodo actúa como coordinador local para los dispositivos finales debe implementar la mayoría de las capacidades del coordinador.
- *Coordinador*: controla y gestiona la comunicación en la red.
- *Nodo final*: captura datos y envía al router o al coordinador.

El estándar 802.15.4 define 2 tipos de dispositivos con el objetivo de minimizar el costo del sistema:

- FFD (*Fully Functional Device*): son dispositivos capaces de funcionar en cualquier topología, pueden soportar dos modos de operación como: coordinador PAN o enrutador. Al ser coordinador PAN puede comunicarse con cualquier otro dispositivo en la red y proporcionar servicios de sincronización a otros dispositivos y coordinadores.

- RFD (*Reduced Functional Device*): proporciona función limitada (dispositivos de baja complejidad con bajo requerimiento de procesamiento y memoria), estos dispositivos están previstos para aplicaciones simples donde se requiera el envío de pequeñas cantidades de datos. Además, solo pueden comunicarse con un dispositivo FFD a la vez.

En la actualidad, la Alianza ZigBee ofrece tres especificaciones que sirven como el sistema de red de base para facilitar sus estándares de mercado interoperables (ZigBee Alliance, 2018).

- *ZigBee PRO*: está diseñado para proporcionar la base para el IoT debido a que está optimizado para soportar grandes redes con miles de dispositivos y para un bajo consumo de energía.

- *ZigBee RF4CE*: se diseñó para aplicaciones bidireccionales de control de D2D (p. ej. apertura de puertas de garaje, sistemas de entrada, etc.) que no requieren las capacidades de red en malla con todas las funciones que ofrece la especificación ZigBee.

- *ZigBee IP*: es el primer estándar abierto para una solución de red, que ofrece una arquitectura escalable, que permite la integración de los protocolos IPv6 en la red basada en IEEE 802.15.4 para proporcionar una conexión sin fisuras de los dispositivos ZigBee a Internet. También es compatible con la estandarización de 6LoWPAN, así como todos los estándares de protocolo establecidos por los grupos IETF. Sin embargo, aún no se alcanzado su adopción generalizada.

- **LoRaWAN***:* Es un protocolo de comunicación de largo alcance recientemente desarrollado y diseñado por la Alianza LoRa. Está dirigido a aplicaciones de red de área amplia y está diseñado para ser un protocolo de baja potencia para habilitar el IoT. Su objetivo principal es garantizar la interoperabilidad entre varios operadores en un estándar global abierto.

  LoRa puede referirse comúnmente a dos capas distintas:

  (i) *LoRa:* que representa a la capa física, es una modulación de tipo de espectro ensanchado (CSS, por sus siglas en inglés, *Chirp Spread*

Spectrum), que permite múltiples velocidades de datos (0,3 kbps a 50 kbps), ya que tanto el ancho de banda como los factores de propagación (SF, por sus siglas en inglés, *Spreading factor*) son configurables según los requisitos de las aplicaciones de los dispositivos finales(Nolan *et al.*, 2016). Esto ayuda a optimizar el consumo de batería de los dispositivos (alrededor de 10 años de vida útil de la batería de un dispositivo). Además, LoRa soporta un amplio alcance de cobertura; se puede utilizar dentro de un entorno urbano, suburbano o rural (rango de 2 a 5 km en un área urbana, 15km en zonas suburbanas y 45km en zonas rurales). Funciona en las bandas ISM de 433, 868 o 915 MHz, según la región en la que se implementa.

(ii) *LoRaWAN:* es un protocolo de capa MAC. LoRaWAN proporciona un mecanismo de control de acceso medio, que permite a muchos dispositivos finales comunicarse con una puerta de enlace mediante la modulación LoRa. Si bien la modulación LoRa es propietaria de Semtech (Semtech, 2015), LoRaWAN es un estándar abierto desarrollado por la Alianza LoRa.

Las redes LoRaWAN están organizadas en una topología de red en estrella, que incluye varios componentes que se describe a continuación y que se muestran en la Figura 2.15:

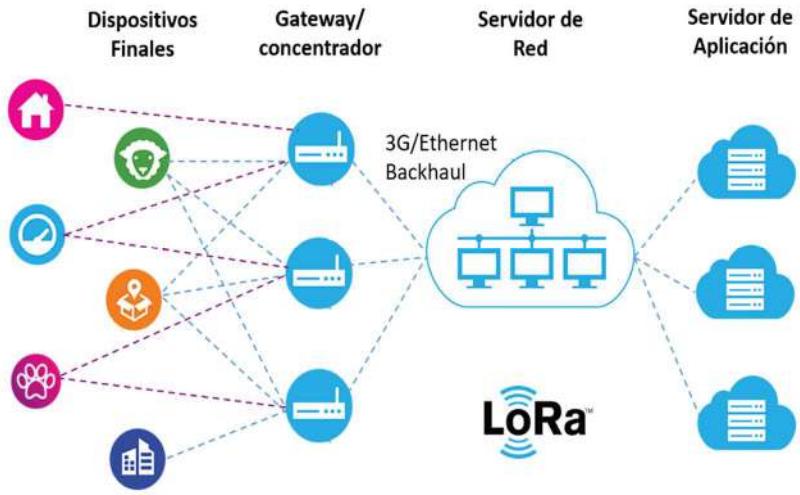


Figura 2.15: Componentes de una red LoRaWAN

- *Gateway o concentrador*: retransmiten mensajes entre los dispositivos finales y un servidor de red central a través de una interfaz de *backhaul* con un rendimiento más alto, generalmente utilizando conexiones Ethernet o 3G.

- *Dispositivos finales:* envían datos a las puertas de enlace a través de un solo salto inalámbrico en función de un enfoque programado o basado en eventos.

- *Servidor de la red*: es responsable de decodificar los paquetes enviados por los dispositivos realizando verificaciones de seguridad y adaptando la velocidad de datos, y generando los paquetes que deben enviarse de vuelta a los dispositivos o a las aplicaciones y servidores del usuario final.

- *Servidor de aplicación*: cada aplicación recibe los datos del servidor de red, decodifica los paquetes de seguridad y utiliza la información.

Una característica distintiva de la red LoRa es que contempla tres clases de dispositivos finales, *Clase A* (para todos*), Clase B* (para balizas*(beacons*) y *Clase C* (para escucha continua), cada una asociada con un modo de operación diferente (LoRa Alliance, 2017).

(i) *Clase A*: Los dispositivos finales de Clase A permiten comunicaciones bidireccionales. Las transmisiones siempre son iniciadas por los dispositivos finales de forma totalmente asíncrona. Después de cada transmisión de enlace ascendente, el dispositivo final abrirá (al menos) dos ventanas de recepción, con una pequeña variación utilizando el acceso *ALOHA puro,* en espera de cualquier comando o paquete de datos devuelto por el servidor de red. La segunda ventana se abre en una sub-banda diferente (previamente acordada con el servidor de red) para aumentar la resistencia frente a las fluctuaciones del canal. La clase A es la clase de dispositivos LoRaWAN con el menor consumo de energía. Las redes LoRa con dispositivos clase A están destinadas principalmente a aplicaciones de monitorización, donde los datos producidos por los dispositivos finales deben ser recopilados por una estación de control.

(ii) *Clase B*: Los dispositivos de Clase B están diseñados para aplicaciones con necesidades de tráfico de enlace descendente adicionales. Estos

dispositivos se sincronizan con el servidor de red por medio de paquetes de balizas (*beacons*), que son transmitidos por el *gateway*, por lo tanto, pueden recibir paquetes de datos o comandos de enlace descendente en ventanas de tiempo específicas, independientemente del tráfico de enlace ascendente. Esta clase está diseñada para dispositivos finales que necesitan recibir comandos de control (p.ej. interruptores o actuadores).

(iii) *Clase C*: se define para los dispositivos finales sin restricciones de energía estrictas (p.ej. dispositivos conectados a la red eléctrica), que pueden mantener la ventana de recepción siempre abierta, es decir siempre están escuchando el canal, excepto cuando están transmitiendo.

Las tres clases pueden coexistir en la misma red y los dispositivos pueden cambiar de una clase a otra. Sin embargo, no hay un mensaje específico definido por LoRaWAN para informar al *gateway* acerca de la clase de un dispositivo y esto depende de la aplicación.

Análisis comparativo de los protocolos de capa física y de enlace

Existe un creciente impulso para adoptar y diseñar tecnologías que se adhieran explícitamente a los requisitos de IoT. Esto incluye la modificación de tecnologías existentes, por ejemplo, de Bluetooth clásico a BLE, y de ZigBee clásico a ZigBee IP o el diseño de nuevas tecnologías como el IEEE 802.11ah. Estas tecnologías apuntan a abordar los requisitos clave de los dispositivos inalámbricos de IoT, como el bajo consumo de energía, las capacidades de computación más bajas, los costos operativos y de implementación reducidos, y un rango de cobertura más amplio. La sección 2.4.3.1.1.1 proporcionó una breve revisión de las tecnologías IEEE 802.15.4, BLE y la tecnología IEEE 802.11ah y LoRa. Sin embargo, todas las tecnologías anteriores tienen sus debilidades y obviamente sus fortalezas. Por ejemplo, en el espacio de tecnologías inalámbricas de baja potencia, BLE tiene la velocidad de datos más alta de 2.1 Mbps. Mientras que la tecnología ZigBee, soportada por el estándar IEEE802.15.4, tienen una velocidad de datos de 250 Kbps en la banda de frecuencia de 2.4GHz.

En cuanto al rango de cobertura, IEEE 802.11ah supera el rango de cobertura de ZigBee y BLE con un alcance de aproximadamente 1 km. No obstante, se debe tener en cuenta que el estándar 802.15.4 admite redes con topología en malla. En estas redes un mensaje se enruta a través de varios nodos hasta que llega a su destino. Por lo tanto, el rango de cobertura en redes ZigBee se puede

ampliar fácilmente con el uso de repetidores y/o una alta densidad de nodos en una formación de malla. Por otro lado, la tecnología LoRa promete tener una cobertura similar a la de las redes celulares. Esto constituye un gran salto en la cobertura de rango. Sin embargo, esto no sería posible sin involucrar a un proveedor de servicios LoRa, lo que puede aumentar el costo asociado derivado del uso de esta tecnología.

Con respecto a la capacidad del tamaño de la red, el protocolo BLE admite un máximo de ocho nodos por red que incluyen un dispositivo maestro y siete dispositivos esclavos. ZigBee puede tener hasta 65,000 nodos por red en una topología en estrella (Baker, 2005). Estas tecnologías también pueden extenderse a redes más sofisticadas. Por ejemplo, ZigBee puede extenderse a un árbol de clúster o red de malla; mientras que BLE se puede extender a una red de *scatternet*. Estas redes son topologías de red compuestas de *piconets* interconectadas; que son redes de topología en estrella simples compuestas por un maestro y sus esclavos (Darroudi & Gomez, 2017). Por otro lado, en comparación con las redes inalámbricas convencionales que operan en la frecuencia 2.4Ghz y 5Ghz. La tecnología IEEE 802.11ah puede admitir aproximadamente 8000 nodos, gracias a la cantidad de estaciones que se puede asociar en un punto de acceso (AP, por sus siglas en inglés, *Access Point*), que alcanza las 8191 estaciones (Seo *et al.*, 2014). También se espera que LoRa admita un gran volumen de dispositivos, sin embargo, los costos asociados, como se mencionó anteriormente, aún no se han explorado.

Por lo tanto, la elección de la tecnología con respecto a la velocidad y el rango de los datos tienen una estrecha relación con los requisitos de las aplicaciones de IoT en cuestión. En consecuencia, si una aplicación de IoT requiere el uso de un mayor número de nodos, ZigBee parece ser un candidato adecuado dada su ventaja de velocidad de datos sobre su contraparte de 802.11ah. Por otro lado, las aplicaciones de IoT que requieren el despliegue de menos nodos con tráfico mínimo, 802.11ah es un competidor fuerte para ZigBee. Esto se debe a que 802.11ah tiene un área de cobertura más amplia sin depender de ninguna técnica de malla. Además, está diseñado para ser compatible con versiones anteriores de las tecnologías de 802.11 Wi-Fi.

Las áreas de IoT involucran diversos conjuntos de dispositivos que utilizan diversas tecnologías de comunicación para compartir e intercambiar información. Dentro del IoT, algunas aplicaciones pueden ser simples aplicaciones *peer-to-peer*. Otras aplicaciones de IoT también pueden basarse en configuraciones de *Wireless Personal Area Networks* (WPAN), que involucran el uso de algunos dispositivos y

usuarios. Otras aplicaciones complejas pueden implicar la utilización de una variedad de dispositivos heterogéneos que se comunican utilizando una amplia gama de tecnologías, en diferentes configuraciones (p.ej. *Wireless local Area Network (WLAN) y* WWAN: *Wireless Wide Area Network)* y topologías. Por lo tanto, una tecnología que puede considerarse adecuada para una aplicación de IoT en particular puede no ser necesariamente adecuada para muchas otras. De hecho, la capacidad para conectar y coexistir varios dispositivos que funcionan con varias tecnologías de comunicación es la visión de IoT.

La Tabla 2.2 resume las principales características de cada una de las tecnologías de conectividad inalámbrica descritas anteriormente.

Tabla 2.2: Principales características de las tecnologías de comunicación inalámbrica

Características	Wi-Fi	Bluetooth	ZigBee	LoRa
Estándar	IEEE 802.11 ah	IEEE 802.15.1	IEEE 802.15.4	LoRaWAN R1.0
Frecuencia (MHz)	<1000	2400	868/915/2400	433/868/ 915
Rango (m)	<1000	10-100	10	Entornos -urbanos 2500 -suburbanos 15000 -rural 45000
Velocidad de datos (kbps)	150-7800	1000	250	0,3 a 50
Tamaño de la dirección (bits)	48	48	16/64	32
Tipo de red	WLAN	WPAN	WPAN	WWAN
Topología de red	Estrella	Estrella	Estrella, malla y árbol	Estrella
Pila de protocolos	Física/Enlace	Física a Aplicación	Red/Aplicación	Física/Enlace
Organización de estandarización	IEEE	Bluetooth SIG	ZigBee Alliance	LoRA Alliance

2.4.3.1.1.2 Capa de red y de transporte

Los protocolos de red y de transporte de cualquier aplicación IoT son extremadamente importantes para lograr la interoperabilidad entre diferentes soluciones. En las aplicaciones modernas de Internet, la conectividad IP de extremo a extremo está disponible en los puntos finales a través del protocolo TCP y UDP como protocolos de transporte prevalentes. Cualquier dispositivo habilitado para Internet puede procesar paquetes IP independientemente de los medios físicos de los cuales se transfieren. Posteriormente, los protocolos de aplicación pueden usar esta información para diferentes propósitos.

Para potenciar la conectividad extrema a extremo en redes LowPAN institucionalizadas para dispositivos con energía y almacenamiento restringidos, que se consideran incapaces de ejecutar IP, se propone una capa de adaptación de IPv6 sobre LowPAN (6LowPAN). Mientras que para el enrutamiento de los paquetes dentro de estas redes se propone el protocolo de enrutamiento RPL.

- **6LowPAN:** Es un estándar abierto definido por el IETF, (2011), que aprovecha las operaciones de baja energía de las capas MAC y física del estándar 802.15.4 y las capacidades de la red IP. En lugar de ser una tecnología de protocolos de aplicación IoT como Bluetooth o ZigBee, 6LowPAN es un protocolo de red que define una capa de adaptación eficiente para transportar paquetes IPv6 a través de enlaces IEEE 802.15.4 (Mulligan, 2007). En otras palabras, para transmitir paquetes IPV6 a través de redes LoWPAN.

  Los dispositivos 6LoWPAN pueden comunicarse con todos los demás dispositivos basados en IP en Internet. La elección de IPv6 se debe al gran espacio de direccionamiento disponible en IPv6. Las redes 6LoWPAN pueden conectarse a Internet a través de *border router*s con soporte 6LowPAN, que también tiene soporte de protocolo para la conversión entre IPv4 e IPv6, ya que la implementación de Internet en la actualidad es principalmente IPv4 (Zach; Shelby & Carsten;Bormann, 2009).

  El MTU mínimo que los host deben soportar en IPv6 no es lo suficientemente pequeño (1280 bytes) para caber dentro de la pequeña MTU (*Maximum Transmission Unit*) de 127 bytes del estándar 802.15.4. Por lo tanto, comprimir y fragmentar los paquetes para llevar solo la información esencial es una optimización que realiza la capa de adaptación

6LowPAN. La Figura 2.16 ilustra la infraestructura de una red 6LowPAN.

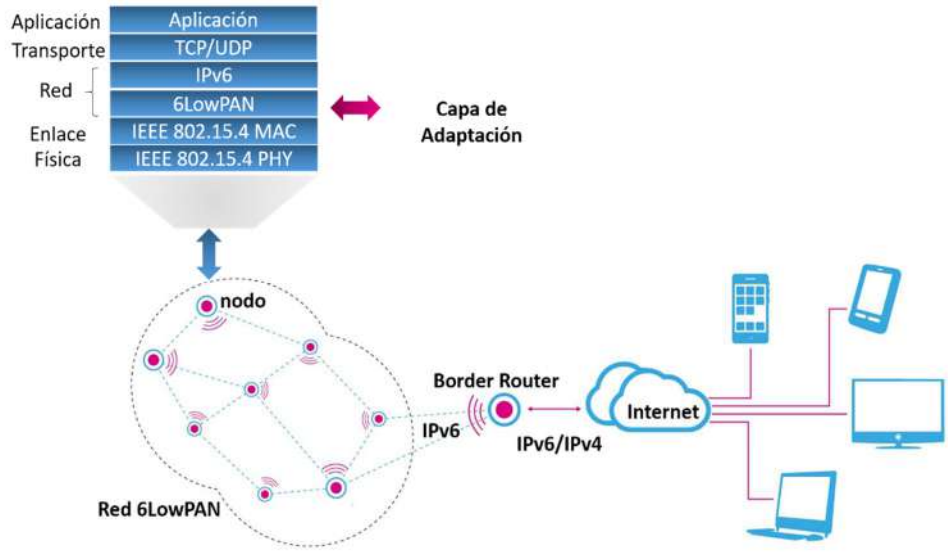


Figura 2.16: Red y pila de protocolos 6lowPAN

Específicamente, la capa de adaptación 6LowPAN realiza las siguientes tres optimizaciones para reducir la sobrecarga de comunicación (Hui & Culler, 2008; Culler & Chakrabarti, 2009):

(i) *La compresión de encabezado:* Algunos de los campos de encabezado de IPv6 se eliminan porque se pueden derivar de la información de nivel de enlace o en base a suposiciones de contexto compartido entre paquetes (p. ej. un prefijo de enrutamiento global común para toda la red LowPAN).

(ii) *Fragmentación*: Los paquetes IPv6 (1280 bytes) se fragmentan en múltiples *frames* de nivel de enlace para adaptarse al requisito mínimo de MTU del estándar 802.15.4 (127 bytes).

(iii) *Reenvío de capa de enlace*: Para el reenvío de los datagramas IPv6 en la capa de enlace, 6LowPAN admite el método *mesh under routing* (Chowdhury *et al.*, 2009), utilizando direcciones cortas a nivel de enlace en lugar de en la capa de red. Este es el método utilizado para comunicarse dentro de una red 6LowPAN.

- **RPL:** Fue estandarizado en 2012 por el IETF para el enrutamiento en redes de baja potencia y con pérdida (LLN, por sus siglas en inglés *low-power and lossy networks*). En las LLN la calidad del enlace fluctúa y se ve interrumpida por problemas como el ruido del piso, la variación de impedancia, inestabilidad, entre otros. Por lo tanto, la disponibilidad de alternativas de enrutamiento en cada nodo es esencial, debido a que en cualquier momento puede presentarse una degradación de la calidad del enlace o puede haber una falla de un nodo vecino. Dada la importante superposición entre LLN e IoT, y el hecho de que IPv6 es una característica esencial para IoT, RPL (del inglés, *routing protocol for low-power and lossy networks*) se ha convertido en el estándar de enrutamiento más práctico para LLN, ya que está diseñado para redes IoT de dispositivos IPv6(Banh *et al.*, 2015;Tsvetkov, 2011).
  RPL crea una topología de enrutamiento en forma de gráfico a cíclico orientado al destino (DODAG, por sus siglas en inglés, *destination oriented DAG*): un gráfico dirigido sin ciclos, orientado hacia un nodo raíz DODAG, por ejemplo, un enrutador de borde. De forma predeterminada, cada nodo mantiene varios padres hacia la raíz y eligen un padre preferido en la lista para alcanzarla. Las rutas RPL están optimizadas para el tráfico hacia o desde una o más raíces que actúan como sumideros para la topología.

  La topología se crea y mantiene mediante paquetes de control denominados objetos de información DODAG (DIO), anunciados por cada nodo. El paquete contiene información sobre las características de los gráficos a cíclicos dirigidos (DAG) y una función objetivo de enrutamiento (OF, por sus siglas en inglés, *objective function*) utilizada por cada nodo para seleccionar a los padres entre sus vecinos. La OF contiene un conjunto de métrica de enrutamiento (p ej., latencia y conteo de saltos). Los paquetes DIO son retransmitidos por cada nodo de acuerdo con una técnica adaptiva, el algoritmo *Trickle* (Levis *et al.*, 2004) que produce un compromiso entre la reactividad a los cambios de topología y la eficiencia energética. En una red con enlaces estables, *Trickle* garantiza que RPL envíe un menor número de mensajes de control, mientras que en un entorno en el que la topología cambia con frecuencia hará que RPL envía mensajes de control con mayor frecuencia. Para ser útil en una amplia gama de dominios de aplicaciones, RPL separa el procesamiento de paquetes y el reenvío del enrutamiento a fin de minimizar la energía, la latencia y satisfacer las restricciones de la red.

2.4.3.1.1.3 Capa de aplicación

Los dispositivos IoT requieren protocolos de aplicación livianos, seguros y eficientes en términos de ancho de banda y consumo de energía para la transmisión de los datos. En tal sentido, se han evaluado dos distintivos protocolos que cumplen con estos requerimientos en IoT, MQTT y CoAP. A continuación, se presenta una descripción detallada de estos protocolos.

- **CoAP** (Z. Shelby, K. Hartke, 2014)**:** Es un protocolo de capa de aplicación ligero diseñado por el grupo de trabajo IETF *Constrained RESTful Environments - CoRE p*ara su uso entornos restringidos (p.ej., en IoT, WSN y comunicaciones M2M). Es decir, en entornos formados por dispositivos y redes de baja potencia, y a menudo con altas tasas de error de paquetes (con perdida), como por ejemplo las redes *IPv6 over Low power Wireless Personal Area Networks (6LowPAN).*

  En cuanto a funcionamiento CoAP es similar a HTTP basado en una arquitectura REST (*Representational State Transfer*). Sin embargo, CoAP se usa en combinación con 6LowPAN sobre UDP, mientras que HTTP opera sobre TCP a menudo con el protocolo IP a nivel de capa de red (Ver Figura 2.17). CoAP apunta a reemplazar a HTTP en nodos restringidos debido a que proporciona una transferencia de datos mucho mayor y con menor sobrecarga en la red. Esto se debe al tamaño significativamente menor de las cabeceras, dado que todos los métodos y códigos de estado se codifica de forma binaria, lo cual reduce la sobrecarga del protocolo. En consecuencia, consume menor energía por lo que es un protocolo adecuado para las aplicaciones de IoT.

  Un análisis comparativo de la cantidad de bytes al usar CoAP y HTTP, realizado por Colitti *et al.* (2011), revela que el número de bytes por transacción es 1.288 y 128 al usar HTTP y CoAP, respectivamente. Es decir, una transacción CoAP tiene una cantidad de bytes significativamente más pequeña que la transacción HTTP, como consecuencia de la compresión del encabezado. No obstante, CoAP está diseñado para interactuar con HTTP de forma sencilla a través de *cross proxies* (Castellani *et al.*, 2011)*,* que pueden traducir de manera directa las solicitudes/respuestas entre los dos protocolos, a fin de que éstos puedan trabajar juntos en entornos de Internet tradicionales y restringidos permitiendo así una interoperación transparente entre ellos, tal como se muestra en la Figura 2.17.

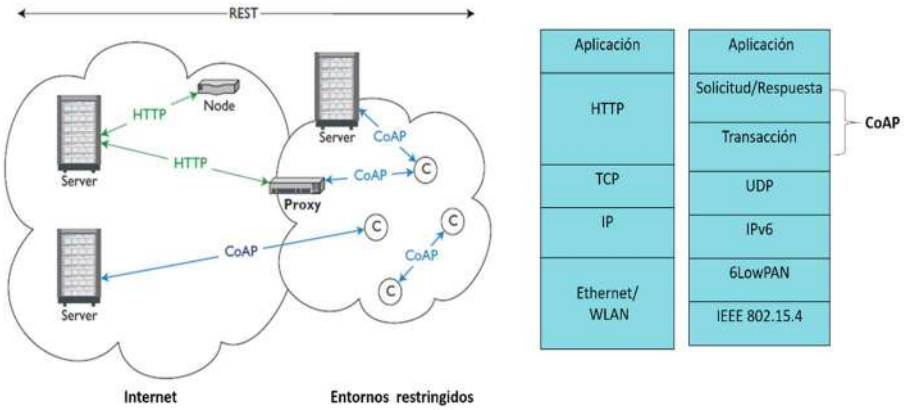


Figura 2.17: Funcionalidad de CoAP y HTTP (Adaptado de (Bormann *et al.,* 2012))

Como se observa en la Figura 2.17, CoAP está organizado en dos subcapas:

(i) *La subcapa de Transacción*. Maneja el intercambio de mensajes asíncrono entre los puntos finales a través del protocolo UDP, detectando duplicaciones y proporcionando una comunicación confiable.

(ii) La subcapa de Solicitud/Respuesta. Es responsable de la transmisión de solicitudes y respuestas para el manejo y transmisión de recursos. En esta capa se produce la comunicación basada en REST, en la que los recursos son abstracciones controladas por el servidor, los cuales se identifican mediante un URI (*Uniform Resource Identifier*) y pueden ser gestionados utilizando cuatro métodos: *GET, POST, PUT y DELETE*, los cuales se definen como sigue:

- **GET**: recupera una representación de la información que corresponde actualmente al recurso identificado por el URI de solicitud.

- **POST:** solicita que se procese la representación incluida en la solicitud. Por lo general, se crea un nuevo recurso o se actualiza el recurso de destino.

- **PUT:** solicita que el recurso identificado por el URI de la solicitud se actualice o cree en la representación adjunta. El

formato de representación se especifica mediante el tipo de medio.

- **DELETE**: solicita que se elimine el recurso identificado por el URI de la solicitud.

CoAP define cuatro tipos de mensajes: (Colitti *et al*., 2011; Al-Fuqaha *et al*., 2015):

- *Confirmable (CON):* requiere confirmación.
- *No confirmable (NON)*: no necesita confirmación.
- *Asentimiento o reconocimiento (ACK)*: reconoce que llegó un mensaje confirmable específico.
- *Reset (RST)*: indica que se ha recibido un mensaje (confirmable o No confirmable) pero falta el contexto para ser procesado correctamente.

La confiabilidad de CoAP se logra mediante una combinación de mensajes confirmables (CON) y no confirmables (NON). Este hecho es debido a que UDP no incorpora mecanismos de retransmisión, ni de recuperación de errores. Por lo tanto, si se realiza una comunicación que requiere confirmación, el servidor (destinatario) envía un mensaje de asentimiento (ACK) con el ID del mensaje. Este es el caso más básico, la respuesta se transmite directamente en el mensaje de asentimiento que confirma la solicitud, independientemente de si la respuesta indica éxito o fracaso. Es decir, no se requiere un mensaje separado para devolver la respuesta. Este modo de respuesta se denomina *piggybacked.*

En cambio, si el servidor no puede responder inmediatamente a una solicitud transmitida en un mensaje confirmable, simplemente responde con un mensaje de asentimiento vacío para que el cliente pueda detener la retransmisión de la solicitud. Cuando la respuesta está lista, el servidor lo envía en un nuevo mensaje confirmable (que a su vez debe ser reconocido por el cliente). Este modo de respuesta se denomina *separate.*

Si el servidor no es capaz de procesar el mensaje confirmable (es decir, ni siquiera proporcionar una respuesta de error adecuada), responderá

con un mensaje *Reset* (RST). En este caso, el cliente retransmite el mensaje confirmable (CON) utilizando un tiempo de espera predeterminado y un retroceso exponencial (*back-off*) entre retransmisiones, hasta que el servidor envíe el mensaje de asentimiento (ACK).

Por otra parte, si se realiza una comunicación que no requiere confirmación, el servidor enviará la respuesta en un mensaje no confirmable (NON). Sin embargo no enviará ningún mensaje de asentimiento (ACK). Cuando un servidor no puede procesar un mensaje no confirmable (NON), puede responder con un mensaje *Reset* (RST).

En general, el lado del servidor responde con un mensaje *Reset* (RST) cuando se pierde mensajes o se producen problemas de comunicación. En la Figura 2.18 se proporcionan ejemplos de flujo de solicitudes GET donde se muestra los tipos de mensaje CoAP y los modos de respuesta.

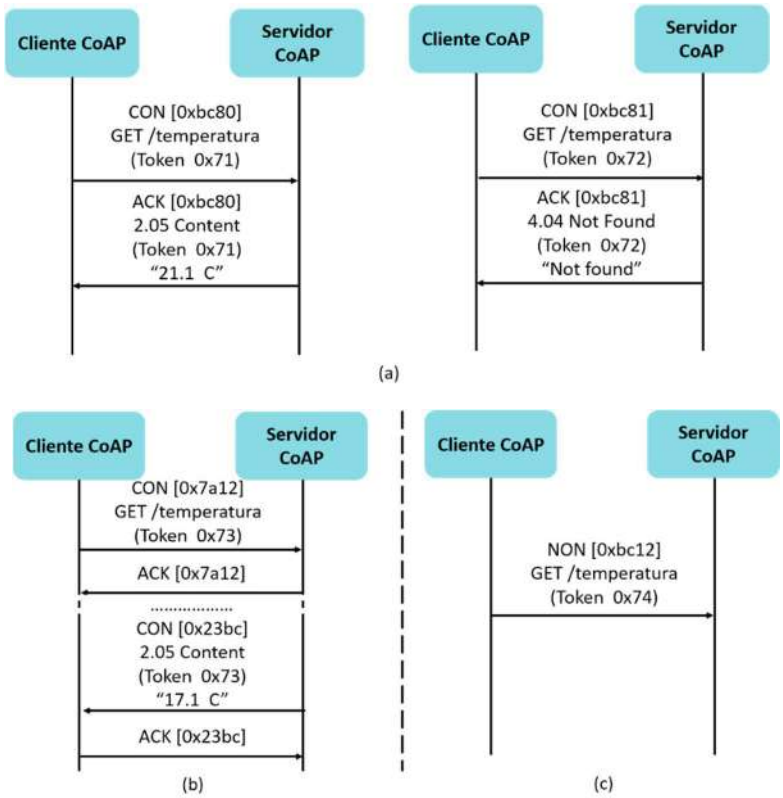


Figura 2.18: Tipos de Mensajes (a) Confirmable con respuesta *piggybacked; (b)* Confirmable con respuesta *separate (c)* No-confirmable.

Las ventajas de CoAP y su potencial para ser utilizado en IoT se derivan de sus principales características comparativas con el protocolo HTTP (Bormann *et al.*, 2012):

- *Observación de recursos*: Un cliente al enviar una solicitud GET puede indicar su interés en nuevas actualizaciones de un recurso especificando la opción *Observe*. Así el cliente se convierte en un observador de este recurso y cada vez que exista una actualización sobre el mismo, el servidor enviará un mensaje de notificación. Esta característica es habilitada mediante un mecanismo de publicación/subscripción.
- *Descubrimiento de recursos:* El servidor proporciona a los clientes una descripción de recursos disponibles a través de rutas URI (/.well-known/) bien conocidas (Nottingham & Hammer-Lahav, 2010). Así el cliente puede acceder a esta descripción con una solicitud GET en ese URI.
- *Transporte de recursos en forma de bloques:* El servidor y cliente pueden intercambiar datos sin la necesidad de actualizar los datos completos, reduciendo así la sobrecarga de comunicación.
- *Seguridad:* CoAP es un protocolo seguro ya que está construido sobre la seguridad de DTLS (Data Transport Layer Security) para garantizar la confidencialidad e integridad del contenido de los mensajes intercambiados.

- ***MQTT (Message Queue Telemetry Transport)*** (International Business Machines Corporation (IBM), 2010): es un protocolo de transporte de mensajería ligero que fue impulsado en 1999 por IBM y estandarizado en 2013 por la *Organization for the Advancement of Structured Information Standards(OASIS)*. Actualmente, MQTT es uno de los protocolos ampliamente utilizados en IoT y en comunicaciones M2M debido a su capacidad de ancho de banda y eficiencia de energía. Gracias a esta capacidad, MQTT al igual que CoAP, también está diseñado para redes restringidas con ancho de banda limitado y baja velocidad de transmisión, y dispositivos con recursos limitados. No obstante, este protocolo similar a HTTP solo puede ser ejecutado sobre los protocolos TCP e IP.

En contraste con el paradigma de Solicitud /Respuesta de HTTP, MQTT usa una arquitectura *publish/subscribe* para el intercambio de mensaje a través de un nodo central denominado *broker*. Su estrategia de comunicación se basa en la definición de *topics*, que son mensajes de tipos específicos enviados por los publicadores al *broker* y recibidos por los subscriptores. En general el protocolo MQTT clasifica los actores que participan en la red en clientes y servidores. Los clientes pueden tomar el rol de publicadores o subscriptores o ambos, mientras que los servidores representan a los *brokers*. No obstante, en una comunicación pueden estar presentes varios *brokers*. En este caso un *broker* puede actuar como cliente publicando los mensajes a otros *brokers (*que también pueden actuar como clientes*)*.

Los mensajes que se publican en la red se clasifican según diferentes tipos y los clientes solo recibirán aquellos tipos de mensajes a los que previamente se han subscrito. Para ilustrar lo explicado en la Figura 2.19 se muestra el funcionamiento básico del protocolo MQTT.

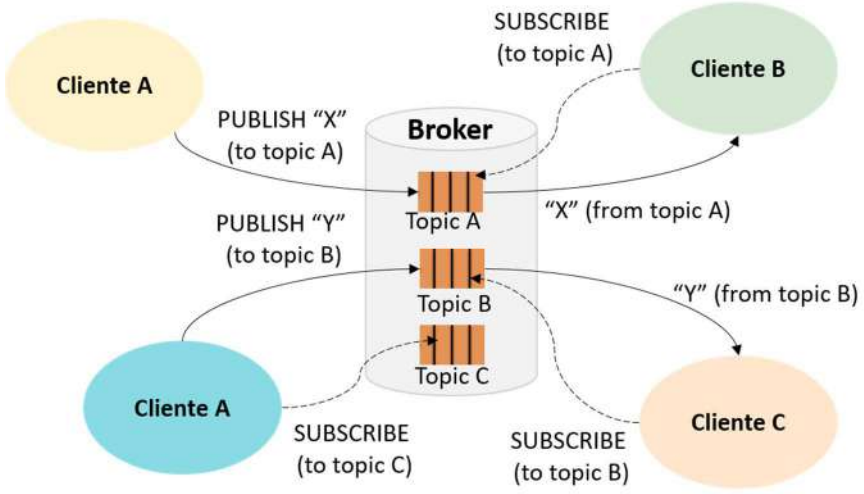


Figura 2.19: Funcionamiento del Protocolo MQTT.

Como se puede apreciar, un mensaje publicado (*Publish*) es enviado por un cliente a un servidor (*Broker*) para su distribución a los subscriptores. Cada mensaje publicado está asociado a un nombre de *topic* (en la figura

p. ej., A, B, C, D). El *topic* define una taxonomía de las fuentes de información para las cuales los suscriptores pueden registrar un interés. Un mensaje que se publica a un nombre de *topic* específico se entrega a los suscriptores conectados a ese *topic*. Si un cliente se suscribe a uno o más *topic*s, cualquier mensaje publicado sobre esos *topics* es enviado por el servidor al cliente como un mensaje publicado. Por lo tanto, MQTT puede admitir el intercambio de mensajes de *one-to-one* (uno a uno), *one-to-many* (uno a muchos), y *many-to-many* (muchos a muchos), en otras palabras, el protocolo MQTT admite diferentes tipos de enrutamiento (International Business Machines Corporation (IBM), 2010).

La principal ventaja que aporta este modelo de comunicación es el desacoplamiento que se produce entre los publicadores y subscriptores, gracias a la intermediación del *broker* de mensajes, que se resume en:

- El cliente que publica un mensaje no requiere conocer la existencia del cliente que consumirá ese mensaje.
- El cliente que publica el mensaje y el cliente que recibe el mensaje no necesitan estar conectados ni sincronizados a la vez. El *broker* es responsable de almacenar los mensajes y entregarlos cuando los clientes se conecten.

Además, un aspecto fundamental de MQTT es la definición de una estructura jerárquica de *topics*, la cual permite establecer relaciones padre-hijo. La estructura jerárquica permite subscribirse a un *topic* padre, y recibir la información de sus hijos. En general, cada *topic* puede ser separado en varios niveles utilizando el separador de nivel (/). Este principio de diseño permite el uso de los llamados comodines en las subscripciones:

- Multinivel (#): admite cualquier nivel por debajo del nivel mostrado en la cadena.
- Simple (+): admite solo hasta el nivel mostrado en la cadena.

Así, por ejemplo, si un cliente se subscribe a /UPV/#, recibirá mensajes sobre los temas:

/UPV/doctorados/

/UPV/maestrías/

/UPV/doctorados/telecomunicaciones

/UPV/doctorados/informática

/UPV/maestrías/telecomunicaciones

/UPV/maestrías/informática

Siguiendo con este ejemplo, si un cliente se suscribe a /UPV/+, recibirá mensajes sobre los temas:

/UPV/doctorados/

/UPV/maestrías/

Otra característica fundamental de MQTT es que garantiza la confiabilidad de las transmisiones. Para ello, define tres niveles de QoS, que reflejan el acuerdo con respecto a la transferencia de mensajes entre el *broker* y el cliente de publicación o subscriptor. Estos son:

- *QoS 0* (**Como máximo una vez**): Este es el nivel de QoS más rápido y requiere solo 1 mensaje. Un mensaje con QoS 0 se entrega como máximo una vez o no se entrega en absoluto. Este mensaje no se almacena en el remitente, tampoco se confirma o se reconoce su entrega a través de la red; si el cliente esta desconectado o el servidor falla, el mensaje podría perderse. Una vez que se ha enviado el mensaje, este se elimina de la cola de mensajes salientes. Por lo tanto, con este nivel no hay posibilidad de mensajes duplicados.

- *QoS 1*(**Al menos una vez**): Este es el nivel de QoS predeterminado de transferencia. Este nivel garantiza que un mensaje se entrega al menos una vez. Esto requiere que el mensaje se almacene en el lado del remitente hasta que se reciba un acuse de recibo (PUBACK). Si no se recibe dicho acuse, según los tiempos de espera, un remitente retransmitirá el mensaje con el indicador DUP (identificador de duplicado). Por lo tanto, los subscriptores pueden recibir el mismo mensaje varias veces. Una vez que se ha recibido el acuse de recibo, el mensaje se elimina de la cola de mensajes de salida.

- *QoS 2* (**Exactamente una vez**): Este es el nivel más alto de servicio en MQTT. El mismo que garantiza que un mensaje sea

recibido solo una vez por los destinatarios previstos, sin posibilidad de que este llegue duplicado. La garantía es proporcionada por al menos dos flujos de solicitud/respuesta, es decir se establece un acuerdo de cuatro vías entre el remitente y el receptor. El remitente y el receptor utilizan el identificador de paquete del mensaje publicado original para coordinar la entrega del mensaje. Cuando un receptor obtiene un mensaje responde al remitente con un paquete PUBREC que reconoce el paquete publicado. Si el remitente no obtiene este reconocimiento, envía el paquete nuevamente con el indicador DUP hasta que reciba el acuse de recibo. Una vez que el remitente recibe el acuse de recibo del receptor, el remitente descarta de forma segura el paquete inicial y responde con un paquete PUBREL. Aunque es el nivel de QoS más seguro, es más lento en procesamiento.

La Figura 2.20 muestra ejemplos de publicación de mensajes MQTT utilizando los diferentes tipos de QoS soportados.

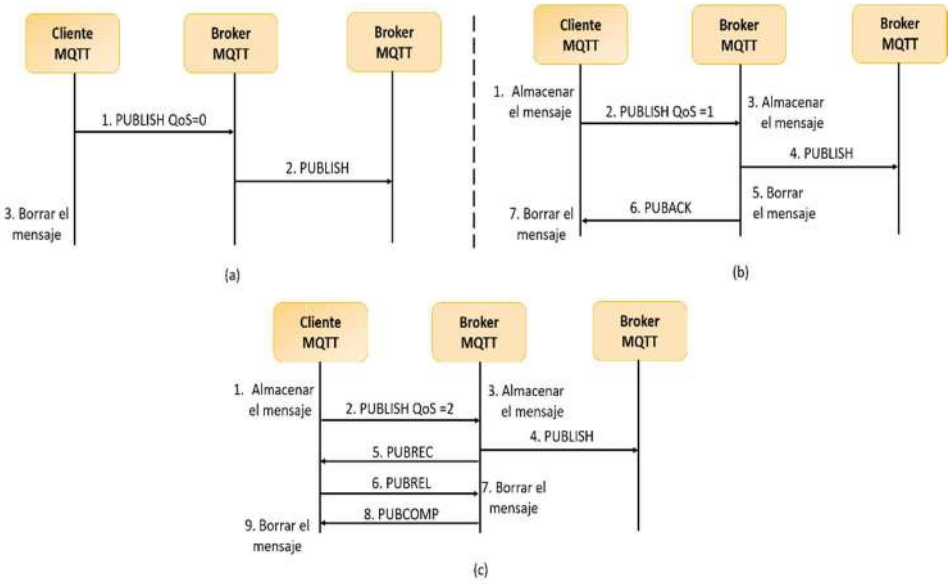


Figura 2.20: Ejemplos de mensajes MQTT utilizando diferentes tipos de QoS

En cada uno de los niveles de QoS se considera los dos lados de la entrega del mensaje del:

1. publicador al *broker*.
2. *broker* al subscriptor.

El cliente de publicación define el nivel de QoS cuando envía el mensaje al broker. Por su parte, los subscriptores definen también el nivel de QoS durante el proceso de subscripción. Si el cliente suscriptor define una QoS más baja que el cliente de publicación, el broker transmite el mensaje con la menor QoS, caso contrario, transmite con la QoS definida por el cliente suscriptor.

Para adaptar MQTT a redes restringidas que no son TCP/IP y que admiten velocidades de datos bajas y paquetes de longitud muy pequeños, como las WSNs ZigBee basadas en el estándar IEEE 802.15.4, se ha especificado MQTT-SN. Esta versión de MQTT define una asignación UDP de MQTT y permite la indexación de temas(Stanford-Clark & Truong, 2013).

Análisis comparativo de los protocolos de capa de aplicación

Desde un punto de vista cualitativo, los protocolos CoAP y MQTT difieren en varios aspectos que incluyen el patrón de comunicación utilizado, protocolos de transporte y de seguridad utilizados en la comunicación, confiabilidad y control de la congestión (Caro *et al.*, 2013).

- *Patrón de la comunicación:* Ambos protocolos, CoAP y MQTT, están inspirados en diferentes paradigmas de comunicación. MQTT utiliza el paradigma de pub/sub, mientras que CoAP al ser un protocolo RESTful utiliza un modelo petición/respuesta, el cual utiliza un subconjunto de verbos HTTP para manipular los recursos. No obstante, tiene soporte incorporado para utilizar el mecanismo pub/sub a través de la opción *Observe*. Esta doble naturaleza de CoAP lo convierte en una solución más flexible para los desarrolladores de aplicaciones. Sin embargo al utilizar la opción *observe*, la integración con aplicaciones web se torna más compleja, debido a que HTTP no es compatible con el modelo pub/sub.

- *Protocolos de transporte y seguridad de la comunicación:* La principal diferencia entre el protocolo CoAP y MQTT es que el primero se ejecuta sobre el

protocolo UDP, mientras que el último se ejecuta sobre TCP, por consiguiente, heredan de UDP y TCP sus ventajas y desventajas en términos de funcionalidad y rendimiento. La seguridad de la información no se aborda directamente en estos protocolos. Sin embargo, el cifrado a través de la red puede lograrse mediante el uso de SSL para MQTT o DTLS para CoAP. MQTT incluye una seguridad básica para el usuario que permite enviar el nombre de usuario y contraseña.

- *Confiabilidad y control de congestión:* Otra diferencia entre CoAP y MQTT es la disponibilidad de diferentes niveles de QoS. MQTT define 3 niveles de QoS (QoS-0, QoS-1, QoS-2) mientras que CoAP no proporciona QoS diferenciada. Como UDP no es inherentemente confiable, CoAP proporciona su propio mecanismo de confiabilidad. Esto se logra con el uso de mensajes confirmables (CON) y mensajes no confirmables (NON). Los mensajes CON requieren un acuse de recibo (ACK), mientras que los mensajes NON no necesitan un acuse de recibo. Los mensajes MQTT con QoS-0 y CoAP NON no garantizan la confiabilidad, mientras que los mensajes MQTT con QoS-1 y CoAP CON garantizan la confiabilidad basada en ACKs. El nivel de QoS 2 de MQTT, que garantiza que los mensajes duplicados no se entreguen al receptor, no corresponde a ningún nivel de QoS similar en CoAP. En tal sentido, MQTT es más flexible y adecuado para aplicaciones que no toleran mensajes duplicados. Con respecto al control de congestión, MQTT aprovecha los mecanismos incorporados de TCP. Además, ambos protocolos proporcionan un control de congestión básico en la capa de aplicación, al retransmitir los mensajes que no son reconocidos por un intervalo de retroceso (*back-off).*

- *Orientación de diseño y descubrimiento:* CoAP incluye soporte para etiquetar mensajes con tipos u metadatos (define el formato del contenido de datos). Mientras que MQTT está centrado en los datos, por lo que es agnóstico al formato del contenido. Los mensajes MQTT se pueden utilizar para cualquier propósito, pero todos los clientes deben conocer la estructura de los *topics* por adelantado para permitir la comunicación. CoAP, por el contrario, proporciona soporte incorporado para la negociación y el descubrimiento del contenido a través de la ruta de recursos bien conocida (/. well-known) [RFC 5785], lo que permite a los dispositivos sondearse entre sí para encontrar formas de intercambiar datos.

Las principales diferencias entre CoAP y MQTT se resumen en la Tabla 2.3

Tabla 2.3: Comparativa entre los protocolos de capa de aplicación CoAP y MQTT

Características	MQTT	CoAP
Patrón de comunicación	Pub/Sub	Petición/Respuesta Pub/Sub (Observe)
Protocolo de transporte	TCP	UDP
Mecanismos de seguridad	SSL	DTLS
Mecanismos de confiabilidad	QoS-1, QoS-2 y QoS-3	Mensajes confirmables (CON), mensajes no confirmables NON, Acuses de recibo (ACKs) y retransmisiones.
Orientación de diseño	Estructura de *topics*, que los clientes deben conocer previamente.	Estructura de recursos y descubrimiento a través de la URI /.well-known.

2.4.3.2 Interoperabilidad sintáctica

Como se ha mencionado anteriormente, el intercambio y uso de la información sin fisuras entre dispositivos IoT requieren múltiples niveles de interoperabilidad. Una vez que la interoperabilidad técnica proporciona una base para la conectividad con éxito de los dispositivos. El segundo desafío de interoperabilidad a resolver es la notación de datos. Los problemas de interoperabilidad sintáctica surgen cuando las reglas de codificación del remitente son incompatibles con las reglas de decodificación del receptor, lo que conlleva a que los mensajes no sean interpretados adecuadamente.

Hoy en día es cada vez más común que una aplicación obtenga sus datos de dispositivos IoT heterogéneos. Es bastante difícil admitir diferentes formatos de datos derivados de estos dispositivos. Por lo tanto, los datos deben normalizarse o convertirse a un formato común, que pueda ser leído por todos los elementos del sistema de IoT o incluso por diferentes sistemas de IoT que conduzcan a la interoperabilidad sintáctica. El remitente y el receptor deben interpretar esos datos de una manera compatible, lo que significa interoperabilidad a nivel sintáctico entre las estructuras de datos correspondientes en ambos extremos.

Los esquemas, formatos y estándares restringen la amplia gama de posibles tipos de información que potencialmente podrían incluirse en cualquier mensaje. Ser compatible con las restricciones conduce a mensajes robustos y estándar que facilitan la interoperabilidad.

La interoperabilidad sintáctica se logra mediante el uso de estándares. Afortunadamente, existen estándares maduros y sólidos para lograr la interoperabilidad sintáctica. En esta sección, describimos los estándares sintácticos existentes utilizados en Internet, su propósito general y uso.

- **XML – *Extensible Markup Language*** (W3C Information and Knowledge domain, 2016): es un lenguaje de marcado basado en texto que es estándar para el intercambio de datos en la web. XML fue diseñado para permitir que los documentos incluyan metadatos sobre el contenido, de modo que sea legible tanto por personas como por máquinas. Al igual que el protocolo HTML, los datos son identificados utilizando etiquetas de inicio y fin, que son identificadores encerrados entre paréntesis <…> <… />. El conjunto de estas etiquetas se conoce como marcadores, que incluye información sobre el contenido de un elemento y le da una estructura específica al documento. Para que múltiples aplicaciones usen los mismos datos XML, tienen que ponerse de acuerdo sobre los nombres de las etiquetas que pretenden usar.

  En la Figura 2.21 se presenta un ejemplo simple de algunos datos XML que se pueden usar para representar una medida de un sensor.

```
<? xml version = "1.1" encoding = "UTF-8"
standalone="no"?>
<measurement date = "15-10-2018 " >
        <value type="Decimal">20</value>
        <unit>ºC</unit>
        <time>20:15</time>
        <longitude>42.243057</longitude>
        <latitude>-0.58941</latitude>
        <location>indoor</location>
        <location>outdoor</location>
</measurement>
```

Figura 2.21: Ejemplo de representación de datos en XML

La primera línea es la declaración que identifica al documento como XML. La declaración también puede contener información adicional como:

- *Encoding*: identifica el conjunto de caracteres utilizado para codificar los datos. UTF-8 es el valor predeterminado *Unicode* comprimido. Este parámetro a pesar de ser opcional es útil ya que resuelve el problema de interoperabilidad de codificación.
- *Standalone:* indica si este documento hace referencia o no a una entidad externa o una especificación de tipo de datos externos.

Retomando el ejemplo, la segunda línea tiene un elemento llamado *measurement* que también tiene un atributo *date*, que tiene una fecha simple como valor. Esta es una de las características más útiles de XML, ya que el procesador XML puede buscar solo las medidas que tienen la fecha deseada. Es decir, XML proporciona identificación de datos mediante el uso de etiquetas. Dentro del elemento de *measurement*, hay varios elementos (*value type, unit, time, longitude y latitude*). Así, al recibir el elemento *measurement*, también se obtienen todos aquellos elementos que contiene. Este uso versátil para la estructura en el documento XML es una característica muy útil. Además, XML mantiene una jerarquía a través de las etiquetas anidadas, y proporciona texto sin formato lo que hace que sea un formato de datos legible para la representación de datos.

- **JSON (JavaScript Object Notation)**

JSON (Ecma Internacional, 2017) está ganando atención en el mercado de IoT, ya que es un estándar de serialización abierto, ligero, simple (basado en texto) de intercambio de datos. Además, JSON ofrece capacidades similares a las de XML para la anotación de datos; está diseñado para ser leído y creado fácilmente por humanos y máquinas. Este formato es completamente independiente del lenguaje de programación. No obstante, tiene muchas propiedades compartidas con varios lenguajes de programación (p.ej. C, C++, Java, Python, Perl, etc) debido a que posee dos de las construcciones básicas más comunes: un conjunto de pares (clave-valor) y una matriz (conjunto ordenado de valores o una lista).

Retomando el ejemplo de la medida de un sensor, la Figura 2.22 muestra la representación de dicha medida en formato JSON.

Este ejemplo tiene la misma estructura de anidamiento que la representación XML presentada anteriormente. En primer lugar, el ejemplo es un objeto JSON que tiene las claves *measurementDate* y *measurement*. *measurement* es un objeto JSON, que está anidando a su vez un matriz de 6 objetos.

```
{
"measurementDate": "15-10-2018",
"measurement":
      {
       "type": "Decimal",
       "value": "20",
       "unit": "ºC",
       "time": "20:15",
       "longitud": "42.243057",
       "latitude": "-0.58941",
       "location":
            [
           "indoor", "outdoor"
        ]
      }
}
```

Figura 2.22: Ejemplo de representación de datos en JSON

Al comparar la representación JSON con la representación XML, ambas ofrecen las mismas funcionalidades. El contenido se puede buscar por fecha y los mismos valores se almacenan en ambos formatos.

En JSON se necesita un objeto adicional de *measurement* para contener los datos. Pero en XML, la lista de propiedades de *location* tiene dos elementos con la misma etiqueta debido a la falta de funcionalidad de lista o matriz. Esto crea diferencias de rendimiento, en las que JSON es una alternativa más simple al esquema XML. No obstante, el esquema XML y el esquema JSON comparten muchos de sus objetivos y características, como se expresa en la Tabla 2.4

Tabla 2.4: Comparación de características de los formatos de serialización XML y JSON

Características	XML	JSON
Baseline data	Texto	Texto
Gramática	Compleja	Simple
Autodescripción	Sí	Sí
Esquema	Basado en etiquetas	Basado en tokens
Rendimiento	No optimizado para el rendimiento debido a la utilización de etiquetas	Más rápido que XML debido a su tamaño.
Orientación	Está orientado a documentos	Está orientado a los datos

Interoperabilidad	Es interoperable	Tiene el mismo potencial de interoperabilidad de XML
Adopción en IoT	XML es ampliamente adoptado por la industria en general, y especialmente adecuado para el intercambio de datos en la Web.	Su similitud con las estructuras de datos en lenguajes de programación (p. ej. C, C++, Java, Python, Perl, etc.) y su simplicidad hacen que JSON sea más fácil de adoptar en IoT.

Además, ambos formatos de representación de datos están basados en texto y soportan esquemas. XML conserva el aspecto de HTML, con marcas de texto basadas en etiquetas de inicio y fin, mientras que JSON delimita datos con *tokens* de sintaxis (Delgado, 2013), con una gramática simple que tiene cierta similitud con estructuras de datos de lenguajes similares a C. XML ha apuntado a la flexibilidad y generalidad, mientras que JSON ha enfatizado la simplicidad, que es el secreto de su popularidad.

2.4.3.3 Interoperabilidad semántica

En el nivel más alto, el problema de la interoperabilidad semántica radica en obtener un significado diferente del mismo contenido de un mensaje enviado y recibido entre dispositivos y/o usuarios humanos. Para ser más precisos, los datos generados por los dispositivos pueden tener un formato de datos definido (p. ej. JSON o XML), pero los modelos de datos y los esquemas utilizados por diferentes fuentes generalmente son diferentes y no siempre son compatibles.

Por ejemplo, en la Figura 2.23, el proveedor de la bombilla A usa los comandos "abrir (1)/cerrar (0)", para activar y desactivar respectivamente, mientras que el proveedor de la bombilla B usa los comandos "ON/OFF".

Abrir (1)/cerrar (0) ON/OFF

Proveedor "A" Proveedor "B"

Figura 2.23: Dispositivos no interoperable semánticamente

Obviamente, estas dos bombillas no pueden ser interoperables semánticamente. Este problema, de manera general se puede resolver al asignar un significado semántico a la información que se intercambia entre dispositivos; descripciones comunes y representación de datos, que caracterizarán a los dispositivos, sus capacidades y los datos que producen, que permita una interpretación correcta, comprensión inequívoca y sin ambigüedades del significado del contenido. Significado que puede ser interpretado y compartido por varias aplicaciones de dominio de IoT para obtener los mayores beneficios de las redes de IoT subyacentes implementadas.

Retomando el ejemplo concreto de las bombillas, la interoperabilidad semántica permitirá que los significados de activación y desactivación se puedan entender mediante una bombilla con los comandos respectivos abrir (1) y cerrar (0), y al mismo tiempo, mediante otra bombilla diferente, con un proveedor diferente y especificaciones diferentes con los respectivos comandos ON y OFF.

Para enfrentar el problema de interoperabilidad semántica en IoT, se utilizan ontologías. Una ontología es una especificación explícita de una conceptualización compartida (Gruber, 1993; van Heijst *et al.*, 1997) y está compuesta por un conjunto de objetos y relaciones entre ellos. Las ontologías promueven el intercambio de una comprensión unificada de información estructurada específica de dominio entre agentes de software lo que permite a los sistemas consumir datos basados en conceptos y relaciones predefinidos en la ontología. Por lo tanto, al establecer una notación semántica a los datos sensoriales de los dispositivos, las aplicaciones y los servicios de IoT podrán consumir estos datos, independientemente de sus fuentes o formatos. Además, será posible integrarlos con otras fuentes de datos semánticos en la web.

Para proporcionar interoperabilidad semántica en IoT, varias ontologías se han desarrollado para ser usadas para varios propósitos incluyendo la descripción de sensores, redes de sensores, recursos IoT y servicios, entre otros. El esfuerzo más notable en el campo de IoT para describir sensores y conceptos relacionados es la ontología SSN (del inglés *Semantic Sensor Network*) (Compton *et al.*, 2012).

- **SSN:** es una ontología independiente del dominio, desarrollada por el *W3C,* para la anotación semántica de dispositivos. La ontología puede describir sensores, su precisión, capacidades, observaciones y métodos utilizados para la detección. Además, incluye otras especificaciones de los sensores, como el rango de medición, restricciones de operación y la

sensibilidad. La ontología SSN, disponible en (W3C Semantic Sensor Network Incubator Group, 2011) está organizada conceptualmente, en diez módulos. Además, esta ontología consta de 41 conceptos y 39 propiedades de objeto. SSN se basa en un patrón de diseño de ontología central que describe las relaciones entre los sensores, el estímulo y las observaciones (SSO, por sus siglas en inglés, Stimulus-Sensor-Observation) como se puede apreciar en la Figura 2.24.

El patrón SSO vincula los sensores, lo que monitorizan (*stimulus*) y las observaciones resultantes. Este patrón se ha desarrollado como una base mínima y común para definir ontologías para la Web de sensores semánticos, así como para abordar explícitamente la necesidad de una semántica ligera para los datos vinculados (Compton *et al.*, 2012).

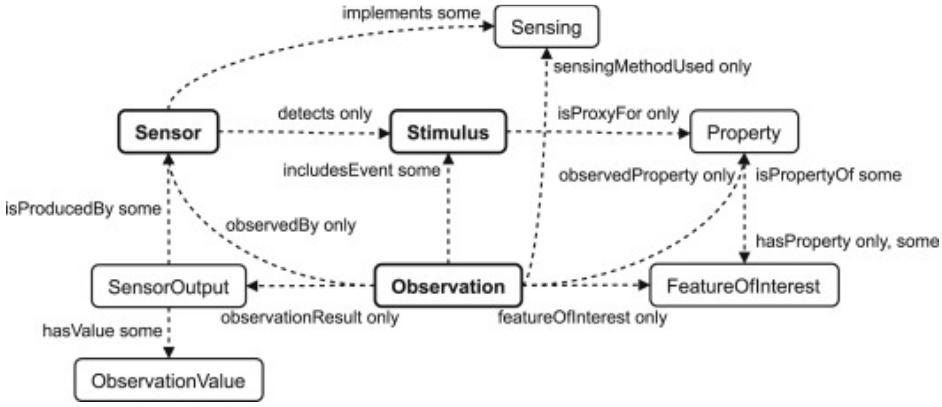


Figura 2.24: Una visión general del patrón SSO de la ontología SSN.

A continuación, se describe cada uno de componentes que abarca SSO.

- *Los estímulos*: son cambios o estados en un entorno que un sensor puede monitorizar/detectar y usar para medir una propiedad. Un estímulo, es, por lo tanto, un proxy para una propiedad observable, o un número de propiedades observables. Las propiedades en sí mismas son características observables de entidades del mundo real.

- *Los sensores:* son objetos físicos que observan, transforman los estímulos entrantes en otra representación, a menudo digital. Los sensores pueden ser dispositivos de hardware, sistemas de detección, cualquier dispositivo que detecte.
- *Las observaciones:* son el nexo del patrón SSO. Para un evento de detección, una observación puede vincular el acto de detección, el evento que es el estímulo, el sensor, un método, un resultado, una característica observada y propiedad, colocando todo en un contexto interpretativo. Es decir, las observaciones son contextos para interpretar los estímulos entrantes y fijar parámetros como el tiempo y la ubicación.

Una forma formal de describir una ontología y expresar la semántica es usar el lenguaje de ontologías Web (OWL, por sus siglas en inglés, *Web Ontology Language*). El lenguaje OWL propuesto por el W3C, está diseñado para ser usado en aplicaciones que necesitan procesar el contenido de la información en lugar de únicamente representar información para los humanos, es decir permite representar ontologías, que son comprensibles para el ser humano y legibles por máquinas.

Tres técnicas principales pueden ser utilizadas para la definición de ontologías(Ganzha *et al.*, 2018):

- *Alineación de ontologías*: Es un conjunto de correspondencia entre dos o más ontologías. Las correspondencias pueden ser simples (entre entidades atómicas) o complejas (entre grupos de entidades y subestructuras), pero siempre se relacionan entidades de diferentes ontologías. Las alineaciones contienen predicados sobre la similitud, denominados coincidencia (p. ej. axiomas de equivalencia), o un axioma lógico (p.ej. una asignación).

- *Fusión de ontologías:* Es un proceso de combinar dos o más ontologías en una ontología, cuyo resultado almacene el conocimiento de todas las ontologías fusionadas. Un ejemplo de fusión es una ontología que es el resultado de una suma de conjuntos de axiomas de ontologías combinadas.

- *Traducción ontológica:* Es un proceso de cambio de la semántica subyacente de un conocimiento. Dada cierta información descrita semánticamente, en términos de una ontología de origen, se transforma en información descrita en términos de una ontología de destino.

La información resultante contiene información interpretable (comprensible) en el alcance de la semántica objetivo.

Las aplicaciones que se desarrollan sin conocer los modelos de datos (descripciones) de los dispositivos, no se podrán implementar automáticamente. Por lo tanto, el problema de la interoperabilidad semántica radica en obtener un significado diferente del mismo contenido. Este problema se puede resolver al asignar un significado semántico a la información que se intercambia entre los dispositivos y las aplicaciones para facilitar la interpretación sin ambigüedades del significado.

2.5 Conclusiones

La proliferación de una nueva generación de dispositivos para IoT, equipados con inteligencia incorporada, capacidades de detección, comunicación y activación, impulsa una rápida realización de la visión de IoT con una mínima intervención humana. En particular, los sensores y actuadores son cada vez más potentes, menos costosos y más pequeños, lo que hace que su uso sea omnipresente. De acuerdo a este paradigma emergente, todo estará perfectamente conectado para formar un continuo de objetos interconectados y direccionales en una infraestructura de red global para proporcionar servicios avanzados a los usuarios finales.

Este capítulo ha introducido los principales conceptos relacionados con este trabajo de investigación. Se ha abordado IoT, sus principales características, componentes de construcción, así como las aplicaciones previstas con la finalidad de contextualizar a qué tecnología se referirá esta tesis, así como también sus particularidades y conceptos. Además, se han discutido los principales desafíos que deben abordarse para respaldar la visión de IoT, que cubre diferentes áreas de investigación: conectividad, arquitectura del sistema, interoperabilidad e integración, complejidad computacional y de almacenamiento, seguridad, confianza y privacidad. Como resultado, muchos problemas pendientes de investigación aún esperan soluciones adecuadas.

Desarrollar la capacidad de comunicarse entre los dispositivos, a fin de intercambiar y utilizar la información proporcionada por ellos, desde otras entidades (dispositivos, aplicaciones, servicios, etc.) es el principal desafío para la realización de IoT. Desde la aparición de IoT, la importancia del desafío de la interoperabilidad en IoT ha sido enfatizada de manera general tanto por el mundo académico como por la industria. Este capítulo, presenta una visión general completa de la

interoperabilidad de dispositivos en IoT. Al hacer esto se identificaron los niveles de interoperabilidad que necesitan ser direccionados en este campo: técnica, sintáctica y semántica. Además se definieron los patrones de comunicación, protocolos y las tecnologías de comunicación de capa física, enlace y de aplicación que puede ser utilizadas para manejar los niveles de interoperabilidad técnica y sintáctica, cuya utilización representa el aporte de este trabajo.

Capítulo 3
Especificación de la arquitectura

« La teoría es espléndida, ponerla en práctica tiene más valor»

James Cash Penny (1875-1971)

3.1 Introducción

El objetivo principal de este capítulo, es la descripción general de la arquitectura propuesta para la interoperabilidad de dispositivos físicos en IoT, la misma que va a ser implementada y validada mediante dos casos de uso aplicados a los dominios: AAL-AHA (*INTER-Health*) y Transporte y Logística (*INTER-LogP*).

La arquitectura guarda relación con los tipos de interoperabilidad previamente analizados en el estado del arte, sin perder la aplicabilidad a los diferentes casos de uso. La arquitectura tendrá como finalidad garantizar la interoperabilidad técnica y sintáctica de los dispositivos IoT heterogéneos, y su integración con plataformas IoT estándar, tratando de seguir siempre que sea posibles modelos y diseños estándar.

Como se ha indicado en estado del arte, IoT puede ser aplicado a múltiples dominios cada uno de los cuales conlleva varios requisitos, por lo que ninguna arquitectura individual, podría adaptarse a todos estos dominios. Sin embargo, la definición de una arquitectura genérica que admite la adición de capacidades, así como la compatibilidad con muchos requisitos aplicados a una amplia variedad de dominios de aplicación y que permita llevar a cabo la secuencia de acciones que se realizan a nivel de las capas de la arquitectura básica de una solución IoT (Tan, L., & Wang, 2010) (R. Khan *et al.*, 2012) es inherentemente útil y valiosa.

La arquitectura que proponemos es inherentemente neutral del proveedor, de las tecnologías o protocolos de comunicación de los dispositivos. Además, no es específica de un conjunto de tecnologías, aunque está altamente influenciada por

el uso de protocolos IoT livianos y seguros. Proporciona un punto de partida para los arquitectos que buscan crear soluciones de IoT donde la interoperabilidad es el primer desafío a resolver, así como una base sólida para un mayor desarrollo. La arquitectura propuesta se utiliza como motivación para el resto de la tesis.

Este capítulo comienza analizando los requisitos de la arquitectura propuesta. A continuación, se ofrece una descripción general de los principios de diseño que motivan la creación de la arquitectura, así como los estilos fundamentales de diseño. Posteriormente, se presenta una descripción general de la arquitectura que aborda cada dominio de la arquitectura para explicar sus respectivos componentes funcionales, así como también sus principales características.

Este capítulo está basado en las publicaciones (D. C. Yacchirema & Palau, 2016; D. C. Yacchirema, Sarabia-Jácome, *et al.*, 2018; Diana Yacchirema, Sarabia-Jácome, *et al.*, 2018)

3.2 Análisis de requerimientos

La naturaleza multifacética de IoT plantea importantes requisitos en el diseño de la arquitectura. En primer lugar, el dominio de IoT por sí mismo define sus propios requisitos para el diseño de la arquitectura y su comportamiento como los definidos por la ITU (2014). En segundo lugar, los dispositivos IoT tienen requisitos que surgen de sus capacidades limitadas de comunicación, memoria y potencia. A su vez estás restricciones afectan a las propiedades de la red, como se ha analizado en el Capítulo 2. Finalmente, las características de implementación de varias aplicaciones de IoT influyen en el diseño general de la arquitectura.

Para un diseño claro de la arquitectura, en función de los requisitos de la naturaleza multifacética de IoT; esta sección se enfoca en definir lo que la arquitectura debe cumplir para alcanzar los objetivos planteados. Se realiza un listado de requerimientos a cubrir tanto para la interoperabilidad de los dispositivos heterogéneos, como para la integración de los dispositivos con plataformas IoT estándar y abiertas, así como también para la construcción de aplicaciones IoT. La Tabla 3.1 resume los principales requerimientos a cubrir por la arquitectura propuesta.

Tabla 3.1: Requerimientos de la arquitectura

Requerimiento	Descripción
R1. Soporte multi-protocolo a nivel de capa física, enlace y de red	La arquitectura debe poder admitir los protocolos de comunicación más comunes para IoT. El mercado de Internet de las cosas se ha diversificado para tener diferentes tecnologías adoptadas que utilizan diferentes protocolos de comunicación, tanto a nivel de capa física, enlace como de red. Es decir, las comunicaciones entre dispositivos pueden tener lugar a través de varios tipos de tecnologías cableadas o inalámbricas, como Ethernet, ZigBee, Bluetooth, Wi-Fi, 6LowPAN, LoRA en la capa física y de enlace, y a través de varios protocolos como IPv4 e IPv6 en la capa de red. Debido al hecho de que se pueden usar diferentes tipos de dispositivos, no debe haber ninguna limitación en los protocolos de comunicación compatibles.
R2. Conversión de protocolos	La arquitectura debe ser compatible con la interconexión de redes y la comunicación sin fisuras entre dispositivos IoT, para ello se requiere la conversión de protocolos que trabajan en diferentes capas. Una situación es cuando las comunicaciones en la capa de enlace utilizan diferentes protocolos, por ejemplo, ZigBee y 6LowPAN; La otra situación es cuando las comunicaciones entre la capa enlace y de red implican diferentes protocolos, por ejemplo, un protocolo de tecnología 6LowPAN en la capa de enlace y el protocolo de tecnología 3G en la capa de red. Además, los dispositivos también admiten protocolos de capa de aplicación como HTTP, CoAP y MQTT, en esta situación también es necesario la conversión de protocolos.
R3. Soporte de protocolos IoT optimizados para las comunicaciones M2M	Como se ha mencionado en el Capítulo 2, la mayoría de los dispositivos y redes que conforman IoT presentan ciertas restricciones, es decir son dispositivos con recursos limitados en términos de memoria, CPU, almacenamiento, etc. La arquitectura debe soportar protocolos IoT de capa de aplicación livianos y abiertos para establecer las comunicaciones entre estos dispositivos restringidos.

R4. Intercambio de información bidireccional	La interacción entre los dispositivos IoT puede seguir el concepto de comunicación M2M para recopilar y difundir los datos monitorizados por parte de los sensores, y controlar los dispositivos actuadores. La arquitectura debe admitir la gestión de la conectividad y de la comunicación bidireccional entre los dispositivos IoT.
R5. Almacenamiento de datos	Para derivar valor de los datos que provienen desde los dispositivos de IoT, la arquitectura debe facilitar la gestión de la información de los dispositivos IoT que envían constantemente datos que requieren ser almacenados de acuerdo a políticas predefinidas.
R6. Procesamiento y análisis de datos en tiempo real	Las acciones deberían ser "casi en tiempo real", por lo que se requiere una capacidad de procesamiento y análisis de la información en tiempo real para identificar conocimientos adicionales.
R7. Comunicaciones periódicas y/o basadas en eventos	La arquitectura debe actuar en referencia a la información a través de comunicaciones periódicas y/o basadas en eventos.
R8. Notación sintáctica	Los diferentes tipos de dispositivos IoT proporcionan información utilizando un formato de datos específico. La arquitectura debe permitir la definición de un formato de datos común, con el fin de lograr una interoperabilidad entre los diferentes silos de datos o un intercambio transparente de información.
R9. Seguridad	En respuesta a la seguridad en la entrega de estos datos, la arquitectura debe proporcionar mecanismos de autenticación.
R10. Calidad de servicio	La arquitectura debe poder proporcionar servicios de QoS. Las aplicaciones que requieren la entrega de datos de los dispositivos y/o eventos generados en tiempo real deben recibir alta prioridad para mejorar su rendimiento.

R11. Integración con plataformas IoT estándar	La arquitectura debe facilitar la integración y coexistencia transparente de los dispositivos IoT a las diferentes plataformas adoptadas/integradas. La arquitectura requiere conectores a las diferentes plataformas de IoT para acceder a sus servicios con el objetivo de facilitar los *proxies* de software que permitirán la interoperabilidad de los dispositivos con estas plataformas.

3.3 Visión general de la arquitectura

La arquitectura está alineada con tendencias de investigación recientes, como, el modelo de referencia arquitectónico IoT-ARM (por sus siglas en inglés *IoT Architectural Reference Model*) desarrollado por el proyecto europeo *Internet of Things –Architecture* (IoT-A)(Bauer, Boussard, *et al.*, 2013) y la arquitectura funcional M2M derivada del ETSI (2011) basada en dominios, como se identificó anteriormente. Además, la arquitectura se base en el patrón de comunicación D2G analizado en la sección 2.4.2.3.

IoT-ARM (Bauer, Boussard, *et al.*, 2013) es un modelo de referencia arquitectónico que apunta a conectar sistemas y arquitecturas creados verticalmente. IoT-ARM pretende modelar todo el dominio de IoT, las entidades que contribuyen a él y sus relaciones. IoT-ARM consta de un modelo de referencia (MR-IoT) una arquitectura de referencia (AR-IoT) y un conjunto de directrices.

Un modelo de referencia es un marco abstracto para comprender relaciones significativas entre las entidades de algún entorno (Bauer, Boussard, *et al.*, 2013). Y consiste en un conjunto mínimo de conceptos, axiomas y relaciones unificadoras dentro de un dominio de problema particular, y es independiente de estándares, tecnologías, implementaciones u otros detalles específicos.

Por su parte una arquitectura de referencia es un modelo de referencia mapeado a elementos de software (que implementa cooperativamente la funcionalidad definida en el modelo de referencia) y relaciones entre ellos. El objetivo principal de una arquitectura de referencia es proporcionar orientación para el desarrollo de arquitecturas concretas.

MR-IoT proporciona los conceptos y definiciones sobre los cuales se pueden construir las arquitecturas de IoT, es decir establece una base común y un lenguaje común para la descripción de alto nivel del dominio de IoT para el cual se

construye la arquitectura. Por su parte AR-IoT proporciona diferentes vistas de la arquitectura de acuerdo con las partes interesadas del sistema. Las vistas arquitectónicas clave de AR-IoT incluyen la vista de contexto y vista funcional.

La base del MR-IoT es el modelo de Dominio de IoT, que describe los conceptos principales de IoT (como los dispositivos, servicios de IoT, entidades virtuales) y sus relaciones. Este modelo es consistente con el estándar ISO / IEC / IEEE 42010 y también ha sido utilizado para la definición de arquitecturas de alto nivel como HAL (del inglés *High level architecture*) propuesta por el grupo de trabajo de estandarización IoT, WG03 de AIOTI (Alliance for Internet of Things Innovation, 2018) centrada en la interoperabilidad semántica. Todos los demás modelos, como el modelo de información de IoT, el modelo funcional, el modelo de comunicación, el modelo de seguridad y privacidad, junto con la AR-IoT se basan en los conceptos introducidos en el modelo de dominio.

La arquitectura propuesta en esta tesis está en línea con el enfoque del MR-IoT, ya que los dominios de la arquitectura siguen las directrices del modelo de dominio, de comunicación y funcional. Por otro lado, debido a su respaldo en la industria, la arquitectura funcional propuesta por el ETSI (descrita en la sección 2.2.5.3) se considera como la base para la definición de los dominios funcionales de la arquitectura, con énfasis en la definición de nodos intermedios (*gateways*) que ocultan las especificaciones de la red y promueve la interoperabilidad de dispositivos y su integración con plataformas IoT integradas. Además, para describir la arquitectura con más detalle, de manera similar a la identificación de partes interesadas y actores en las prácticas estándar de ingeniería de software, recurrimos al uso de las vistas arquitectónicas de contexto y funcional de AR-IoT. Rozanski & Woods. (2012) define una vista como "*una representación de uno o más aspectos estructurales de una arquitectura que ilustra cómo la arquitectura aborda una o más inquietudes de uno o más de sus interesados*". La creación genérica de una vista se realiza a través del concepto de punto de vista, definido de acuerdo con el estándar IEEE 1471(2000) como "*una colección de patrones, plantillas y convenciones para construir un tipo de vista".*

3.3.1 Vista de Contexto

La vista de contexto define los principales elementos de la arquitectura, y sus interacciones con el entorno como se muestra en la Figura 3.1. Utilizamos el modelo de dominio IoT del MR-IoT como punto de vista para la generación de la vista de contexto. En otras palabras, la vista de contexto se deriva del modelo de dominio del MR-IoT. El modelo de dominio IoT permite identificar:

1. entidades físicas y entidades virtuales relacionadas;
2. recursos (al menos desde una perspectiva de funcionalidad);
3. dispositivos (u opciones de dispositivo);
4. servicios y;
5. usuarios.

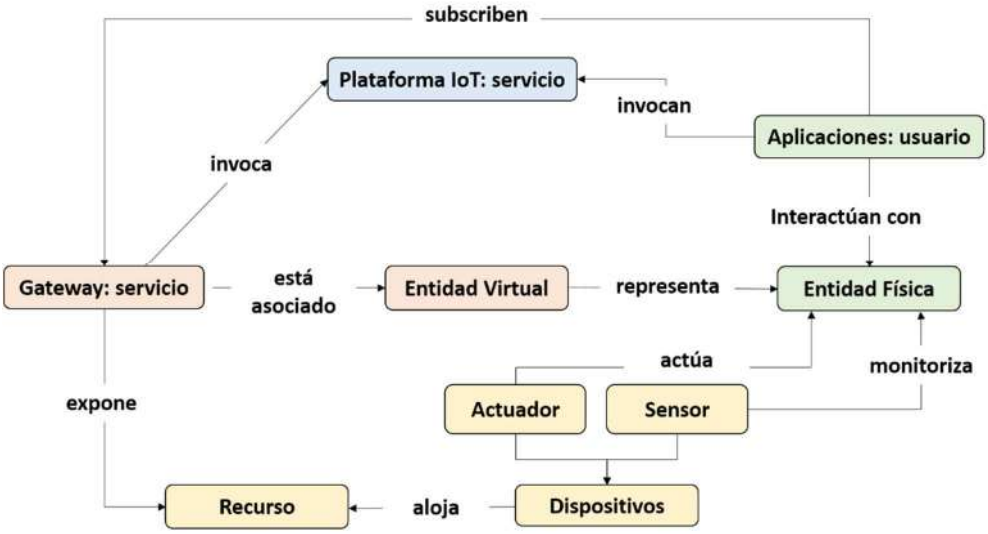


Figura 3.1: Vista de contexto de la Arquitectura.

Los *usuarios* (humanos o artefactos digitales activos (p, ej., agentes de software o aplicaciones)) interactúan con una *entidad física*. La interacción está mediada por un *servicio* (plataformas IoT o *gateway)* que está asociado con una *entidad virtual*; una representación digital de la *entidad física*. En el caso de una interacción mediada un usuario invoca o se suscribe a un *servicio*. Un *servicio* proporciona una interfaz bien definida y estandarizada, que ofrece todas las funcionalidades necesarias para interactuar con *entidades físicas* y procesos relacionados. Una descripción de una *entidad virtual* consta de un identificador único y un conjunto de atributos. Idealmente, los atributos de la *entidad virtual* proporcionan una representación sincronizada de las propiedades de la *entidad física.* Un *dispositivo* es un objeto identificable del mundo físico que es relevante para una aplicación de IoT.

Los dispositivos se clasifican en dos grupos: sensores y actuadores. Los dispositivos alojan componentes de software denominados *recursos*. Los *recursos* proporcionan información sobre o habilitan la actuación en *entidades físicas*. Los recursos se pueden abordar utilizando un esquema de direccionamiento uniforme. Para interactuar con *el recurso* (y el dispositivo) se necesita una interfaz estándar que defina las reglas y la sintaxis para la interacción. Como se ha comentado anteriormente, esta interfaz se llama *servicio*. La relación entre el *servicio* (*gateway*) y la *entidad virtual* se llama asociación. La asociación modela qué atributos de la *entidad virtual* se pueden leer y modificar a través del *servicio* específico. El *servicio* (*gateway*) puede estar asociado a cualquier número de *entidades virtuales*. Dado que la arquitectura es independiente del caso de uso y de la aplicación*, los* dispositivos, entidades físicas, entidades virtuales y recursos (que son específicos de los casos de uso en los que se aplique la arquitectura) han sido definidos a un alto nivel. Por ejemplo, en el caso de uso de transporte y logística que se detallará en el capítulo 5, la entidad física representa al camión de carga.

3.3.2 Vista Funcional

Esta vista proporciona una descripción funcional de la arquitectura. El punto de vista utilizado para construir la vista funcional de IoT son los requerimientos de la arquitectura, los cuales se asignan a los diferentes dominios o grupos funcionales. La arquitectura consta de cuatro dominios funcionales: Aplicación, *cloud*, *gateway* y dispositivo. El dominio de aplicación, *gateway* y dispositivo están alineados a los dominios de aplicación, red y dispositivo, respectivamente, de la arquitectura de referencia M2M del ETSI.

Cada dominio está compuesto por componentes funcionales (CF) necesarios para el correcto funcionamiento de la arquitectura en su conjunto. Además, otros componentes funcionales pueden integrarse ocasionalmente en la arquitectura de acuerdo con las necesidades de los escenarios específicos de aplicación. Esta característica crea una arquitectura versátil que, a diferencia de los modelos de capas tradicionales (p.ej., el modelo de referencia OSI) permite personalizar cada capa de la arquitectura en función de las necesidades específicas de cada caso de uso, de modo que es más sencillo de implementar y más fácil de modificar o cambiar sus componentes funcionales en caso de que fuera necesario. Como ejemplo, una implementación particular de la arquitectura se describe en el capítulo 4**.** Esta implementación se llama *Smart IoT Gateway For Heterogeneous Devices Interoperability* (D. C. Yacchirema & Palau, 2016)*,* que es adecuado para ejecutarse en dispositivos restringidos clase 2.

La Figura 3.2 muestra un esquema simplificado de la arquitectura propuesta. En ella, se pueden observar los principales componentes funcionales, así como sus relaciones, necesarios para conseguir la interoperabilidad entre los dispositivos IoT conectados a la arquitectura y su integración con las diferentes plataformas IoT. A partir de la interoperabilidad, los componentes funcionales también permiten crear soluciones IoT.

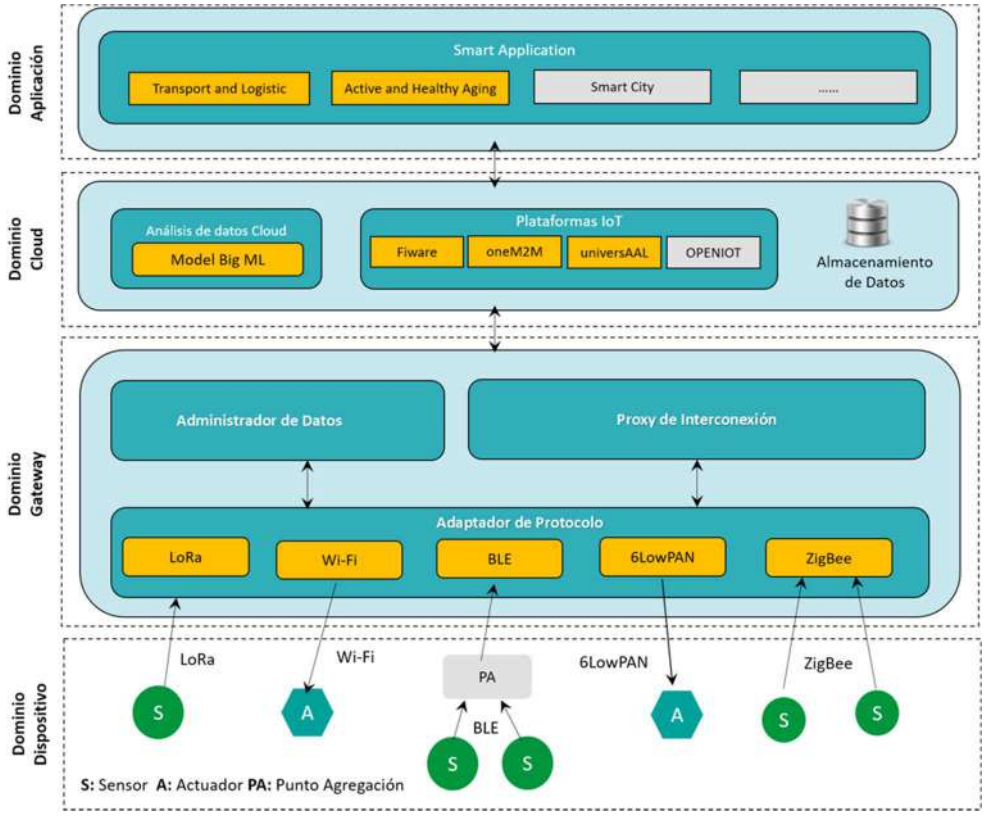


Figura 3.2: Arquitectura de interoperabilidad

A continuación, se realiza una descripción detallada de la arquitectura con un enfoque de arriba hacia abajo en relación con la Figura 3.2.

3.3.2.1 Dominio Dispositivo

Este dominio es la base de toda la arquitectura ya que se encarga de capturar información del mundo físico y reaccionar en consecuencia. Una característica importante de este dominio es la heterogeneidad creciente de los dispositivos IoT en términos de hardware, tecnologías y protocolos de comunicación utilizados. Los dispositivos IoT son nodos que pueden incluir uno o más transductores (sensores y actuadores).

Los sensores forman parte de la red troncal de los datos digitales de las soluciones IoT, estos dispositivos son capaces de detectar condiciones (eventos o cambios) en o alrededor del entorno físico al que están conectados. Un ejemplo de sensores pueden ser cámaras, sensores de temperatura, humedad, gas, entre otros.

Los actuadores por su parte, requieren una señal de control para activarse y actuar en el entorno físico. Cuando reciben la señal de control, responden convirtiendo la energía de la señal en movimiento mecánico. En el ámbito de IoT, el movimiento puede ser virtualmente de cualquier forma, como encender o apagar, abrir o cerrar, bloquear o desbloquear, etc.

Los dispositivos IoT recopilan datos de los sensores y los envía al *dominio del gateway*, a través de diferentes redes inalámbricas como BLE, ZigBee, Wi-Fi, 6loWPAN y LoRA o través de redes cableadas como Ethernet, entre otras. En ocasiones debido a los límites de las tecnologías de comunicación utilizadas, los dispositivos IoT envían los datos a puntos de acceso o agregación de datos, estos últimos representados generalmente por *smartphones*, quienes a su vez realizan cierto procesamiento y envían los datos al *dominio del gateway*. Los puntos de agregación desempeñan una función clave de interfaz entre los dispositivos IoT y *dominio del gateway*. Dado que un *smartphone* incorpora un conjunto de sensores, este dispositivo por sí mismo, puede ser considerado como un dispositivo IoT.

Un dispositivo IoT está constituido por cuatro elementos principales: i) uno o más sensores y/o actuadores; ii) un microcontrolador; iii) uno o varios dispositivos de conectividad; iv) una memoria externa y v) una unidad de administración de energía como se muestra en la Figura 3.3.

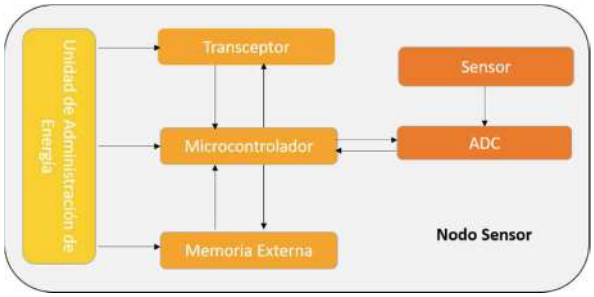


Figura 3.3: Componentes de un dispositivo IoT

El número, tipo y clase de dispositivo (consultar sección 2.4.1) utilizado dependerán del caso de uso donde se aplique la arquitectura. Los dispositivos IoT a su vez forman diferentes redes dependiendo de la tecnología de comunicación que utilicen. Como consecuencia cada red puede ser considerada como un subsistema homogéneo debido a que comparten la misma tecnología de comunicación y una capacidad de hardware similar.

3.3.2.2 Dominio Gateway

El segundo dominio, es el *dominio gateway* cuya función principal es proporcionar una capa de servicios comunes para permitir la interoperabilidad sin fisuras de dispositivos IoT y redes, y la integración de estos dispositivos con plataformas IoT, permitiendo así el desarrollo de aplicaciones inteligentes. Para ello, el *dominio gateway* se conecta en dirección norte al *dominio cloud* mediante conectividad IP, y en dirección sur hacia los dispositivos IoT con o sin conectividad IP; funcionando como un mediador entre dispositivos heterogéneos basados en diferentes tecnologías, para habilitar el intercambio dinámico del medio cableado e inalámbrico.

Según los requerimientos de las aplicaciones (p, ej., reducido tiempo de latencia) o debido al diseño de las comunicaciones o falla en las comunicaciones, una declaración de diseño clave es que no se espera que el *dominio del gateway* esté conectado permanentemente al *dominio cloud*. Por lo tanto, varias funcionalidades del *dominio cloud,* podrían considerarse en el *dominio gateway* formando de esta manera un dominio *IoT fog*.

El *dominio gateway* es responsable de realizar funciones específicas, tanto para garantizar la interoperabilidad de dispositivos como para permitir el desarrollo de soluciones IoT. Tales funciones son: abstracción de las características de *hardware*

y *software* de los dispositivos IoT, conversión de protocolos, transformación, almacenamiento, procesamiento y análisis de los datos, publicación y subscripción de eventos, e integración simplificada de dispositivos IoT con plataformas alojadas en el *dominio cloud* . Para cumplir con estas funciones, el *dominio gateway* está constituido por un conjunto de componentes funcionales con responsabilidades bien definidas como se muestra en la Figura 3.4

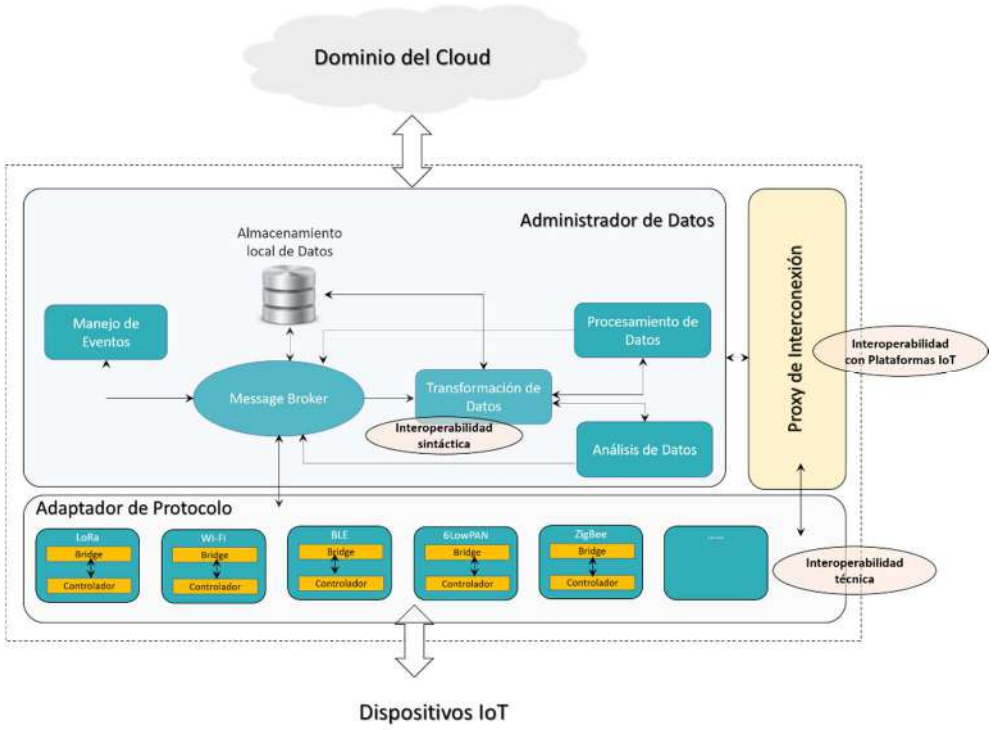


Figura 3.4: Componente Funcional Dominio de Gateway

La Figura 3.4 proporciona una vista más detallada de los componentes funcionales que resume la arquitectura interna del *dominio gateway,* en el que se pueden observar los componentes mínimos necesarios para el funcionamiento de la arquitectura planteada:

3.3.2.2.1 Adaptador de protocolo

Este componente funcional tiene por objetivo tratar con el ecosistema heterogéneo de dispositivos IoT del *dominio dispositivo* para alcanzar la interoperabilidad técnica. Su principal función consiste en coordinar las tareas de comunicación

con los dispositivos IoT (control de flujo, acceso a la red, etc) y resolver el problema de incompatibilidad de diferentes protocolos y el conflicto de mensajes entre diferentes redes. Para cumplir esta funcionalidad, este componente a su vez incorpora dos sub-componentes: controlador de dispositivo y *bridge*.

3.3.2.2.1.1 Controlador de dispositivo

Representa una implementación de la pila de protocolos de comunicación específicos admitidos por los dispositivos, ofrecen primitivas de servicio que los *bridges* pueden invocar para recibir o transmitir datos. Estos controladores se encargan de configurar la red de IoT, inicializar los parámetros de configuración del protocolo, e iniciar el descubrimiento de los dispositivos según las especificaciones del protocolo. Cada controlador dispone de una interfaz hacia los dispositivos cuyos protocolos son compatibles con la pila de protocolos implementada y otra hacia los *bridges,* que permiten la comunicación hacia ambos sentidos.

Con base en el modelo de comunicación de MR de IoT-A, se han identificado un conjunto de pilas de protocolos que crecen a partir de una tecnología de comunicación específica, representada por cada una de las redes admitidas por la arquitectura. Como se mencionó anteriormente, cada red puede ser considerada como un subsistema homogéneo, por lo que cada uno es el punto de partida para construir una pila de protocolo, iniciando a partir de la capa física de la red de sensores específica. Esta regla impone optimizaciones tecnológicas y garantiza la viabilidad en todas las redes de sensores.

La Figura 3.5, muestra un ejemplo de las pilas representadas por los controladores de dispositivo, para soportar la conectividad de dispositivos mediante el uso de las tecnologías detalladas en la sección 2.4.3.1.1.

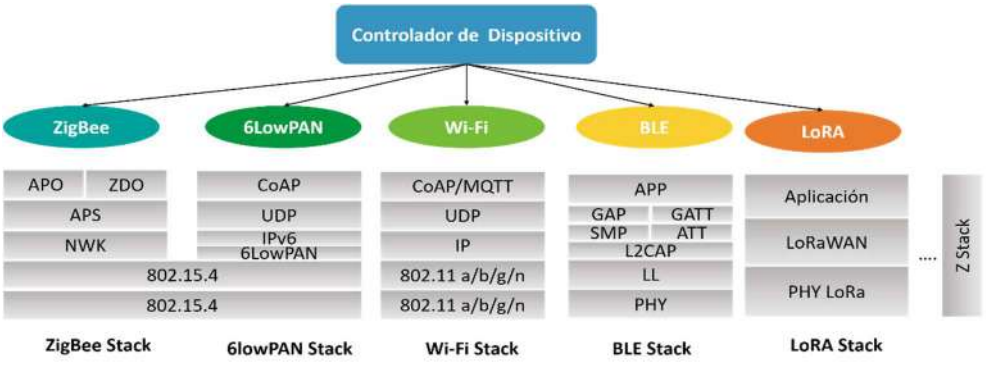


Figura 3.5: Ejemplos de pilas de protocolos de comunicación admitidas por el controlador de dispositivo

La arquitectura interoperable es lo suficientemente flexible como para permitir la compatibilidad con dispositivos que utilizan pilas de protocolos adicionales. El soporte para tales pilas se puede agregar añadiendo un nuevo controlador, representado genéricamente como "*Z Stack*" en la Figura 3.5. Un solo controlador tiene la capacidad de administrar múltiples dispositivos compatibles con la pila de protocolos implementada.

3.3.2.2.1.2 Bridge

Por cada pila de protocolo específico que admita el *controlador de dispositivo* es necesario un *bridge* que permita un acceso de bajo nivel al dispositivo para descubrir el dispositivo, registrar el dispositivo y, lo más importante obtener los valores de los recursos (sensores), así como ejecutar comandos para activar los recursos (actuadores) según las especificaciones del protocolo. Para realizar esta última tarea el bridge encapsula los datos enviados por el protocolo origen en un formato compatible con el protocolo de comunicación de destino en el dominio del *gateway*.

El *bridge* permite el descubrimiento automático de los dispositivos, excepto de aquellos que trabajan con pilas de protocolos que no admiten esta característica como el caso de la pila de comunicación LoRa. Por otra parte permite el registro de los dispositivos mediante el aprovisionamiento por parte de los administradores de la siguiente información: identificador de dispositivo, tipo de dispositivo (proporciona información de contexto relacionado con la plataforma de desarrollo del dispositivo (p.ej., ESP8266, Arduino V3, STM32 etc.), nombre de todos los recursos (sensores y actuadores) junto a sus servicios (p. ej., temperatura, humedad, gas, etc.), plataforma IoT y tipo de configuración (p.ej., ZigBee, 6lowPAN-CoAP, etc.).

El parámetro de la plataforma IoT es opcional, debido a que los dispositivos no necesariamente deben estar conectados a las plataformas para su funcionamiento.

Para cada pila de protocolos admitida y tipo de dispositivo existen configuraciones predeterminadas, así cuando un conjunto de dispositivos similares se conecte al *adaptador de protocolo*, el administrador deberá proporcionar solo el identificador del dispositivo y nombre de la pila a la cual pertenece su dispositivo. En este caso, toda la información predeterminada en la configuración se fusiona con la información del dispositivo para crear el objeto del dispositivo definitivo que se

almacenará. Las configuraciones pretenden ser un medio para simplificar el registro de dispositivos para grupos de dispositivos similares, sobre todo en el caso de dispositivos que incluyen varios sensores y actuadores. Los *brigdes* pueden admitir el registro automático de dispositivos, si las pilas de los protocolos que admiten permiten el descubrimiento automático de sus recursos (p.ej., dispositivos compatibles con la pila 6LowPAN-CoAP). En cualquier caso, posteriormente al descubrimiento y registro, la definición de recursos junto con algunos metadatos que describen características adicionales del dispositivo (p. ej., ubicación del dispositivo, marca de tiempo etc.) se anuncian al componente funcional de *administración de datos*. A partir de este momento, el *dominio del gateway* puede detectar los datos provenientes de estos dispositivos.

3.3.2.2.2 Administración de Datos

Este componente funcional tiene por objetivo respaldar la naturaleza heterogénea de los datos provenientes de los dispositivos IoT mediante funciones de transformación, almacenamiento, procesamiento y análisis de datos, a fin de generar y publicar datos de valor agregado que sean relevantes a las necesidades de los destinatarios finales autorizados. Para ello, este componente incorpora seis módulos funcionales:

3.3.2.2.2.1 Transformación de datos

Debido a que la heterogeneidad también está presente en los diferentes formatos de datos admitidos por los dispositivos IoT, un aspecto fundamental relacionado con la interoperabilidad sintáctica es la transformación de datos. La transformación de datos es el proceso de convertir un formato de datos en otro formato de datos compatible con la arquitectura. En particular, este módulo funcional es responsable de convertir el formato de datos procedentes de los dispositivos IoT para el que se ha diseñado, a un formato de datos común (adecuado y relevante) definido en la arquitectura y viceversa; utilizando una sintaxis de transferencia de datos de alto nivel (Lenguaje de representación de datos estándar), lo que permite que los datos puedan ser accesibles de forma homogénea por todos los componentes de la arquitectura que lo requieran(p. ej., para su procesamiento y análisis posterior) y por los diferentes dispositivos.

Si nos centramos en el flujo de datos, la transformación puede realizarse en primera instancia al obtener datos sin procesar (*raw data*) de los dispositivos IoT a través del módulo *adaptador de protocolo* como al actuar sobre los dispositivos IoT

o enviar notificaciones a los destinatarios externos configurados, que puede entenderse como un ciclo:

1) De heterogeneidad a homogeneidad: Recibe los datos del módulo *adaptador de protocolo* que ha sido capturado con un formato específico y asigna a los datos un formato común para su posterior procesamiento por parte del módulo de *procesamiento de datos* y/o *análisis de datos*.

2) De homogeneidad a heterogeneidad: Recibe los datos procesados (p.ej., acciones, comandos, etc.) y analizados (p.ej., predicciones) de la salida del módulo de *procesamiento y análisis de datos* respectivamente, con el fin de transformarlos al formato específico requerido por los destinatarios finales autorizados, antes de ser enviados a través del módulo de *manejo de eventos*. En el caso en el que los destinatarios finales son los dispositivos IoT transforma el formato común al formato específico (nativo) del dispositivo.

El proceso de transformación describe completamente la traslación del formato de datos origen del dispositivo IoT al formato destino, esto se ilustra en la Figura 3.6.

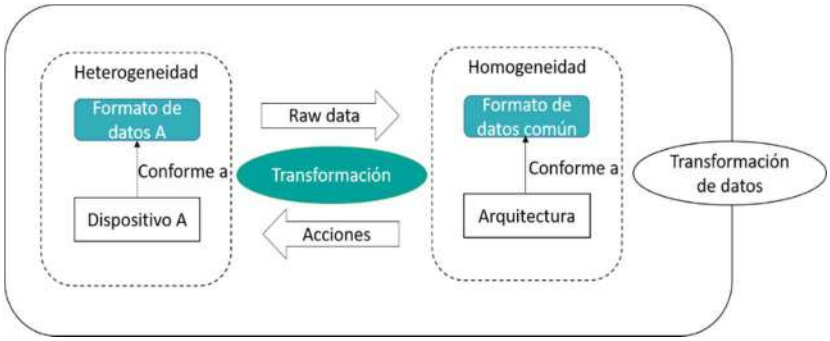


Figura 3.6: Proceso de transformación de datos

El módulo de *transformación de datos* se utiliza para alcanzar la interoperabilidad sintáctica. Por lo tanto, tiene que ser capaz de admitir los formatos de datos de IoT utilizados normalmente, en particular los que se encuentran en los casos de uso de la arquitectura. Como se muestra en la Figura 3.7, cada instancia del formato de datos tiene campos, que tienen un identificador de dispositivo, un tipo de dispositivo y una lista de atributos.

Como se muestra en la Figura 3.8, cada uno de los atributos del formato de datos interno tiene un valor real (medida) y un tipo de dato (unidad de medida o clase de dato que se maneja), también puede contener un campo metadato adicional o una lista de campos metadatos para admitir información contextual como, por ejemplo, el protocolo de comunicación subyacente, marca de tiempo de las medidas del dispositivo, ubicación del dispositivo etc.

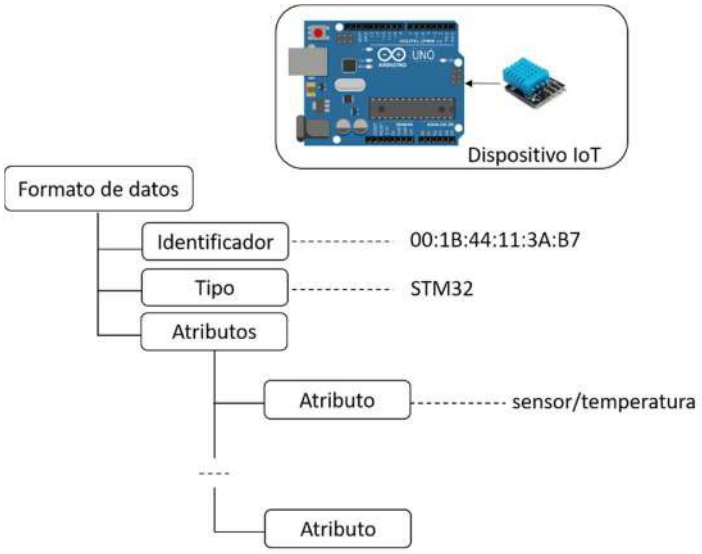


Figura 3.7: Estructura del formato de datos interno

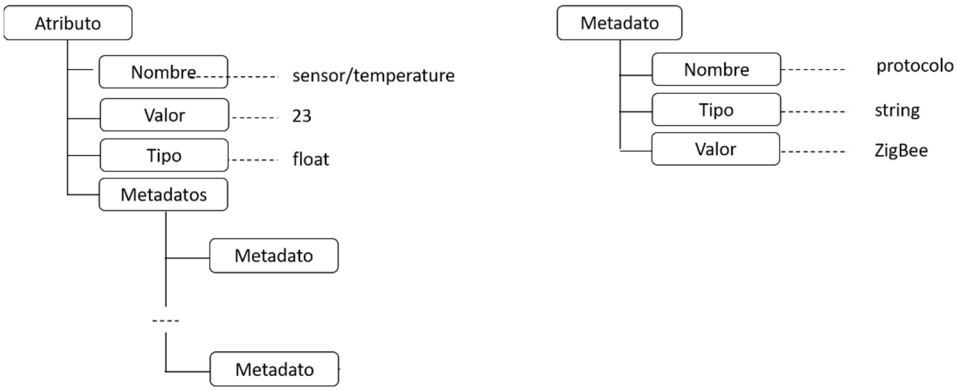


Figura 3.8: Estructura de los atributos y metadatos

A pesar de que la arquitectura, no proporciona interoperabilidad semántica, el uso de metadatos permite proporcionar descripciones semánticas de los dispositivos en línea con la vista de contexto de la arquitectura y puede mejorarse o ampliarse con el conocimiento del caso de uso de aplicación específico.

3.3.2.2.2.2 Almacenamiento local de datos

Dispone de servicios de almacenamiento local de datos. En el caso de los sensores permite almacenar los valores obtenidos en el entorno de despliegue y en el caso de los dispositivos actuadores el valor de su estado. Dependiendo del dominio de aplicación, el *dominio gateway* puede requerir enviar los datos de los dispositivos finales al *dominio cloud*, por lo que puede actuar como un almacenamiento temporal durante un cierto período de tiempo y también durante circunstancias imprevistas (p.ej., en sitios con poca conectividad de red, ancho de banda limitado, fallas en la red, cuellos de botella, etc.). Por lo tanto, para garantizar que la arquitectura pueda seguir funcionando incluso en ausencia de conectividad con el *dominio cloud* y pueda recuperar los datos sin problemas, es decir evitar cualquier pérdida, este módulo, en este caso, almacena los últimos datos entrantes de los dispositivos en un almacenamiento local no volátil.

El almacenamiento de los datos también es necesario para otras funcionalidades del *dominio gateway*. Como se ha comentado anteriormente, el *dominio gateway* también es responsable de la transformación, procesamiento y análisis de datos; estas funciones necesitan un almacenamiento temporal local, que permita el acceso a los datos inmediatamente en caso de requerirlos, así como de almacenar los datos generados de estas funciones. Metadatos como la identificación de la fuente y marcas de tiempo locales se agregan localmente a los datos almacenados para su recuperación.

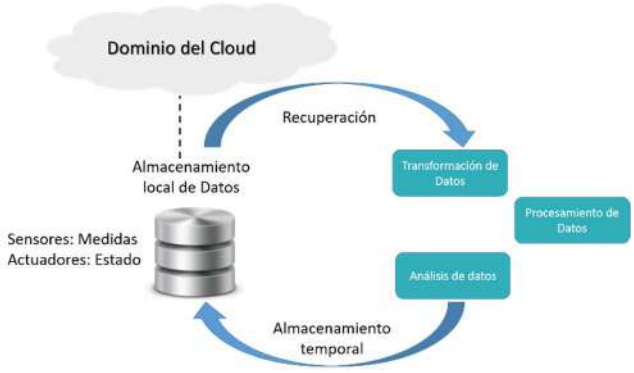


Figura 3.9: Módulo funcional almacenamiento de datos

Estos datos localmente se eliminarán de forma regular, lo que permite escalar según la disponibilidad de la memoria instalada. El almacenamiento en el *dominio gateway* hace que la arquitectura sea confiable y robusta incluso cuando la red no está disponible. La funcionalidad descrita se ilustra en la Figura 3.9.

3.3.2.2.2.3 Procesamiento de Datos

Debido a que la arquitectura propuesta habilita el desarrollo de diferentes tipos de aplicaciones IoT, el módulo funcional de procesamiento de datos proporciona inteligencia en el *dominio gateway* mediante la cual los datos transmitidos desde el *dominio dispositivo* se procesan localmente en tiempo real para responder en poco tiempo y de manera adecuada con respecto a varias condiciones.

Su principal función es la de ofrecer el procesamiento de los datos sensoriales provenientes del módulo de *transformación de datos* a fin de detectar situaciones críticas o relevantes en tiempo real para un dominio en particular. Estos datos sensoriales son considerados como eventos. Un evento es algo que ocurre o se espera que ocurra(Luckham, 2002). Este módulo procesa los eventos de entrada en base a un conjunto de reglas, es decir compara y correlaciona cada uno de los datos de los eventos de entrada con el conjunto de reglas definidas en la arquitectura y deriva nuevos eventos de salida, tal como se muestra en la Figura 3.10. El canal de comunicación para el flujo de mensajes hacia y desde el módulo procesamiento de datos se realiza a través de un *message broker,* un intermediario de mensajes representado en la Figura 3.10 por las líneas discontinuas de color verde y detallado más adelante.

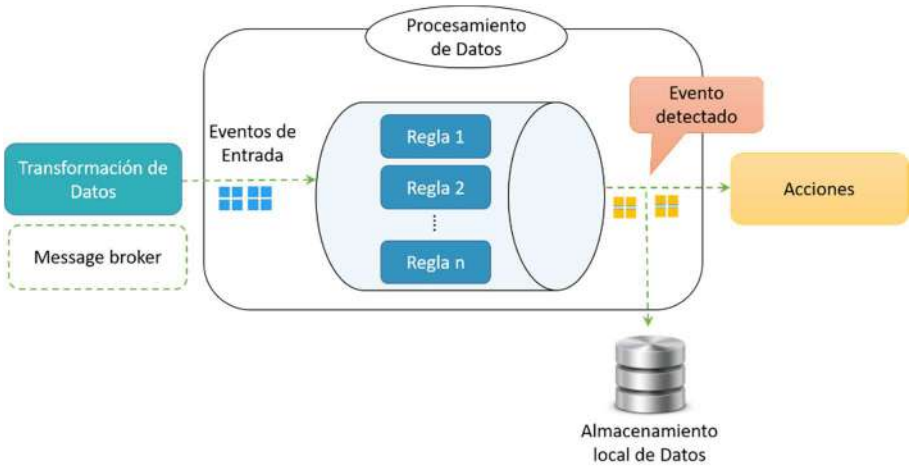


Figura 3.10: Módulo funcional procesamiento de datos

Las reglas son condiciones de activación previamente definidas que deben cumplirse para detectar dichas situaciones de interés. Estas reglas a su vez determinan las acciones específicas que deben ser ejecutadas tras su detección, tales como, enviar acciones de control a los dispositivos (p. ej., comandos), activar alertas y enviar notificaciones a las aplicaciones y destinatarios finales autorizados por el sistema.

Las reglas se especifican como tuplas compuestas de cuatro pares clave-valor de la siguiente manera:

i) Clave: representa el tipo de parámetro físico relacionado con los datos monitorizados, por ejemplo, temperatura, humedad, presión arterial, etc;

ii) Valor: valor cuantitativo que define el umbral con el que se compara los datos entrantes (un número de tipo flotante, entero o cadena);

iii) Operador: operador matemático relacional (p. ej., >, <, >=, <=, ==, !=, etc.) o lógico (p.ej., *AND* o *OR*) que, junto con el valor especificado, se aplica en los datos entrantes para identificar la ocurrencia de un evento; y

iv) Acción: acción que debe ejecutarse cuando se active la regla. Con el objetivo de no duplicar una regla, este campo también puede ser una matriz de acciones. Una acción puede tener opcionalmente un intervalo para limitar la frecuencia de la ejecución de la acción, que representa el período mínimo entre ejecuciones. Una vez que la acción se ejecuta con éxito, no se ejecutará de nuevo hasta que haya transcurrido ese período. Siguiendo esta filosofía este tipo de acciones se pueden considerar como acciones planificadas en comparación con las acciones automáticas que no tienen asociado el intervalo de tiempo.

Las reglas pueden ser tanto simples como complejas. En el primer caso, se define la regla mediante una tupla de un evento utilizando los parámetros anteriormente especificados (p. ej., [clave: temperatura, valor: 20, operador: >, Acción: Alarma: ON]). Mientras que una regla compleja puede integrar y combinar en una tupla un conjunto de eventos vinculados con operadores lógicos (p.ej., [clave: temperatura, valor: 20, operador:>] AND [clave: presencia, valor: 1, operador: ==] Acción: Enviar alerta al móvil del usuario) e incluso aplicar sobre estos eventos funciones de agregación (p. ej., MIN (mínimo), MAX (máximo), AVG (promedio), etc.).

El procesamiento de datos en el *dominio gateway* es una necesidad común en una serie de aplicaciones de IoT, por ejemplo, aquellas sensibles a la latencia como es el caso de emergencias médicas (que representa a uno de los casos de uso de aplicación de la arquitectura), que necesitan que los datos sean procesados a una alta velocidad permitiendo que las situaciones de interés puedan ser identificadas y notificadas en tiempo real, reduciendo el tiempo en la toma de decisiones. Además, el procesamiento local evita la sobrecarga de las capas de la arquitectura de nivel superior y la inundación de la red. Esta ventaja también se aprovecha al analizar los datos localmente.

3.3.2.2.2.4 Análisis de datos

Sin lugar a duda, la tendencia actual de las aplicaciones IoT es el análisis de grandes datos, provenientes de los dispositivos IoT los cuales exhiben una serie de características que parecen ser inusuales en comparación con los conjuntos de datos tradicionales, los cuales se generaban sobre una planificación consistente y cuidadosa.

Big Data originalmente hace referencia al volumen, la velocidad y la variedad de datos que es difícil de almacenar, procesar y analizar cuando se usan bases de datos, herramientas y técnicas tradicionales de procesamiento de datos (Bahga & Madisetti, 2016). Hoy en día, big *data* juega un papel importante en las aplicaciones IoT que están adoptando rápidamente datos de diferentes fuentes, que requieren ser analizados eficientemente para facilitar la toma de decisiones.

Debido a que la arquitectura propuesta permite la interoperabilidad de dispositivos que pueden proveer mucha información, además del procesamiento, el análisis de esta información es muy útil para capturar fenómenos complejos y dinámicos y respaldar la toma de decisiones. Por ejemplo, en un sistema de detección de caídas de adultos mayores, la frecuencia cardiaca, podría usarse para determinar la causa de la caída mediante el procesamiento de datos; sin embargo, una estimación precisa del tipo de caída puede no ser tan sencilla a través del procesamiento de datos. Además, un gran número de aplicaciones IoT, especialmente las aplicaciones sensibles al tiempo, como ya se ha mencionado, requieren que el análisis de los datos se realice en tiempo real (o al menos en el menor tiempo posible) y cerca de las fuentes de datos (es decir en el borde de la red), en lugar de los centros de datos como ocurre en el modelo tradicional *cloud computing*. Modelo en el que el principal problema es la mayor latencia. Esta limitación aumenta el tiempo de respuesta, que a su vez limita el tipo de soluciones

donde se puede aplicar la arquitectura cuando se requieren respuestas de tiempo crítico.

En este tipo de aplicaciones la distribución del análisis de *Big Data* se considera un factor crucial (Cerina *et al.*, 2017). Siguiendo esta filosofía, distribuimos el análisis de *Big Data* entre el *dominio cloud* representada por el *análisis de datos Cloud* (consultar sección 3.3.2.3) y el *dominio gateway* representado por el módulo *análisis de datos* con el objetivo de mejorar la escalabilidad, la eficiencia y reducir el volumen de datos que deben transferirse al *cloud* y lo más importante mantener el tiempo de respuesta requerido por cada una de las aplicaciones IoT.

El módulo *análisis de datos* permite el análisis local del flujo de datos proveniente de los dispositivos IoT mediante el uso de algoritmos de aprendizaje automático aplicados para el análisis inteligente de *Big Data*. El aprendizaje automático es un sub-campo de la informática, y es un tipo de inteligencia artificial que proporciona a las máquinas la capacidad de aprender sin programación explícita (Mahdavinejad *et al.*, 2018). Por su parte, los algoritmos de aprendizaje automático permiten entre otras funciones, realizar predicciones sobre la base de un conjunto de condiciones aplicadas a un conjunto de datos para facilitar la toma de decisiones**.**

Para realizar el análisis de datos en tiempo real, el módulo *análisis de datos* crea una instancia local de los modelos predictivos creados previamente en el *dominio cloud* a partir de diferentes algoritmos de aprendizaje tales como árboles de decisión, *ensamble*, asociaciones, *deep-learning*, etc. los cuales se describen en detalle en el caso de uso del capítulo 6. El acceso al modelo de datos en el *dominio cloud* se realiza mediante una API REST. Dependiendo de la implementación específica de la arquitectura, el *análisis de datos* aplica un modelo de datos predictivo específico al flujo de datos de los dispositivos y como resultado genera una predicción, la cual similar al módulo de *procesamiento de datos* se corresponde a un evento que será notificado a los destinatarios finales a través del módulo de *manejo de eventos*.

Las predicciones detectadas a su vez enriquecen de manera dinámica y continua al *modelo big data* creado en el *dominio cloud*. El proceso descrito se muestra en la Figura 3.11. Similar al módulo *procesamiento de datos* el flujo de mensajes se realiza a través del *message broker*.

El análisis de datos en el *dominio gateway* se realiza de acuerdo a los principios del paradigma *fog computing* que aborda las limitaciones del modelo tradicional *cloud computing*. Este enfoque permite ofrecer respuestas de baja latencia ante la presencia de eventos.

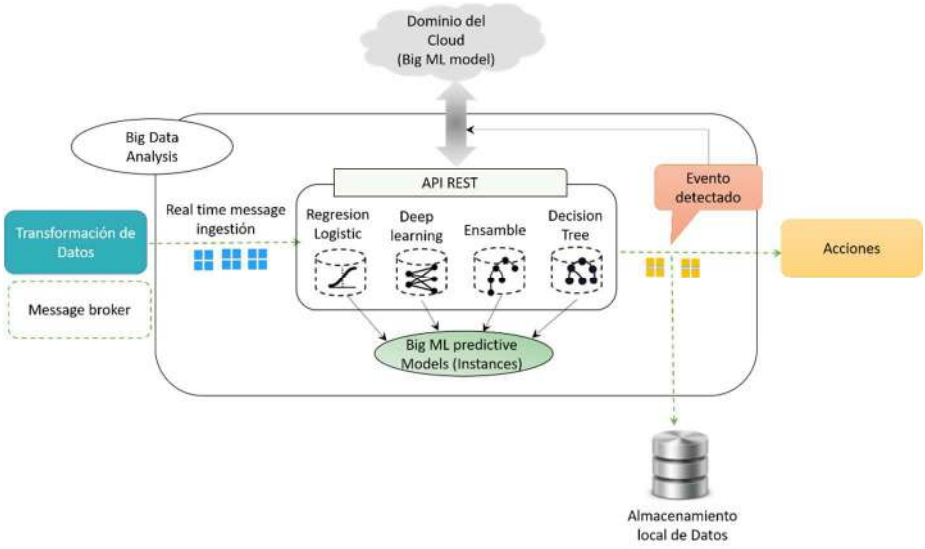


Figura 3.11: Módulo funcional análisis de datos

3.3.2.2.2.5 Manejo de eventos

Este módulo permite la notificación oportuna y el envío de los eventos generados (es decir mensajes de alerta, acciones, notificaciones, etc.) como resultado del procesamiento y análisis de datos a través de un modelo de comunicación *Publish/subscribe* a los destinatarios finales. Este modelo de comunicación permite el intercambio de datos a través de temas utilizando conexiones asíncronas. Un tema es un flujo de datos del mismo contexto mediante el cual la información puede ser catalogada, y permite a los productores publicar los datos en un servidor, llamado *broker*, el cual a su vez distribuye los datos a los consumidores subscritos a los diferentes temas.

El módulo p*rocesamiento de datos* y *análisis de datos* actúan como productores y los destinatarios finales a los que van dirigidos los eventos generados como subscriptores, mientras que el módulo *manejo de eventos* actúa como *broker*.

Los destinatarios finales (consumidores) son el lugar de destino de los eventos detectados. En la actualidad existen dos tipos de destinatarios finales, como son entidades externas y los actuadores. Las entidades externas pueden ser aplicaciones móviles o *web* que a través de una interfaz gráfica de usuario permiten visualizar los eventos generados. Los actuadores, son los más importantes, ya que son

los responsables de llevar a cabo acciones concretas, a partir de los eventos detectados.

Los consumidores deben subscribirse a los temas de interés para recibir la información publicada en cada uno de los temas específicos. Los temas se definen de acuerdo a una estructura jerárquica que permite establecer relaciones padre-hijo. Cada tema puede ser separado en varios niveles utilizando el separador de nivel (/). Por ejemplo, si una aplicación del consumidor está interesada en los eventos generados de un sensor de humedad administrado por un dispositivo IoT, tendría que subscribirse al tema Arduino0001/sensor/humedad. Para esta subscripción, el módulo *manejo de eventos* obtendrá todos los eventos publicados en este tema, y notificará a los consumidores suscritos al mismo.

El principio de diseño basado en una estructura jerárquica permite el uso de comodines con el objetivo de expresar un interés no solo en un recurso particular de un dispositivo IoT, sino también en un subconjunto de ellos. Por ejemplo, si una aplicación del consumidor está interesada en eventos generados por todos los sensores del dispositivo Arduino0001 tendría que subscribirse al tema Arduino0001/sensor/+. Además, si la aplicación del consumidor está interesado por la información tanto de sensores como de actuadores del dispositivo IoT Arduino0001, tendría que subscribirse al tema Arduino0001/#. Es decir, las subscripciones admiten el uso de dos tipos de comodines:

- Multinivel (#): admite cualquier nivel por debajo del nivel mostrado en la cadena.
- Simple (+): admite solo hasta el nivel mostrado en la cadena.

Las suscripciones se definen mediante mensajes que se componen de campos específicos. La estructura de un mensaje de subscripción se muestra en la Tabla 3.2.

Tabla 3.2: Estructura de un mensaje de subscripción

Parámetro	Descripción
Tema	Especifica el tema de interés al que se quiere subscribir el consumidor,
Servidor *broker*	Define la IP o el nombre del servidor que gestiona los temas a los que el consumidor desea subscribirse.

Payload format	Especifica el lenguaje de descripción que se utilizará para definir la carga útil del mensaje del tema. JSON se utiliza de forma predeterminada.
QoS	Define el nivel de QoS con el que desea recibir los mensajes el consumidor. Se manejan tres niveles de QoS: 1, 2 y 3.
Número de mensajes	Indica el número máximo de mensajes que el consumidor desea recibir sobre el tema. Posterior a ello, ya no recibirá más mensajes. Por defecto está opción no se utiliza.
Autenticación	Define el nombre de usuario y contraseña para acceder al tema.

Cabe señalar que las funciones *brokering* que utilizan el mecanismo de *publicación/subcripción* están disponibles en otros módulos funcionales de la arquitectura, como por ejemplo en el módulo *broker message* descrito a continuación. Esto no debe considerarse como una redundancia inútil: algunas características se distribuyen deliberadamente a diferentes niveles de granularidad, para brindar confiabilidad, modularidad a toda la arquitectura y también para despachar la carga de datos de manera apropiada a diferentes niveles.

3.3.2.2.2.1 Broker message

Este módulo orquesta la comunicación entre todos los componentes del *dominio gateway* y, por extensión, entre el *dominio dispositivo y el dominio cloud* a través de un modelo de *publicación/subscripción.* Es decir, permite la trasferencia de los datos de los dispositivos IoT al *dominio cloud* para su almacenamiento permanente y en el caso de los eventos generados por el módulo *análisis de datos* para retroalimentar los datos de entrenamiento a partir de los cuales se crea el modelo *Big Data.*

Por lo tanto, el funcionamiento del modelo de comunicación basado en el mecanismo *publish/subscribe* es similar al modelo de comunicación del módulo *manejo de eventos;* cada módulo necesita subscribirse al *broker message* para recuperar los datos que necesiten. El *broker message* se comunica de forma activa y simultánea con los diferentes grupos funcionales y/o módulos para obtener exactamente la información que requieren los diferentes subscriptores. Sin embargo, sus objetivos de diseño son muy diferentes, debido a que este módulo permite manejar el flujo de datos interno de la arquitectura.

El diagrama de la Figura 3.12 muestra el flujo de datos si un adaptador envía información al *dominio del gateway* utilizando el modelo *publish/subscribe* proporcionado por el *broker message*.

- **Paso 1:** Inicialmente el dispositivo sensor (temperatura) registrado con éxito en el *gateway* utilizando el *adaptador de protocolo* adyacente *(Adaptador 1)* comienza a enviar datos.

- **Paso 2:** El *adaptador de protocolo* envía la información de los dispositivos IoT hacia el *administrador de datos* en el *dominio del gateway*.

- **Paso 3:** El *broker message* almacena los datos en la base de datos por medio del módulo *almacenamiento local de datos* y los propaga al módulo *transformación de datos*.

- **Paso 4:** El módulo *transformación de datos* realiza la transformación de datos antes del procesamiento y/o análisis de datos según el dominio de aplicación de la arquitectura.

- **Paso 5.** El módulo de *procesamiento de datos*, cada vez que una regla se activa genera un evento de salida y envía el evento de salida hacia el módulo t*ransformación de datos*.

- **Paso 6.** Similar al módulo *procesamiento de datos*, el módulo a*nálisis de datos*, cada vez que se genera una predicción, envía el evento de salida hacia el módulo t*ransformación de datos*.

- **Paso 7.** El módulo t*ransformación de datos* inicia un proceso de publicación de los datos de los eventos transformados a un tema específico en el *broker message*.
- **Paso 8.** El *broker message* a su vez publica los datos de los eventos al módulo *manejo de eventos*.

- **Paso 9.** El módulo *manejo de eventos inicia* un proceso de notificación de la información de los eventos (notificaciones, acciones y/o predicciones) a los consumidores suscritos a estos temas.

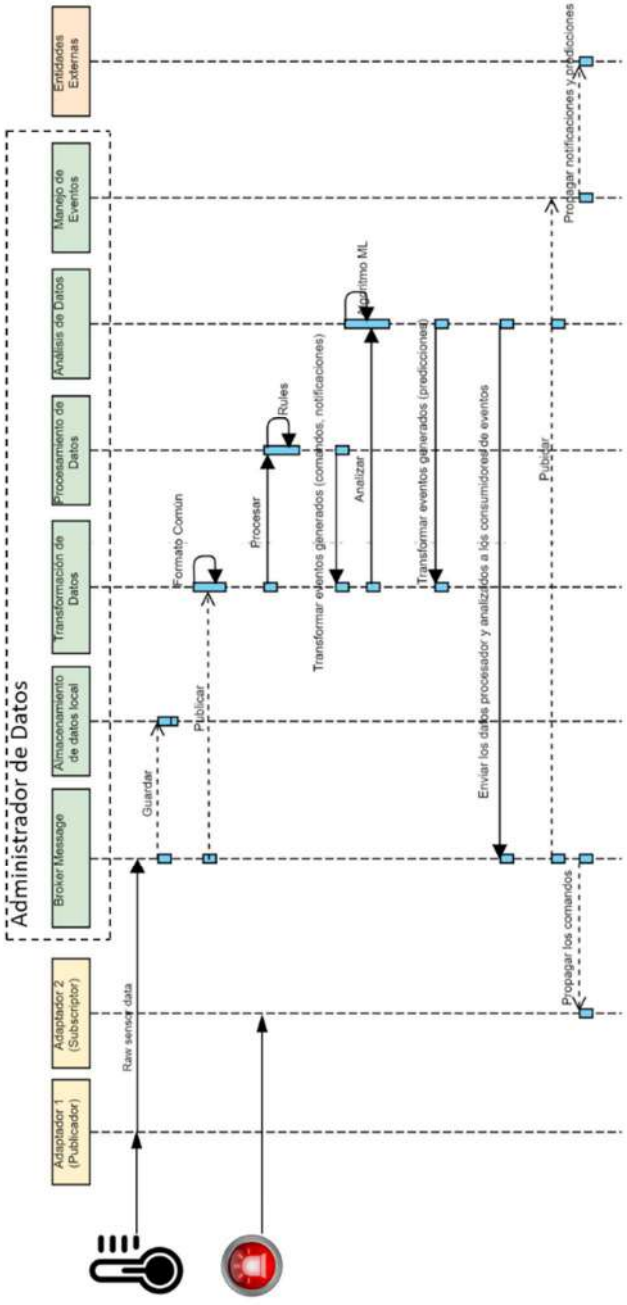


Figura 3.12: Flujo de datos de la arquitectura utilizando el mecanismo *publish/ subscribe*

3.3.2.2.3 Proxy de Interconexión

Varias plataformas IoT pueden ser instanciadas en el *dominio cloud*. El componente *proxy de interconexión* es responsable de habilitar la interacción entre los dispositivos registrados en el *dominio gateway* y las plataformas IoT disponibles en el *dominio cloud*. Dependiendo del dominio de aplicación y caso de uso, no todos los dispositivos disponibles en el *dominio gateway* estarán vinculados a las plataformas IoT existentes. Se accede a los dispositivos conectados a una plataforma IoT a través de la interacción entre las plataformas IoT y este componente, mientras que se accede a los dispositivos que no están vinculados a una plataforma IoT directamente a través del *adaptador de protocolo.*

La implementación del *proxy de interconexión* en el *dominio gateway,* ha dado lugar a la definición de una nueva arquitectura de interconexión descrita en detalle en el capítulo 5. Esta arquitectura permite que el *dominio gateway* actúe como cliente de las plataformas IoT, por lo que es responsable de la interconexión de dispositivos heterogéneos *legacy* en las plataformas IoT existentes.

3.3.2.3 Dominio Cloud

Este dominio se encarga de proporcionar la infraestructura (hardware y software) necesaria para almacenar los datos pre-procesados recibidos del *dominio gateway*. Estos datos estarán a disposición del *dominio aplicación* para su posterior explotación. Servidores y capacidades de almacenamiento constituyen la infraestructura de hardware que puede ser implementada en cualquier lugar del mundo. Su acceso se realiza a través de internet. Las *plataformas IoT*, el componente *análisis de datos cloud* constituyen la infraestructura del software de esta capa. Cabe señalar que en la arquitectura, *clouds* privadas se desarrollan para admitir las capacidades de computación en el *cloud.*

3.3.2.3.1 Plataformas IoT

Existen varias plataformas de referencias de IoT (p.ej., FIWARE, SOFIA2, universAAL, OpenIoT, SENIORSOME e IoTivity) y la integración y compatibilidad de los dispositivos IoT con algunas de ella es uno de los objetivos de esta tesis. El dominio *Cloud* está constituido por las plataformas IoT que forman parte de esta tesis, a saber, FIWARE y oneM2M. Sus características principales se explican ampliamente en el capítulo 5.

Una plataforma IoT provee un conjunto de capacidades de servicio común y recursos que diversas aplicaciones pueden aprovechar. Algunos ejemplos de tales

capacidades incluyen administración de datos, subscripciones, aprovisionamiento, publicación de mensajes, seguridad, etc. Estas capacidades de servicio permiten que las aplicaciones interactúen con los dispositivos. Además, cada plataforma provee una API uniforme para el acceso a los recursos, la autenticación y la autorización en la plataforma. Por lo tanto, el acceso a los recursos se realiza directamente entre las aplicaciones del *dominio de aplicación* y *las plataformas IoT* para que las plataformas puedan mantener un control completo sobre sus datos. Una plataforma de IoT también suele denominarse *middleware* de IoT, lo que subraya su funcionalidad como la de un mediador entre el *dominio del gateway* y *el dominio de aplicación*.

La arquitectura de interconexión definida en el capítulo 5 habilita que los dispositivos puedan estar integrados con diferentes plataformas IoT alojadas en el dominio *cloud*, lo que evita el bloqueo de los clientes a una plataforma de IoT y a un proveedor de IoT específico, además de permitir el desarrollo de nuevas aplicaciones de IoT multiplataforma.

3.3.2.3.2 Análisis de datos *cloud*

El principal componente es el *model Big ML* que permite la creación y entrenamiento de modelos de predicción y de diagnóstico basados en algoritmos de aprendizaje automático para la predicción efectiva (en el *dominio gateway*) de un potencial evento en los datos monitorizados de los dispositivos IoT y/o descubrimiento de relaciones significativas en estos datos.

Los algoritmos de aprendizaje automático generalmente se clasifican en tres categorías, que incluyen algoritmos de aprendizaje supervisado (p.ej., clasificación y regresión), algoritmos de aprendizaje no supervisado (p.ej., agrupación) y algoritmos de aprendizaje semi-supervisado que combinan los dos algoritmos anteriores para poder clasificar de manera adecuada. De acuerdo a los requerimientos y necesidades de los sistemas IoT en los cuales se aplique la arquitectura, el *model Big ML* creará un modelo de aprendizaje específico.

Para crear el *model Big ML*, un proceso típico de aprendizaje supervisado se lleva a cabo en el dominio *cloud* como se ilustra en la Figura 3.13.

Este proceso requiere dos tipos de datos: datos de entrenamiento y pruebas. Los datos de entrenamiento consisten en muestras de vectores de entrada junto con sus correspondientes vectores objetivos, también conocidos como etiquetas.

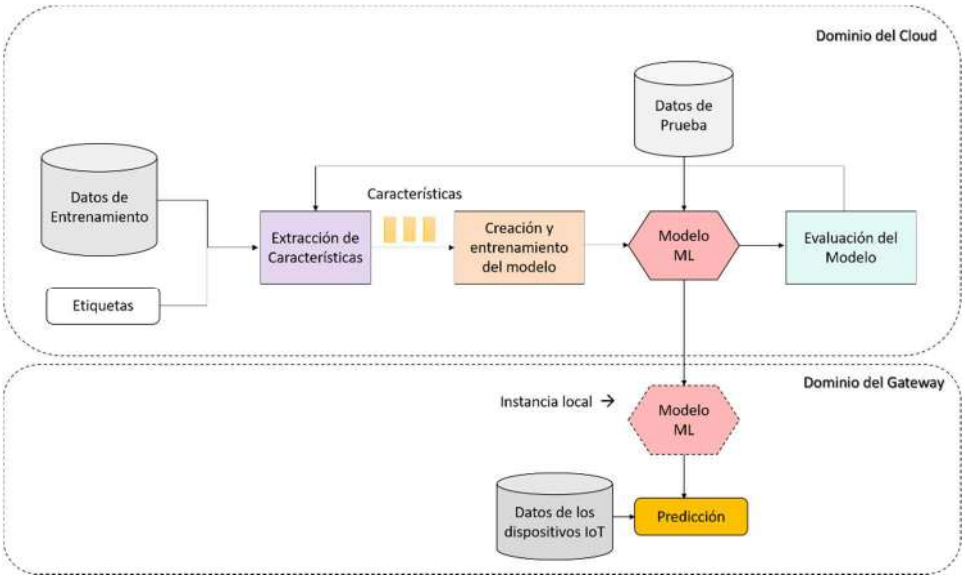


Figura 3.13: Proceso del aprendizaje supervisado ejecutado en el dominio Cloud para crear el modelo ML (parte superior) e instancia local del modelo ML en el dominio del gateway (parte inferior).

Los datos de entrenamiento pueden ser datos sin procesar provenientes de diferentes fuentes, incluidos los dispositivos IoT, por ejemplo, datos en tiempo real de un flujo de datos continuo o discreto, datos en tiempo no real, datos de los sensores de los dispositivos, conjunto de datos abiertos o cualquier combinación de ellos. Un conjunto de características o atributos se extrae como entrada a un algoritmo de aprendizaje del conjunto de datos de entrenamiento en función de las etiquetas. El conjunto de datos resultante de la extracción se utiliza para crear y entrenar el *model Big ML*. Mientras que los conjuntos de datos de prueba se utilizan para evaluar el modelo. Una vez que se evalúa el modelo, el mismo se usa para predecir el resultado potencial de un evento. Es decir, como se comentó anteriormente, el módulo *análisis de datos* instancia este modelo y aplica a los datos obtenidos de los dispositivos IoT para el diagnóstico y pronóstico en tiempo real.

3.3.2.4 Dominio de aplicación

En este dominio, la información procesada estará disponible para el usuario a través de aplicaciones *web* vía interfaces gráficas de usuario (GUI, por sus siglas en inglés, *Graphical user interface*) para la toma de decisiones. Las GUI actuales se basan en dispositivos móviles inteligentes y computadoras portátiles para nave-

gar por la información procesada y obtener los resultados. Como ya se ha mencionado, el *dominio gateway* no siempre está conectado al *dominio cloud,* por lo que dependiendo de los requerimientos las aplicaciones se ejecutarán directamente en el *dominio gateway*.

3.4 Conclusiones

La existencia de diferentes dispositivos heterogéneos operando sobre diferentes redes es uno de los principales obstáculos para la adopción y el despliegue de IoT. Este capítulo propone una arquitectura de soporte para la interoperabilidad de dispositivos heterogéneos en IoT. En él se han identificado los principales requerimientos considerando la naturaleza multifacética de IoT (características del dominio IoT, capacidades limitadas de los dispositivos y redes y, características de implementación de varias aplicaciones IoT), y en respuesta a los mismos, se presenta una arquitectura modular que permite la interoperabilidad técnica y sintáctica de los dispositivos IoT, así como su integración con plataformas IoT.

Esta arquitectura usa el patrón de comunicación D2G y está alineada con las ideas propuestas por el modelo de referencia arquitectónico IoT-ARM así como con la arquitectura M2M del ETSI. La definición formal de la arquitectura se ha realizado a través de vistas arquitectónicas: contexto y funcional. En la vista de contexto se ha definido los principales elementos de la arquitectura y sus relaciones. En la vista funcional se ha definido y descrito en detalle los dominios, componentes y módulos funcionales para cubrir los requerimientos planteados.

La arquitectura propuesta implementa dominios, componentes y módulos funcionales independientes pero compatibles y colaborativos entre sí, haciendo posible la comunicación e intercambio de información entre dispositivos heterogéneos; manejando adecuadamente la conversión de protocolos y la representación sintáctica de datos para garantizar la interoperabilidad entre dispositivos independientemente del formato de datos, protocolos y tecnologías de comunicación subyacente de cada dispositivo. Además de permitir la creación de aplicaciones IoT.

La modularidad y versatilidad de la arquitectura permite personalizar cada dominio de la arquitectura en función de las necesidades específicas del dominio de aplicación IoT donde se vaya a implementar la arquitectura. En el siguiente capítulo se presenta un *Smart IoT Gateway For Heterogeneous Devices Interoperability* que fue desarrollado durante el transcurso de esta tesis, como un ejemplo de una implementación particular de la arquitectura.

Capítulo 4

Smart IoT gateway para la interoperabilidad de dispositivos heterogéneos

«La innovación significativa no se logra hablando de nuevas ideas: hay que construir y probar prototipos.»

Peter Senge (1947)

4.1 Introducción

El *smart IoT gateway* proporciona soporte a la interoperabilidad de dispositivos en varios dominios de aplicación de IoT, en este capítulo para describir su implementación lo hemos enmarcado dentro de un sistema prototipo para AAL (por sus siglas en inglés *Ambiente assisted living*), AAL-IoTSys.

El *smart IoT gateway* es una versión simplificada de la arquitectura para habilitar la interoperabilidad técnica y sintáctica de dispositivos heterogéneos en entornos restringidos (dispositivos y redes restringidas) mientras se mantienen las características y módulos funcionales clave de la arquitectura global para la creación de soluciones IoT.

Este capítulo comienza con una descripción general del *smart IoT gateway*. A continuación, se describe cómo el *smart IoT gateway* cumple con los requisitos de la arquitectura de interoperabilidad de dispositivos físicos definidos en el capítulo

3. Posteriormente, se presenta una propuesta de caso de uso, donde se muestra y valida la potencialidad del *smart IoT gateway*. Luego, se realiza la evaluación del rendimiento del *smart IoT gateway* en términos de CPU, RAM y consumo de energía. Finalmente se realiza un análisis de los trabajos relacionados con el diseño e implementación de *gateways*, y se presentan las conclusiones del capítulo.

Este capítulo está basado en las publicaciones (D. C. Yacchirema & Palau, 2016; D. C. Yacchirema *et al.*, 2016; D. Yacchirema *et al.*, 2017; Diana Yacchirema, Palau, *et al.*, 2019).

4.2 Arquitectura smart IoT gateway

La idea inicial de nuestra primera versión embrionaria, *smart IoT gateway* tenía por objetivo implementar una versión simplificada de la arquitectura para habilitar la comunicación sin fisuras entre dispositivos heterogéneos en entornos restringidos. La Figura 4.1, muestra la arquitectura funcional de la versión *smart IoT gateway* basada en la arquitectura presentada en el capítulo 3. En esta primera versión el módulo *análisis de datos* y el componente funcional *proxy de interconexión* no están presentes.

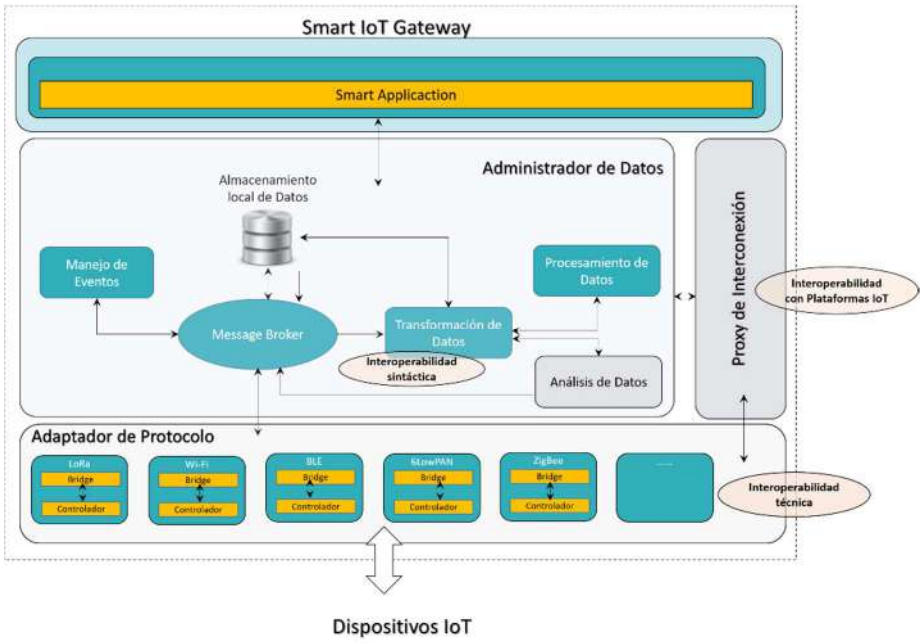


Figura 4.1: Versión embrionaria "*Smart IoT gateway"* de la arquitectura

Cada uno de estos módulos ha sido explicado en detalle en el capítulo 3, por lo que en este capítulo se procederá con la descripción de la implementación de cada uno de ellos.

El *smart IoT gateway* normalmente promueve las implementaciones basadas en la computación local más allá de los enfoques ubicuos centrados exclusivamente en la computación *cloud,* que como ya se ha mencionado este tipo de implementaciones son propensas a generar problemas de latencia, rendimiento, ancho de banda, etc.

La computación *fog* permite satisfacer en tiempo real la demanda computacional de aplicaciones sensibles a la latencia, que requieren que el procesamiento de los datos se realice lo más cerca posible de los dispositivos finales (sensores y actuadores de IoT). De esta manera, la computación *fog* puede funcionar de manera eficiente en términos de latencia, tráfico de red, consumo de energía, distribución de contenido, etc.(Mahmud *et al.*, 2018). La computación *fog* en IoT facilita el conocimiento de la ubicación y el soporte de movilidad de los dispositivos, así como las interacciones en tiempo real y la interoperabilidad.

Con base en la computación *fog*, el *smart IoT gateway* permite la ejecución de varias funciones (almacenamiento, procesamiento, notificación de eventos) a nivel local y extiende la ejecución de aplicaciones del dominio *cloud,* también a nivel local.

Los beneficios de esta versión alternativa pueden mejorar significativamente el rendimiento de las funciones que en su conjunto provee la arquitectura. Por ejemplo, pequeñas *smart IoT gateways* pueden ser desplegadas en ambientes inteligentes compuestos por un número reducido de dispositivos IoT. También, pueden acelerar la velocidad en la toma de decisiones al permitir la ejecución de funciones a nivel local. No obstante, el *smart IoT gateway* también ha sido diseñado para permitir el análisis distribuido, por lo tanto, es compatible con la computación *fog y cloud* como se demostrará en el capítulo 6.

La funcionalidad y el rendimiento del *smart IoT gateway* se probó a través de implementaciones de laboratorio (*testbed*) aplicadas a varios casos de estudio AAL que promovieron trabajos de investigación relacionados (D. C. Yacchirema & Palau, 2016; D. C. Yacchirema *et al.*, 2016; D. Yacchirema *et al.*, 2017). Estos trabajos de investigación consistieron en desplegar entornos inteligentes basados en el *smart IoT gateway* dirigidos a administrar y entregar datos asociados a la salud personal y al entorno inteligente que promueven servicios de vida asistida para el envejecimiento activo y saludable de las personas mayores dentro del hogar (p.ej., detección de notificación de emergencias de salud, monitorización de la

frecuencia cardiaca de los adultos mayores, etc.) a partir de la interoperabilidad de dispositivos IoT. Particularmente en (Diana Yacchirema *et al.*, 2017), se construyó un sistema prototipo AAL-IoTSys cuyo componente clave es el *smart IoT gateway* que proporciona la interoperabilidad de dispositivos. Su implementación la hemos enmarcado dentro de este sistema prototipo para AAL.

4.3 Relación del smart IoT gateway con la arquitectura global

En la Tabla 4.1 se describe cómo la versión *smart IoT gateway* implementada, se adecua a los requerimientos de la arquitectura.

Tabla 4.1: Cumplimiento de los requerimientos de la arquitectura por parte del *smart IoT gateway*

Requerimiento	Cumplimiento
R1. Soporte multi-protocolo a nivel de capa física, enlace y de red	*Smart IoT gateway* soporta diferentes pilas de protocolos a través de adaptadores que están construidos por varios módulos e interfaces de comunicación para permitir la conectividad y comunicación de dispositivos heterogéneos que admiten una variedad de tecnologías y protocolos de comunicación Capa física y de enlace: 802.15.4, 802.11 b/g/n, ZigBee, Wi-Fi, 6LowPAN y LoRa. Capa de red: IPv4 e IPv6.
R2. Conversión de protocolos	*Smart IoT gateway* implementa *bridges* que enfrentan el desafío de la heterogeneidad de los dispositivos, descomprimiendo la carga útil de los paquetes y asignado un formato de paquete, de acuerdo al protocolo de comunicación destino. Por ejemplo, para los dispositivos que operan en redes 6LowPAN el *bridge* crea un túnel que es utilizado por los dispositivos de esta red para interoperar con el resto del sistema, es decir el túnel permite la conversión entre protocolos 6LowPAN-IPv6/IPv4. El detalle de la implementación de los mecanismos de conversión admitidos por los *bridges* para cada una de las tecnologías de red se describe en la sección 4.5.2.3.

R3. Soporte de protocolos IoT optimizados para las comunicaciones M2M	Con base en las restricciones de los dispositivos IoT analizadas en el capítulo 2. El *smart IoT gateway* utiliza los protocolos IoT: CoAP y MQTT para las comunicaciones M2M, es decir para la transferencia eficiente de datos entre redes de sensores y el gateway. CoAP y MQTT solventan la comunicación óptima a nivel de capa de aplicación con los dispositivos que operan en redes basadas en IP y con restricciones (bajo ancho de banda, tasas de transferencia baja, alta latencia). Como se analizó en el capítulo 2, estos protocolos son abiertos, sencillos, consumen muy poca energía, requieren pocos recursos (procesador, memoria) y utilizan un mínimo de ancho de banda. Además, permitir un tiempo de respuesta superior al resto de protocolos web actuales (p.ej., HTTP). Cabe indicar que el *smart IoT gatewa* aborda parcialmente este requerimiento, debido a que las redes no IP como ZigBee y LoRa no soportar estos protocolos a nivel de capa de aplicación.
R4. Intercambio de información bidireccional	Como ya se ha comentado, el *smart IoT gateway* contiene adaptadores a través de los cuales recibe la información de los sensores y envía las acciones de control a los actuadores, utilizando el protocolo MQTT. Esta interacción bidireccional es habilitada por el *smart IoT gateway* debido a que implementa un CEP (del inglés *complex event processing*) basado en reglas para analizar la información de los sensores y emitir acciones en consecuencia.
R5. Almacenamiento de datos	*Smart IoT gateway* incorpora el servidor de base de datos, MySQL para permitir el almacenamiento local de los datos obtenidos a partir de los dispositivos IoT, y el almacenamiento temporal de los datos obtenidos tras la ejecución de las funciones del gateway.
R6. Procesamiento y análisis de datos casi en tiempo real	*Smart IoT gateway* incorpora un CEP para el procesamiento local de los datos, utilizando un enfoque *fog* computing, debido a que ha sido implementado en dispositivos COTS que facilitan la movilidad y el procesamiento, cerca de los dispositivos IoT.

	En este capítulo el *smart IoT gateway* aborda parcialmente este requerimiento, debido a que no incorpora la funcionalidad del análisis de datos, que es incorporada en el capítulo 6.
R7. Comunicaciones periódicas y/o basadas en eventos	*El smart IoT gateway* dependiendo de la red en la que opera el dispositivo IoT, permite dos tipos de comunicación cliente/servidor (comunicaciones periódicas) y publicación subscripción (comunicaciones basada en eventos).
R8. Notación sintáctica	*Smart IoT gateway* asigna un formato común a los datos basado en atributos y metados utilizando el formato de datos JSON.
R9. Seguridad .	*Smart IoT gateway* notifica los eventos a los usuarios finales y/o actuadores, utilizando el protocolo MQTT, que ha sido configurado para proporcionar la autenticación basada en usuario/contraseña y la comunicación cifrada mediante SSL/TLS. La autenticación permite securizar el acceso a los *topics* que gestiona el servidor MQTT incorporado en el *smart IoT gateway* para el manejo de eventos.
R10. Calidad de servicio	*Smart IoT gateway* a través del protocolo MQTT admite tres niveles de calidad de servicio dependiendo de los requerimientos de la aplicación: *QoS 0 (***Como máximo una vez***),* puede haber perdida de mensajes, no se hacen retransmisiones. QoS 1(**Al menos una vez),** se asegura que lleguen los mensajes, pero se pueden producir duplicados. *QoS 2* (**Como máximo una vez**), se asegura que el mensaje llegue exactamente una vez. La mejor opción cuándo la duplicación del mensaje no es aceptable.
R11. Integración con plataformas IoT estándar	En la versión *smart IoT gateway* este requerimiento no es abordado. No obstante, una extensión del *smart IoT gateway* que cubre este requerimiento se describe en el capítulo 5.

4.4 Caso de estudio AAL (Ambient assisted living)

AAL representa una de las aplicaciones más prometedoras de IoT debido a su influencia en la calidad de vida y la salud de las personas mayores. Las soluciones AAL buscan reducir el impacto de la tendencia creciente del número de personas mayores en nuestra sociedad proporcionando entornos inteligentes para que las personas mayores continúen viviendo en sus propios hogares el mayor tiempo posible con una buena calidad de vida, mientras mantienen su independencia. Sin embargo, debido a la naturaleza diversa de los dispositivos existentes (provenientes de diversos fabricantes y sectores industriales incluida la tecnología médica, domótica, electrónica de consumo, etc.) que pueden ser utilizados en entornos AAL, asegurar la interoperabilidad entre ellos es importante para proporcionar soluciones flexibles y adecuadas a las necesidades particulares de un individuo en un contexto dado.

En tal sentido, se ha implementado el sistema prototipo AAL-IoTSys (Diana Yacchirema *et al.*, 2017) que integra al *smart IoT gateway* como componente clave para ofrecer una arquitectura completa que permite interconectar dispositivos heterogéneos, facilitando así la construcción de aplicaciones de vida asistida a partir de los datos proporcionados por los dispositivos IoT, para promover la vida independiente y mejorar la calidad de vida de las personas mayores.

Las principales funcionalidades que ofrece la arquitectura del sistema son

(i) Monitorizar la movilidad de las personas mayores en ambientes interiores, a fin de emitir señales que habilitan la ejecución de acciones; por ejemplo, encender la luz ante la detección de una señal de presencia.

(ii) Monitorizar la localización de las personas mayores en ambientes exteriores, a fin de rastrear la ubicación de las personas mayores en lugares externos (en un radio de 1km) como calles, parques, etc.

(iii) Monitorizar las condiciones ambientales del hogar para realizar un diagnóstico rápido de situaciones que pueden poner en peligro la salud de las personas mayores.

(iv) Detectar y notificar situaciones de riesgo para la salud de las personas mayores; por ejemplo, cuando se produce una medida inusual en el ritmo cardíaco (fuera del rango establecido), en la que la persona mayor puede necesitar apoyo médico.

(v) Almacenar y presentar la información a usuarios externos responsables de la salud de las personas mayores; por ejemplo, a familiares y personal de cuidado (médicos y cuidadores).

La arquitectura del sistema AAL-IoTSys integra una visión funcional que facilita entender como el *smart IoT gateway*, afronta el reto de la interoperabilidad de dispositivos. La Figura 4.2 muestra la arquitectura del sistema AAL-IoTSys que incluye tres componentes principales:

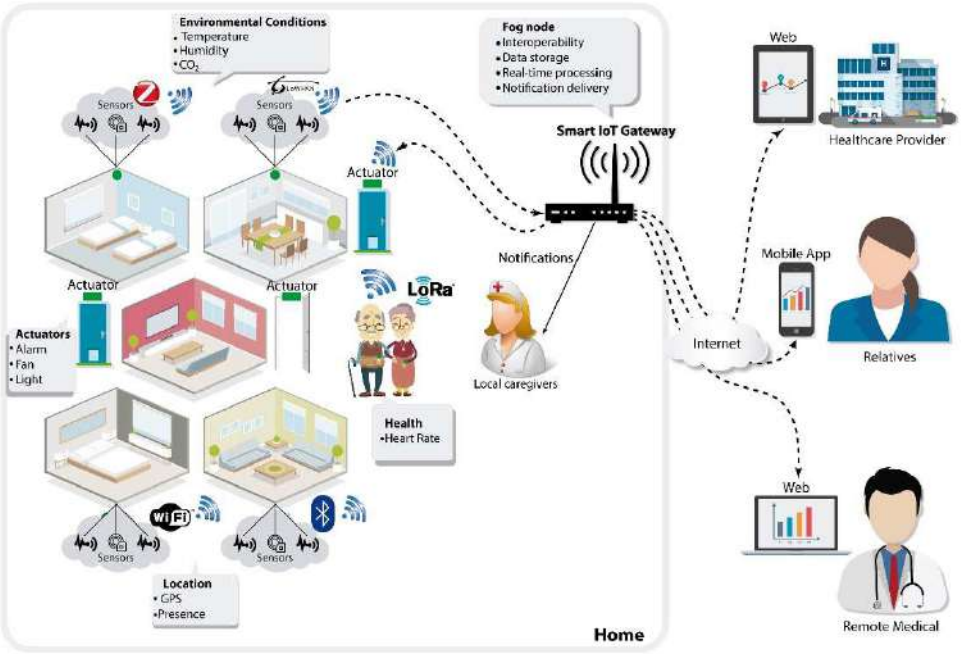


Figura 4.2: Arquitectura del caso de estudio (AAL-IoTSys)

- Red de sensores: es responsable de recopilar los datos de salud, localización de los adultos mayores y condiciones ambientales del hogar y transmitirlos al *smart IoT gateway* para su procesamiento.

- *Smart IoT gateway* que actúa como un puente de interconexión entre las WSNs y los usuarios finales, y es el componente principal donde se ejecuta toda funcionalidad del sistema. Está habilitado para proporcionar las operaciones básicas de conversión de protocolos, almacenamiento de datos, procesamiento y notificación de eventos.

- GUI (*Graphical user interface*) permite la visualización de los datos y la recepción de las notificaciones del sistema a través de una aplicación web, y aplicación móvil, respectivamente. Está diseñada para proporcionar un acceso rápido y fácil al sistema.

Esta arquitectura puede usarse en cualquier ambiente interno (hogares u hospitales). En tales sistemas, la información relacionada con la salud y el ambiente subyacente de las personas mayores se registra mediante sensores colocados en el cuerpo y distribuidos estratégicamente en el ambiente interno. Estos datos de salud también pueden complementarse con información de contexto (p. ej., fecha, hora ubicación). Esta información permite hacer inferencias más precisas sobre la monitorización. También varios actuadores pueden conectarse al sistema para actuar en consecuencia ante una situación, así como también dispositivos externos (de los responsables del cuidado de la salud de los adultos mayores) para recibir información y notificaciones asociada a los diferentes parámetros monitorizados.

4.5 Implementación del caso de estudio

La implementación del sistema AAL-IoTSys basada en la versión *smart IoT gateway* se divide tres fases principales: Implementación de las redes de sensores inalámbricas (WSNs), implementación del *smart IoT gateway* propiamente dicho e implementación de la GUI. El *smart IoT gateway* se comunica con los dispositivos de IoT a través de las WSNs, y con los responsables del cuidado de la salud de las personas mayores a través de Internet vía la GUI.

4.5.1 Redes de sensores inalámbricas

Para demostrar la interoperabilidad técnica, hemos implementado diferentes redes, utilizando varias tecnologías de comunicación heterogéneas inalámbricas, incluyendo: Wi-Fi, ZigBee, 6LowPAN y LoRa, para que cada dispositivo IoT en cada red utilice una placa de desarrollo (microcontrolador) diferente, pero trabaje en armonía con el *smart IoT gateway*. Sin ser las únicas tecnologías disponibles para IoT se ha intentado plasmar las redes WSN más populares y actuales. Cada

red utiliza el adaptador respectivo en el *smart IoT gateway* para interoperar con el resto del sistema. Los dispositivos IoT de las diferentes WSNs están configurados para monitorizar y transmitir datos de:

- Salud: con sensores de ritmo cardíaco integrados en el cuerpo de los adultos mayores se puede determinar factores de riesgo común de las enfermedades cardiovasculares como frecuencias cardíacas altas.
- Condiciones ambientales del entorno: utilizando tres puntos de medición temperatura, humedad y CO_2, se puede determinar la salud y confort del entorno en el hogar de los adultos mayores.
- Ubicación de los adultos mayores: con sensores de presencia instalados en el hogar de las personas adultas se puede determinar la presencia del adulto mayor en diferentes zonas del hogar para permitir un control de iluminación sobre dichas zonas. Si el usuario se encuentra al aire libre, su posición puede derivarse del uso de un sensor GPS instalado en un dispositivo portátil.

Además de los sensores ya descritos, los dispositivos IoT en AAL-IoTSys integran actuadores para alertar a los adultos mayores dentro del entorno monitorizado en caso que se produzca un evento inusual.

La Tabla 4.2, resume los componentes de hardware de los dispositivos IoT utilizados que están compuestos por diferentes sensores/actuadores, placas de desarrollo y módulos de comunicación. Estos dispositivos se muestran gráficamente en la Figura 4.3.

Tabla 4.2: Dispositivos IoT utilizados en el caso de estudio (AAL-IoTSys)

Parámetro	Dispositivos IoT		
	Placa de desarrollo (Microcontrolador)	Sensores/Actuadores	Módulo de comunicación
Frecuencia cardíaca	NodeMCU ESP8266 (Clase 2)	Sensor de Pulso	Wi-Fi 802.11b /g/n
Temperatura/humedad	NUCLEO L152RE+ X-NUCLEO IDS01A5 + X-NUCLEO-IKS01A1 (Clase 2)	Sensor Ambiental(HTS221)	6LowPAN-CoAP (equipado con sub-1GHz RF)

CO_2	Arduino (Clase 2)	Sensor de gas (MQ135)	ZigBee
GPS	Dragino 915 MHz + Arduino Uno (Clase 2)	Sensor de posición (NEO-7M EE-PROM)	LoRa
Presencia	Microcontrolador Arduino Nano V3.0 (Clase 2)	Sensor de Movimiento PIR (HC-SR501)	BLE
Alarma	NodeMCU ESP8266 (Clase 2)	Actuador (*buzzer Alarm*)	Wi-Fi 802.11b /g/n
Smart Light	NUCLEO L152RE+ X-NUCLEO IDS01A5 + X-NUCLEO-IKS01A1 (Clase 2)	Actuador (*Light*)	6LowPAN-CoAP (equipado con sub-1GHz RF)
Fan		SONOFF Switch	Wi-Fi 802.11b /g/n

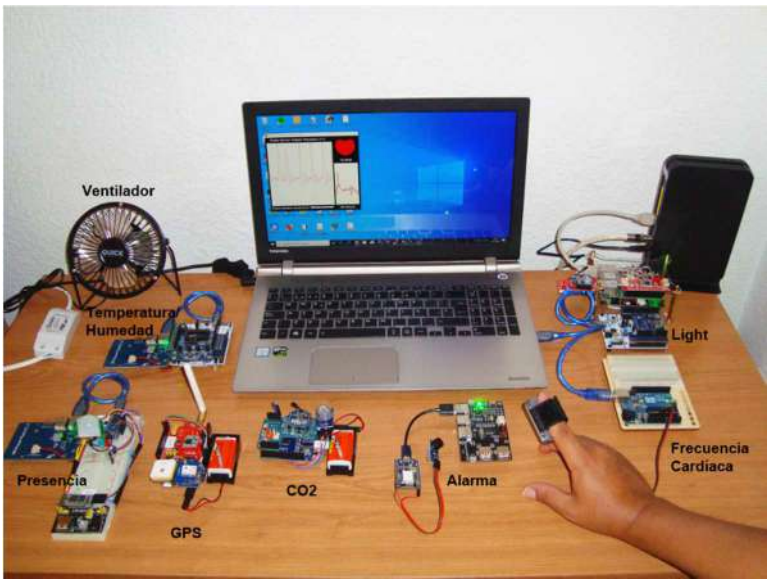


Figura 4.3: Dispositivos IoT utilizados en el prototipo AAL

4.5.2 Smart IoT gateway

En esta sección se detallan los componentes *hardware* y *software* para la implementación del *smart IoT gateway*. En particular, se describe la implementación de la plataforma de desarrollo y del lenguaje de programación utilizado para ejecutar el *smart IoT gateway*, así como cada uno de sus componentes.

4.5.2.1 Plataforma de desarrollo

Como se muestra en la Figura 4.4, inicialmente se utilizó una Raspberry Pi 2 modelo B (RPi2) (que corresponde a un dispositivo clase 2) para desplegar el *smart IoT gateway*. *RPI2* es un ordenador en formato reducido de una sola placa de bajo consumo y de bajo costo basado en el sistema en chip (SoC) Broadcom BCM2836 que incluye un procesador ARM Quad-Core de 900 MHz y 1 gigabyte de RAM (Kula, 2014). RPi2 puede admitir la integración de componentes, tales como: chips de memoria, adaptadores, interfaces de red y memorias externas.

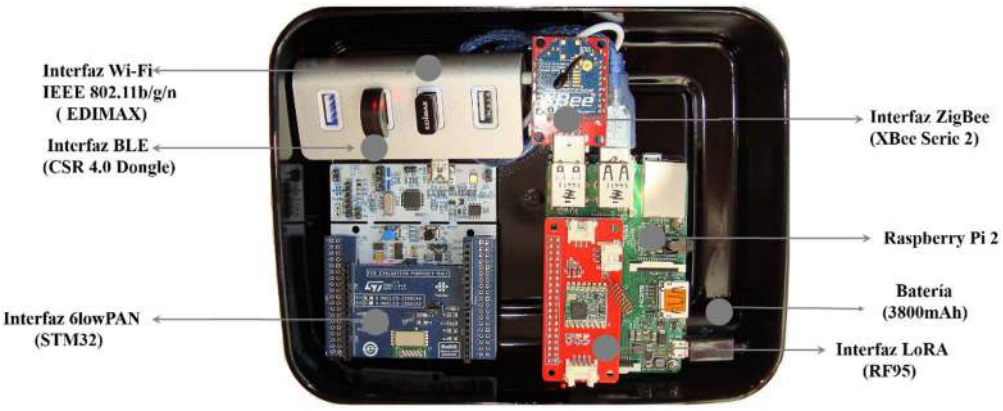


Figura 4.4: Plataforma de desarrollo del *smart IoT gateway* e interfaces integradas

En nuestra implementación, se agregó 32 GB de memoria externa que ejecuta el sistema operativo *Raspbian*. Este sistema operativo permite controlar dispositivos y ejecutar servicios como el almacenamiento, procesamiento de datos y entrega de notificaciones. RPi2 admite la interfaz de red Ethernet 10/100 Mbps. RPi2 ha sido elegido porque es una plataforma de hardware de código abierto y permite la posibilidad de agregar componentes que permiten la escalabilidad del *smart IoT gateway*, por lo tanto, del sistema IoT.

4.5.2.2 Lenguaje de programación

Una encuesta realizada por la fundación Eclipse revela que el escoger adecuadamente el lenguaje de programación es vital para el éxito del desarrollo de un proyecto IoT. De manera clásica, existen dos cuestiones fundamentales a la hora de elegir el lenguaje de programación: el usuario final y el dispositivo con el que se trabaja. El primero marca la experiencia final que se quiere conseguir. Mientras que el segundo acota las posibilidades técnicas con las que contamos. En esta tesis, enfatizando en los dos puntos, para el desarrollo de la funcionalidad del *smart IoT gateway* se ha decidió utilizar Python; un lenguaje de programación de alto nivel, interpretado, interactivo, orientado a objetos y de propósito general (Ainsley, 2018). Python surgió a principios de los 90s como un proyecto de software libre, el cual fue inicialmente desarrollado por Guido Van Rossum y actualmente está gestionado por la *Python Software Foundation*.

Python ofrece muchas ventajas(Rossum & Team, 2018), como las citadas, razón por la cual ha sido escogido en esta tesis para el desarrollo del *smart IoT gateway*:

- Python es liviano y muy potente para controlar cualquier elemento inmerso en el IoT, lo que lo sitúa en un lugar privilegiado dentro de este contexto.

- Python debido a su sencillez y baja demanda de recursos puede ser utilizado como lenguaje de programación en dispositivos con recursos limitados, de hecho, es el lenguaje de programación preferido por uno de los microcontroladores más populares del mercado, la Raspberry Pi.

- Python cuenta con estructuras de datos de alto nivel eficientes y con un enfoque simple pero efectivo para la programación orientada a objetos.

- Su sintaxis y escritura dinámica, junto con su naturaleza interpretada, lo convierten en el lenguaje ideal para el desarrollo de aplicaciones en muchas áreas y en la mayoría de plataformas.

- El intérprete de Python y las extensas bibliotecas estándar están disponibles gratuitamente en formato binario o fuente para todas las plataformas principales.

- Python permite hacer ejecutables muchas librerías hechas en el lenguaje de programación C.

4.5.2.3 Adaptador de dispositivos

Con respecto a los adaptadores, se construyeron cinco adaptadores utilizando varias interfaces de entrada y salida para establecer la comunicación del *smart IoT gateway* con cada uno de los dispositivos IoT de las diferentes redes inalámbricas.

4.5.2.3.1 Adaptador 6LowPAN

Este adaptador se construye mediante la combinación del NUCLEO-L152RE basada en un microcontrolador de potencia ultra baja y una placa X-NUCLEO IDS01A5 operando a 915 MHz conectado a la RPI2 a través del puerto USB. El NUCLEO-L152RE se basa en el procesador Cortex-M3 de 32 MHz con memoria Flash ECC de 512 KB, 80 KB de RAM y EEPROM ECC de 16KB. Como se muestra en la Figura 4.4, el núcleo está conectada a la placa X-NUCLEO IDS01A5 para recopilar los datos de la red 6LoWPAN y luego enviar los datos a RPI2 a través del puerto mini USB. Para habilitar la comunicación y permitir el intercambio de datos entre los dispositivos 6LoWPAN y el *smart IoT gateway* (es decir, el intercambio de paquetes entre los protocolos 6LowPAN e IPv6/IPV4), el controlador ha sido configurado como enrutador de borde (6LBR) en el sistema operativo *Contiki*. Este sistema operativo es de código abierto, ligero y flexible diseñado específicamente para dispositivos con recursos limitados (Dunkels *et al.*, 2004). 6LBR utiliza el protocolo RPL para enrutar y reenviar la información dentro de la red 6LowPAN. Para el descubrimiento de los dispositivos IoT el controlador (6LBR) admite los dos enfoques de descubrimiento de dispositivos proporcionados por la pila de protocolos 6LoWPAN. El primer enfoque prevé que el 6LBR envía mensajes de anuncio periódico denominados RA (del inglés *Router Advertisements*) a los dispositivos; un dispositivo se une a la red una vez que ha recibido un RA. De esta manera, el controlador puede saber cuántos nodos hay en la red. El segundo enfoque prevé que cada dispositivo en la red envía solicitudes RS (del inglés *Router solicitations*) al 6LBR. La respuesta del 6LBR es un RA (del inglés *Router Advertisement*) con la información de red (p.ej., el prefijo IPv6, técnica de compresión adoptada, la opción de enrutador de borde). Cada vez que se elimina o agrega un nuevo dispositivo se modifica la topología de la red. Con base a la información de la topología, el controlador mantiene y expone la lista de todas las direcciones IPv6 de los dispositivos conectados a la red 6lowPAN. Esta lista se proporciona a través de un servidor web incorporado en el 6LBR.

Por su parte, el *bridge* 6lowPAN, obtiene la lista de todos los dispositivos IoT de la red al consultar al servidor HTTP del 6LBR a través de un túnel SLIP. Este

túnel de red virtual (Tun0) ha sido configurado en el *bridge* 6lowPAN (mediante la herramienta *tunslip6*) para establecer la conexión con el 6LoBR a través de una conexión serial.

Un servidor CoAP (*Erbium*) basado en REST ha sido implementado en el dispositivo IoT para exponer los recursos de los sensores y actuadores al adaptador de protocolo 6LowPAN. Todas las actividades en el dispositivo IoT 6LowPAN también se ejecutan en el sistema operativo *Contiki*. Adicionalmente, en el *bridge* 6lowPAN se ha implementado un cliente CoAP para recuperar los valores de dichos recursos utilizando la biblioteca *aiocoap* nativa de Python. Como se mencionó en el capítulo 2, CoAP proporciona una alternativa ligera a HTTP para entornos y nodos restringidos, por lo que es útil dentro de redes caracterizadas por recursos limitados. CoAP permite utilizar un subconjunto de los métodos HTTP (GET, PUT, POST y DELETE) para permitir la interacción entre los puntos finales.

Una vez que se conoce la lista de dispositivos 6lowPAN, el siguiente procedimiento de descubrimiento de recursos, está listo para ejecutarse:

- **Paso 1**: El cliente CoAP del *bridge* 6lowPAN que se ejecuta en el RPi2, envía una solicitud CoAP GET al recurso CoAP /.well-known/core del dispositivo IoT.

- **Paso 2**: El dispositivo 6LoWPAN maneja la solicitud GET y responde con la lista de recursos y servicios CoAP mantenidos (en formato CoRE Link) al cliente CoAP.

```
</.well-known/core>;
ct=40,

</sensors/temperature>;
title="Temperature";
rt="st-temp",

</sensors/humidity>;
title="Humidity";
rt="st-rh",

</actuators/leds>;
title="LEDs: ?color=r|g|b,
POST/PUT mode=on|off";
rt="Control"
…
```

Figura 4.5: Lista de recursos obtenida al ejecutar la petición CoAP GET al recurso CoAP /.well-known/core del dispositivo 6LoWPAN.

La lista devuelta (Figura 4.5) muestra que el servidor tiene, entre otros, dos recursos *sensors* llamados /sensors/temperature y /sensors/humidity que devolvería la temperatura en grados Celsius y la humedad en porcentaje, respectivamente.

- **Paso 3:** Al recibir la lista de recursos CoAP el *bridge* 6lowPAN registra los dispositivos en el *smart IoT gateway*, este registro se almacenada en una base de datos.

A partir de aquí, la red 6LoWPAN está operativa y el *smart IoT gateway* a través del adaptador 6LoWPAN puede enviar más solicitudes GET al dispositivo IoT 6LoWPAN para obtener el valor de un recurso determinado.

En la Figura 4.6 se muestra un ejemplo de petición GET CoAP para obtener los datos de temperatura recopilados por el sensor de ambiente HTS221.

```
…
@asyncio.coroutine
…
def dataRetrieve(URI):
while True:
      protocol = yield from Context.create_client_context()
      request = Message(code=GET)
      request.set_request_uri(URI)
      try:
         response = yield from protocol.request(request).response
      except Exception as e:
         print('Failed to fetch resource:')
         print(e)
      else:
         data=response.payload
…
```

Figura 4.6: Ejemplo de petición GET CoAP para obtener los datos del sensor de temperatura

Similar al adaptador 6LowPAN, se han construido y configurado varios adaptadores de dispositivo para conectarse con los dispositivos IoT de las redes ZigBee.

4.5.2.3.2 Adaptador ZigBee

Para establecer la conectividad con los dispositivos de la red *ZigBee*, utilizamos un adaptador formado por la combinación de un módulo de comunicación RF (radio frecuencia) *XBee serie 2* basado en el estándar IEEE 802.15.4 y un *XBee Explorer Dongle*, que hace de puente entre el puerto serie y USB. El controlador

ZigBee cumple las funciones de coordinador de la red ZigBee, es decir es responsable de formar la red ZigBee mediante el estableciendo del canal de comunicaciones CH (frecuencia utilizada para comunicarse) entre los dispositivos ZigBee y el *smart IoT gateway* a través del PAN ID (identificador de red) que estable la pertenencia de los dispositivos a una misma red. Una vez formada la red, el controlador hace las funciones de *router*, es decir mantienen información sobre la red para retrasmitir los paquetes desde y hasta los dispositivos IoT. El controlador descubre a los dispositivos IoT ZigBee utilizando el servicio de descubrimiento de dispositivo admitido por la pila ZigBee. El servicio de descubrimiento realiza peticiones de sondeo tipo *broadcast* a cada dispositivo, los dispositivos IoT responden a las peticiones de descubrimiento enviando su propia dirección. El proceso del servicio de descubrimiento en ZigBee es la clave para interconectar dispositivos dentro de una red. Por su parte, el *bridge* ZigBee obtiene la dirección de los dispositivos proporcionada por el controlador y los registra en el *smart IoT gateway*. Una vez realizado el registro, el *bridge* ZigBee adquiere los datos del sensor de gas (MQ135) enviado por el dispositivo IoT ZigBee a través de la lectura del puerto serie específico de la RPi2 al que se encuentra conectado el controlador ZigBee.

4.5.2.3.3 Adaptador LoRa

Para establecer la conectividad con los dispositivos de la red LoRa, el adaptador LoRA se crea mediante la configuración del módulo de comunicación Elecrow Lora RFM 95 conectado directamente a la RPI2 a través de los pines GPIO. El adaptador fue configurado y programado como servidor en el lenguaje C++ utilizando la librería bcm2835. El ejecutable de este servidor se creó y ejecutó en *Python*. Se ha implementado una red LoRa privada, es decir el *smart IoT gateway* no envía los datos al servidor de red loRa a través del Internet, debido a que los datos del sistema *AAL-IoTSys* requieren ser procesados locamente.

En la red privada, el controlador actúa con un modem (Figura 4.7) estableciendo la conexión RF con los dispositivos LoRa a través de los siguientes 3 parámetros: el factor de propagación SF (del inglés *Spread Factor*), CR (del inglés *coding rate*) y ancho de banda BW (del inglés *bandwith*).

De este modo la conexión por radio frecuencia queda establecida y el controlador está listo para escuchar los mensajes entrantes. Se decidió utilizar la configuración SF=128 CR=4/5 y BW=500kHz debido a que permitió cubrir el área de interés del presente prototipo (1km) con buenos niveles de cobertura. Sin em-

bargo, de acuerdo a los requerimientos del dominio de aplicación estos parámetros pueden variar. Por ejemplo, si se requiere aumentar el área de cobertura se debe minimizar el BW y maximizar el SF. Pero si el requerimiento más importante es la confiabilidad se debe maximizar el CR.

```
…

if (!rf95.init()) {
    fprintf( stderr, "\nRF95 module init failed, Please verify wiring/module\n" );
  } else {

    if (!rf95.setModemConfig(RH_RF95::Bw500Cr45Sf128)) {
     printf("Invalid setModemConfig() option");
    } else{
    rf95.setTxPower(5, false);
    printf("RF95 radio initialized.");
    }
    rf95.setFrequency(RF_FREQUENCY);
    rf95.setModeRx();
    printf( "Listening packet...\n" );
…
```

Figura 4.7: Opciones de inicio para establecer a comunicación RF en la red LoRa implementada

Una vez que la conexión se ha establecido, el *bridge* LoRa configurado como receptor de paquetes, está listo para escuchar los mensajes entrantes. En cada trasmisión se envía la dirección del dispositivo y el nombre del recurso que proporciona el mensaje. El *bridge* LoRa registra al dispositivo y queda a la espera de nuevos mensajes entrantes del sensor de posición (NEO-7M EE-PROM).

4.5.2.3.4 Adaptador Wi-Fi

Para establecer la conectividad con los dispositivos de la red Wi-Fi, el adaptador se construye mediante la configuración del módulo mini USB 2.0 Wi-Fi 802.11 b/g/n en la RPI2. Su función es similar a la del controlador 6LowPAN es decir actúa como coordinador de toda la red, en este caso, el controlador establece la conexión con los dispositivos Wi-Fi a través de los parámetros típicos de configuración de una red Wi-Fi (p. ej., SSID, usuario y contraseña). Para el descubrimiento de los dispositivos el controlador detecta y escanea los dispositivos conectados a la red Wi-Fi mediante el envío de *beacons*, una vez que se obtiene la lista de los dispositivos se accede a los recursos de estos dispositivos. Un servidor CoAP basado en REST ha sido implementado en el dispositivo Wi-Fi para exponer los recursos de los sensores al cliente CoAP que representa al *bridge* Wi-Fi. Por lo tanto, el *bridge* Wi-Fi registra al dispositivo IoT de la red Wi-Fi y a

partir de ahí, puede recuperar los datos de la frecuencia cardíaca del sensor de pulso al enviar una solicitud CoAP GET al recurso del dispositivo Wi-Fi correspondiente.

4.5.2.3.5 Adaptador Bluetooth

Finalmente, al igual que el adaptador Wi-Fi, para establecer la conectividad con los dispositivos de la red Bluetooth, el adaptador Bluetooth se construye mediante la configuración del módulo BLE USB dongle (WiLink 6.0) conectado a la RPI2, en este caso, el controlador actúa como maestro encargado de crear la piconet BLE. Para ello, el controlador BLE ejecuta un proceso de descubrimiento de esclavos (dispositivos BLE) en la proximidad de la red-mediante la transmisión de paquetes de consulta *broadcast* a los dispositivos BLE hasta obtener respuesta por parte del o los dispositivos que desean ser descubiertos. Posterior a ello, el controlador ejecuta el proceso de emparejamiento con los dispositivos descubiertos y remite esta información al *bridge* BLE quién a su vez registra estos dispositivos en el *smart IoT gateway*. El adaptador implementa la pila de protocolos BLE gracias al *kernel-space bluez* (BLueZ Project, 2016), configurado en la RPI2.

Los adaptadores remiten la información de sus dispositivos subyacentes al *administrador de datos* a través del *message broker*; donde las funcionalidades implementadas del *smart IoT gateway*, tales como transformación de datos, almacenamiento local de datos, procesamiento de datos y entrega de eventos son ejecutados.

4.5.2.4 Message broker

El *smart IoT gateway* puede manejar la ingesta de datos provenientes de varios dispositivos, por lo tanto, las capacidades de intermediación en puntos específicos de la arquitectura son de vital importancia. El *message broker* gestiona los mensajes y es el punto de entrada al módulo de *administración de datos,* en él, los diferentes adaptadores y demás módulos funcionales implementados en el *smart IoT gateway* actúan como publicadores y/o suscriptores. Siguiendo la filosofía de evitar utilizar dependencias de software propietario, para la implementación del *message broker* se ha utilizado el proyecto de código abierto *Mosquito v3.1* debido a que es una implementación liviana que incluye la implementación del protocolo MQTT para la transferencia de datos, que como se ha mencionado en el capítulo 2, es un protocolo confiable, ligero y simple que haciendo relación a las necesidades de esta arquitectura colabora y justifica la

decisión de usarlo como protocolo de comunicación; proporciona capacidades de entrega asíncrona de mensajes para redes con recursos restringidos (bajo ancho de banda y alta latencia, conexiones frágiles, etc.) soportando diferentes niveles de calidad de servicio (QoS0-QoS2) que pueden aplicarse a varios sistemas de acuerdo a los requerimientos de cada uno y con una sobrecarga de protocolo mínima. Además, permite el desacoplamiento entre productores y consumidores (cada productor o consumidor se puede agregar o eliminar de forma independiente) lo que conlleva a una mayor escalabilidad del sistema (comunicación uno-a-uno o uno-a-muchos).

Adicionalmente, su madurez, facilidad de implementación y adopción probada lo ha convertido en la elección por excelencia para proyectos IoT como, por ejemplo, google Pub/Sub[1], WhatsApp, Line, entre otros.

Los parámetros de configuración definidos para el *message broker* se han ajustado de acuerdo a las características del sistema. Sin embargo, los mismos pueden ser parametrizados dependiendo del dominio de aplicación de la arquitectura. Con el objetivo de establecer una comunicación cifrada mediante SSL se ha definido como puerto de escucha el 8883. Durante la sesión SSL el *message broker* y los publicadores o subscriptores intercambiarán mensajes que se cifrarán simétricamente con la clave secreta compartida garantizando que los datos transmitidos no sean alterados.

Para publicar los datos en bruto (*raw data*) provenientes de los sensores de los dispositivos IoT en un tópico específico, así como para recibir estos datos, fue necesario la implementación de clientes MQTT, que se ejecuta en cada uno de los publicadores o subscriptores. La biblioteca *paho-mqtt* de *Python* se utilizó para la programación de los clientes MQTT.

Para facilitar la comunicación interna dentro del *smart IoT gateway*, como se mencionó en la sección 3.3.2.2, cada tópico está constituido por la estructura jerárquica /IdDispositivo/Recurso como se muestra en la Figura 4.8. Así, el cliente MQTT incorporado en cada uno de los *adaptadores* de los dispositivos publica los datos en bruto al *message broker*, incluyendo el tópico respectivo como clave de ruta.

[1] https://cloud.google.com/blog/products/gcp/guest-post-building-iot-applications-with-mqtt-and-google-cloud-pubsubtions

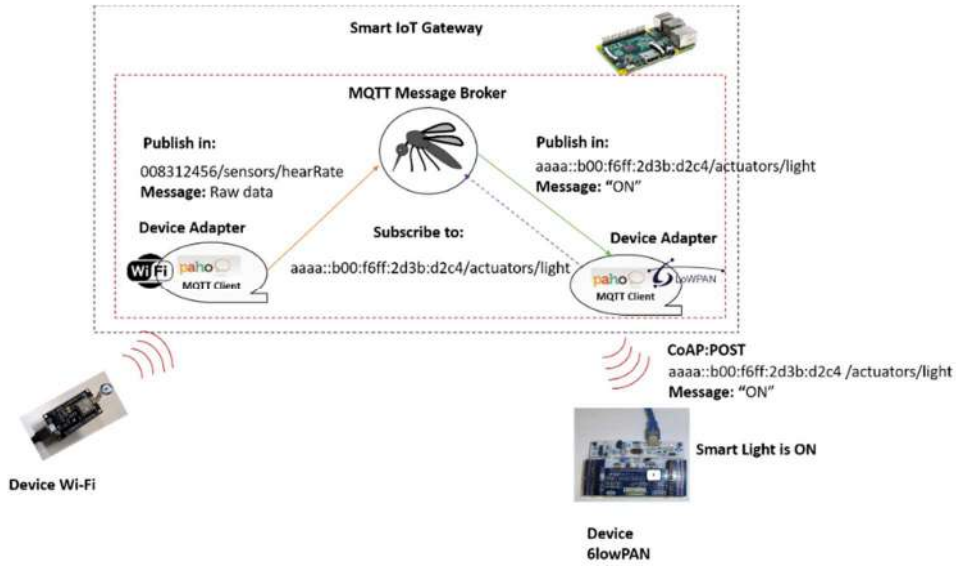


Figura 4.8: Ejemplo de la publicación del *raw data* del sensor *heartRate* por parte del adaptador Wi-Fi al Message *Broker*

```
…
import time
import paho.mqtt.client as mqtt
import ssl
…
client=mqtt.Client()
client.on_message=on_message
client.on_connect=on_connect
print("connecting to Message broker")
MB_CERT=config.get('Messagebroker', 'CERT')
MB_HOST=config.get('Messagebroker', 'HOST')
MB_QoS=config.get('Messagebroker', 'QoS')
MB_PORT=config.get('Messagebroker', 'port')
client.tls_set(MB_CERT, tls_version=ssl.PROTOCOL_TLSv1_2)
print("Sending 0....")
topic=read_topic()
data= read_data()
print("Sending 0....")
client.publish(topic, str(data), qos= MB_QoS)
time.sleep(10)
….
```

Figura 4.9: Ejemplo de publicación de datos desde un adaptador al *Message Broker*.

El código en la Figura 4.9, muestra como el cliente MQTT incorporado en el adaptador 6lowPAN se conecta al *message broker* especificando su rol e inicia

el proceso en segundo plano. Luego publica los datos en bruto indicando además del tópico, el nivel de QoS. En este *testbed*, el nivel más rápido de calidad de servicio (QoS0) ha sido definido para la publicación de mensajes, debido a que el sistema al estar enfocado a la salud de los adultos mayores es sensible a la latencia. Por lo tanto, la entrega de los mensajes debe ser rápida y en el menor tiempo posible, lo que es fundamental en este tipo de escenarios. Además, el cliente MQTT, también define el protocolo de seguridad utilizado para garantizar la seguridad en la entrega de mensajes; el protocolo TLSv1_2 ha sido empleado.

4.5.2.5 Transformación de datos

Para apoyar la interoperabilidad sintáctica optamos por utilizar un estándar de mensajería general que permite estructurar los mensajes recibidos desde los adaptadores en un formato común que sea entendido por cada componente del sistema y que garantice una transición fluida de los mensajes en la arquitectura. JSON ha sido seleccionado como el estándar de mensajería debido a que aporta ventajas a la arquitectura, como las que se describen a continuación.

- JSON proporciona una forma estándar de representación de datos y un formato flexible para hacer frente a los requisitos cambiantes de los diferentes formatos de datos nativos de los dispositivos. Puede representar cualquier estructura de datos pudiendo añadir nuevos campos con total facilidad.
- JSON es un protocolo ligero diseñado para intercambiar información estructurada independiente de cualquier lenguaje de programación, por lo que puede ser usado para el intercambio de información entre distintas tecnologías.

En la Figura 4.10 se puede observar el fragmento de datos del sensor de temperatura y localización sin procesar.

DeviceID:aaaa:b00:f6ff:2d3b:d2c4|deviceType:STM32|protocol:6LowPAN|TimeInstant:2018-08-12T12:00:04|sensor:temperature|data:**0x0921.9**|unit:ºC

DeviceID:00:1C:41:12:3C:B9|deviceType:RF95|protocol:LoRA|TimeInstant:2018-09-12T10:10:08|sensor:location|data:**MzkuNDc=/LTAuMzQ2MQ**==|unit:decimal coordinates

Figura 4.10: Ejemplo de datos sin procesar *(raw data*) de los sensores de temperatura y localización (GPS)

Los datos contienen información adicional, porque los adaptadores agregan información de contexto (marca de tiempo, protocolo, tipo de dispositivo) a los datos sin procesar enviados por los sensores. Los datos en bruto se encuentran resaltados y representan al *payload* del mensaje. La mayoría de datos enviados por los adaptadores (p.ej., identificador de dispositivo, tipo de dispositivo, protocolo, marca del tiempo) se pueden convertir sin procesamiento adicional. Sin embargo, los diferentes sensores envían los datos en diferentes formatos, por ejemplo, el sensor GPS envía los datos en formato hexadecimal mientras que el sensor de temperatura envía los datos en un formato nativo del fabricante que utiliza caracteres ascii, concretamente en base 64.

Para llevar a cabo la transformación de datos el *smart IoT gateway* activa un proceso bash (script Linux), el cual permite: (1) separar cada elemento del mensaje enviado por los adaptadores; (2) verificar si los datos del campo *data* pueden ser procesados; *(3)* decodificar el campo en caso de no poder procesarse y (4) generar la estructura en formato JSON.


```
{
"DeviceID": "aaaa:b00:f6ff:2d3b:d2c4",
 "DeviceType": "STM32",
 "Sensor": [
  {
   "temperature": {
    "value": "21.9",
    "unit": "ºC",
    "metadata": [
     {
      "protocol": {
       "value": "6LowPAN",
       "type": "string"
      },
      "TimeInstant": {
       "value": "2018-08-12T12:00:04",
       "type": "datetime"
      }
     }
    ]
   }
  }
 ]
}
```

```
{
 "DeviceID": "00:1C:41:12:3C:B9",
 "DeviceType": "RF95",
 "Sensor": [
  {
   "location": {
    "value": {
     "latitude": "39.47",
     "logitude": "-0.346173"
    },
    "unit": "decimal coordinates",
    "metadata": [
     {
      "protocol": {
       "value": "LoRA",
       "type": "string"
      },
      "TimeInstant": {
       "value": "2018-09-12T10:10:08",
       "type": "datetime"
      }
     }
    ]
   }
  }
 ]
}
```


Figura 4.11: Ejemplo de la representación de los datos de los sensores de temperatura (izquierda) y localización (derecha)

La Figura 4.11, representa la salida JSON estructurada para el ejemplo del sensor de temperatura y localización, que contiene la información de tipo y valor. Esta información se utiliza más adelante para almacenarse o procesarse.

4.5.2.6 Procesamiento de datos

Tras el proceso de transformación, los datos transformados se envían al módulo de *procesamiento de datos*, además de almacenarlos en el repositorio local; detallado más adelante.

Para la implementación del procesamiento de datos se ha utilizado un procesador de eventos complejo (CEP, por sus siglas en inglés *Complex event Process*). Más específicamente el CEP usado es Cepheus-cep (FIWARE, 2017); un motor de procesamiento de datos basado en Esper (EsperTech, 2018) y diseñado para el procesamiento de eventos complejos en tiempo real. La decisión ha sido motivada por las siguientes características:

- Cepheus-cep es una solución ligera open source que puede implementarse en dispositivos *fog computing* con recursos limitados para responder con baja latencia.
- Cepheus-cep junto con el lenguaje de procesamiento de eventos Esper proporciona gran capacidad de escalabilidad y un uso eficiente de la memoria.
- Cepheus-cep para la definición de las acciones utiliza el estándar SQL y para procesar la información utiliza una estructura óptima de datos basada en JSON que es el formato de datos admitido en el sistema.

La Tabla 4.3, muestra un ejemplo de reglas simples que se han aplicado al flujo de datos entrante para la detección de eventos en el presente *testbed*.

Tabla 4.3: Reglas configuradas en el CEP para la detección de eventos.

Clave	Operador	Valor	Acción
Temperature	>	20º C	Fan(comands/ ON) Caregivers (notification/High Temperature)
Humidity	>	50%	Alarm (commands/ON)

			Health professional and caregivers (notification/ High Relative humidity)
HeartRate	<	60 BPM	Alarm (commands/ON) Health professional and caregivers (notification/Slow HeartRate)
	>	100 BPM	Light (comands/ON) Health professional and caregivers (notification/High HeartRate)
CO_2	>	1500 PPM	Health professional and caregivers (notification/High $C0_2$)
Presence	=	1	Light (comands/ON)
GPS	>	1km	Alarm (commands/ON) caregivers (notification/location outside the coverage range)

La Figura 4.12, presenta una configuración de la acción para la regla *HeartRateAlert* configurada utilizando el lenguaje ESPER para que el CEP genere un evento de notificación cuando el ritmo cardiaco informado por el sensor *HeartRate* es menor a 60 latidos por minuto en una ventana de tiempo de 5 min.

```
{
…
"statements": ["INSERT INTO HeartRateAlert SELECT action FROM sensors.win:time(5 min) where clave="heartRate" and valor < "60"
]
}
```

Figura 4.12: Ejemplo de la configuración de una regla en el CEP

Si los datos recibidos cumplen con las reglas, el *procesador de datos* genera los eventos correspondientes y los notifica al *message broker*, que a su vez los notificará al *manejador de eventos*. Por ejemplo, si el valor proveniente del sensor *HeartRate* es menor a 60 latidos por minuto, la alarma instalada en el ambiente inteligente se activa y una notificación es enviada a los responsables del cuidado de la salud del adulto mayor para informar de la ocurrencia del evento. Aquí nos centramos en el procesamiento de datos basado en umbrales utilizando solo un operador. No obstante, como se ha explicado las capacidades del procesamiento de datos permite crear reglas complejas compuestas de varias funciones y operadores de acuerdo a los requisitos del sistema que se esté desarrollando.

4.5.2.7 Manejo de eventos

De manera similar al *message broker*, para la implementación del manejo de eventos se ha utilizado Mosquitto MQTT como *broker* y la biblioteca *paho-mqtt* de Python para la creación de los clientes (Publicadores y suscriptores).

El uso e intercambio de datos (es decir la interoperabilidad entre dispositivos heterogéneos e interoperabilidad sintáctica) se alcanza mediante el flujo de comunicación entre sensores y actuadores, que involucra todos y cada uno de los procesos anteriormente descritos hasta el envío de las acciones de control generados por el CEP a los respectivos actuadores como se indica a continuación:

Los actuadores interesados a través de sus respectivos adaptadores envían solicitudes de subscripción al *message Broker* para que se les notifique cuando el CEP genere una nueva alerta correspondiente a su dominio. Más específicamente, en este escenario concreto.

(i) El actuador *Alarm* envía una solicitud de subscripción al adaptador Wi-Fi, para que se les notifique cuando el CEP genere una nueva alerta del ritmo cardiaco para responder en consecuencia (activando la alarma). En este caso, el Adaptador Wi-Fi debe haber enviado una subscripción similar al *message broker* por adelantado, ya que el *message broker* es responsable del flujo de comunicación interna. Este procedimiento es similar para el actuador *Fan*.

(ii) De manera similar el adaptador 6LowPAN se subscribe previamente al *message broker*, para que se le notifique cuando el CEP genere una nueva alerta ante la presencia del adulto mayor en el área de cobertura del sensor PIR. En este caso, el adaptador 6LowPAN envía el comando de control del evento (encender la luz) al servidor CoAP implementado en el

dispositivo 6lowPAN (actuador *Smart Light)* a través de una solicitud POST

(iii) Los médicos y el personal responsable del cuidado de la salud de los adultos mayores también envían una solicitud de subscripción, en este caso, directamente al *manejador de eventos* (ya que es responsable de comunicar los eventos a los usuarios externos), para que se les notifique cuando el CEP genere cualquier alerta a fin de responder en consecuencia y asistir a tiempo a los adultos en caso que lo requieran.

De manera similar al *message Broker*, el *manejador de eventos* está configurados para publicar los eventos con QoS-0 porque un escenario AAL se caracteriza por ser sensible al tiempo. Al configurarlos con QoS-1 o QoS2 se puede afectar la latencia y el ancho de banda de la red.

4.5.2.8 Almacenamiento de datos

Hemos implementado un repositorio local (ver Figura 4.13), en el *smart IoT gateway* utilizando el sistema de gestión de bases de datos MySQL Esta base de datos ha sido seleccionada debido a sus múltiples ventajas presentadas para el sistema; es libre, puede ser instalada en múltiples plataformas, tiene soporte técnico ampliamente disponible, soporta un sistema seguro de autorización y permite la ejecución de operaciones de manera rápida equiparada a cualquier base de datos no propietaria gracias a su agilidad de consultas y transacciones.

La base de datos MySQL permite almacenar:

- Los datos de los sensores, así como el estado de los actuadores.
- Las notificaciones y alertas generadas por el CEP.
- Los datos temporales producto de la transformación, procesamiento y análisis de datos.

Para gestionar dicha base de datos y la información almacenada se utilizó el software *phpMyAdmin*. La Figura 4.13 muestra un ejemplo de los registros de los datos insertados en la tabla sensores.

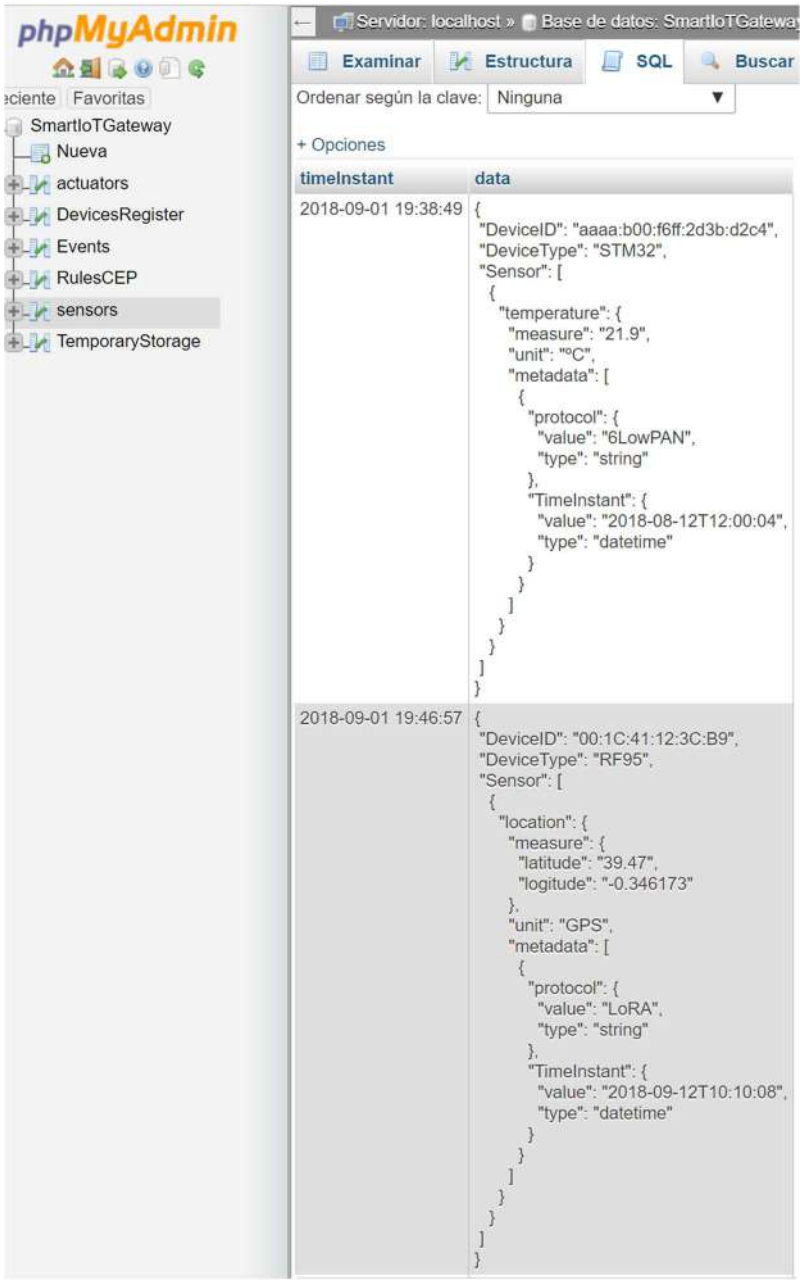


Figura 4.13: Ejemplo de datos de los sensores almacenados en la base de datos local del *smart IoT gateway*

4.5.2.9 Interfaz gráfica de usuario (GUI)

La interfaz gráfica de usuario es responsable de convertir la información procesada en contenido enriquecido y mostrarla. En el sistema, hay dos GUI disponibles: la aplicación web y la aplicación móvil.

- **Aplicación Web**: permite a los cuidadores rastrear la ubicación de las personas mayores a su cuidado y ver las medidas de todos los parámetros monitorizados desde sus propios dispositivos móviles o desde cualquier ordenador. La *aplicación web* se creó con el lenguaje *JavaScript*, el cual permite crear aplicaciones flexibles y fáciles de usar a las que cualquier persona conectada a Internet puede acceder. Además este lenguaje es multiplataforma, tiende a ejecutar las funciones inmediatamente mejorando la escalabilidad y el rendimiento de las aplicaciones web, es soportado por los navegadores más populares y compatible con dispositivos modernos incluyendo iPhone, móviles etc. (Sotiropoulos & Livshits, 2019).

  El *tracking* de la localización de las personas mayores y el informe de los datos monitorizados se desarrollaron utilizando las herramientas de software gratuitas: *OpenStreetMap y HighCharts* respectivamente. La ubicación de las personas mayores se representa en un mapa que muestra las medidas actuales de los parámetros monitorizados: frecuencia cardíaca, temperatura, humedad, CO2, como se muestra en la Figura 4.14 (a).

- ***Mobile App***: Hemos desarrollado una aplicación móvil que se comunica con el *smart IoT gateway* a través de Wi-Fi. La aplicación móvil desarrollada para el sistema operativo Android contiene una *clase ReceiveNotifications()* que ejecuta un cliente MQTT app para permitir a los usuarios finales suscribirse a las notificaciones generadas por el CEP. Si existe una notificación *Mobile App* despliega en la pantalla de los usuarios la información correspondiente al evento generado, así como también información de contexto como la hora y fecha de ocurrencia del evento.

  Así los profesionales de la salud y los cuidadores de los adultos mayores estarán informados en todo momento y en tiempo real sobre las situaciones inusuales detectadas en los parámetros de monitorización. Para ilustrar la recepción de notificaciones, la Figura 4.14 (b) muestra algunas alertas enviadas por el *smart IoT gateway* y desplegadas por la aplicación Android.

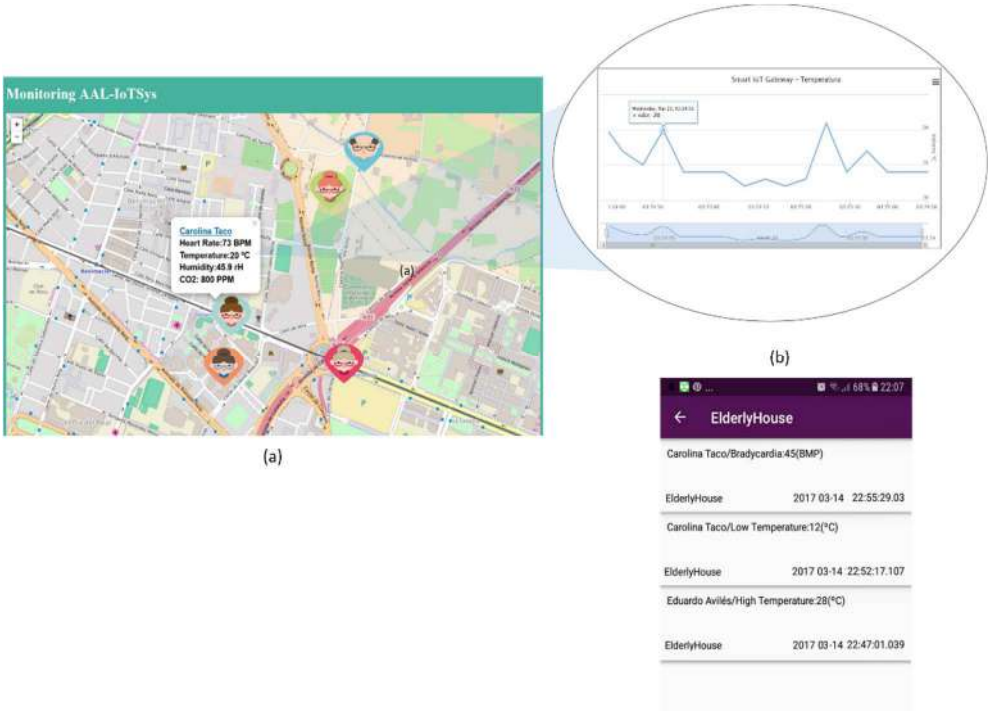


Figura 4.14: Aplicación Web para la monitorización de las personas mayores

4.6 Evaluación y resultados

En esta sección describimos el *testbed* implementado para evaluar el rendimiento del *smart IoT gateway*. En particular, el *testbed* en esta primera evaluación se realizó utilizando la RPI2 con capacidades de hardware y software descritas en la sección 4.5.2.1, así como la infraestructura de hardware tanto a nivel de dispositivos heterogéneos sensores como de actuadores descrita previamente en la Tabla 4.2. Vale la pena señalar que la RPI2 usada no representa la varguardia en el mercado de los ordenadores de placa reducida Raspberry Pi, por lo que puede considerarse representativa de una amplia gama de dispositivos con recursos limitados. El banco de pruebas implementado tiene como objetivo investigar el potencial de los escenarios de comunicación heterogéneos en los que el *smart IoT gateway*, instalado en dispositivos con recursos limitados, representa la interfaz transparente entre los dispositivos IoT y las aplicaciones IoT. Una vez que se ha diseñado el sistema AAL para soportar todos los estándares de comunicación dentro

del escenario presentado, mostramos el rendimiento del *smart IoT gateway* en términos de uso de memoria (RAM), CPU y consumo de energía. Este análisis muestra cómo se utilizan los recursos de hardware del *smart IoT gateway* en el escenario de IoT presentado.

4.6.1 Escenarios de prueba

A lo largo del banco de pruebas realizado, el *smart IoT gateway* recopila y reenvía datos recibidos de los diferentes sensores heterogéneos a través de diferentes interfaces de comunicación (consulte la Tabla 4.4) durante varios periodos de duración de 600 segundos.

Tabla 4.4: Dispositivos IoT conectados al *smart IoT gateway* a través de varias interfaces de comunicación

Escenario	Interfaces				
	ZigBee	BLE	Wi-Fi	6LowPAN	LoRA
Carga baja	1 Sensor de gas	1 Sensor de Movimiento	1 Sensor de Pulso 1 Actuador (Led) 1 SONOFF switch	1 Sensor de temperatura 1Actuador (*light*)	1 Sensor de Posición
Carga alta	2 Sensor de gas	2 Sensor de Movimiento	2 Sensor de Pulso 2 Actuador (Led) 1 SONOFF switch 3 Actuadores(Dispositivos móviles)	2 Sensor de temperatura 2 Actuador (*light*)	2 Sensor de Posición

Para asegurarse de que las condiciones de red no afecten el rendimiento drásticamente del *smart IoT gateway,* se ha realizado una evaluación exhaustiva de 3 semanas, promediando las medidas obtenidas al enviar datos desde los diferentes sensores al *smart IoT gateway* hasta la entrega de comandos de control a los actuadores y notificaciones a los dispositivos móviles de los usuarios finales como

resultado de la detección de eventos. Más específicamente, las operaciones llevadas a cabo en esta evaluación involucran a las funciones: conversión de protocolos, transformación de datos, almacenamiento de datos, procesamiento de datos y notificación de eventos. En particular, se realizaron 15 pruebas, repitiendo 10 veces cada una. Se ha establecido un nivel de confianza de 0.95 excluyendo los primeros 60s del cálculo del error estadístico (es decir el periodo transitorio) a fin de verificar la exactitud del análisis estadístico para los resultados obtenidos.

De acuerdo con este enfoque, todos los resultados presentados satisfacen el nivel de confianza elegido después del primer minuto de prueba.

Para la evaluación se han considerado dos escenarios de carga de trabajo: alta y baja. La carga de tráfico baja se genera mediante 5 sensores diferentes que transmiten constantemente datos al *smart IoT gateway* y 3 dispositivos actuadores que reciben desde el *smart IoT gateway* comandos de control a través de diferentes interfaces. Por el contrario, el escenario de carga alta representa un número superior de sensores (10 en lugar de 5) y de actuadores (8 en lugar de 3); en este último caso se admite la entrega de notificaciones a 3 dispositivos móviles actuando también como actuadores.

4.6.2 Uso de CPU

La Figura 4.15, muestra que el *smart IoT gateway* tiene una carga de CPU promedio reducida de alrededor del 24% en un escenario de carga baja incrementándose su utilización en 14% en un escenario de carga alta; de hecho, es posible apreciar la diferencia entre el escenario de comunicación de carga baja y alta. Esta diferencia es un claro indicativo de la versatilidad del *smart IoT gateway* para afrontar los diferentes casos y dominios de aplicación donde se pueda desplegar. El máximo valor de CPU alcanzado en condiciones de carga pesada es de aproximadamente el 52%, mientras que el valor promedio es de alrededor del 38%. Con base en los resultados reportados se puede deducir que el aumento de sensores enviando datos y actuadores recibiendo comandos de control, así como también dispositivos móviles finales recibiendo notificaciones no produce un aumento lineal en el consumo de la CPU.

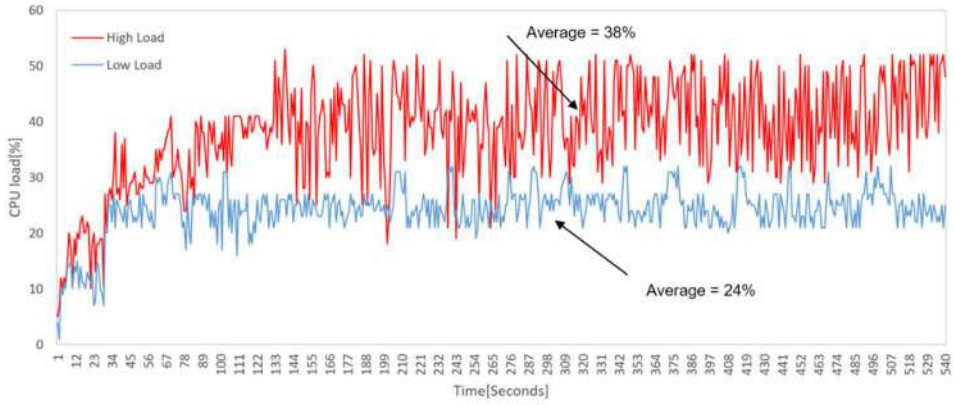


Figura 4.15: CPU promedio del Smart IoT Gateway en los escenarios de carga baja y alta evaluados

4.6.3 Uso de memoria RAM

El análisis del uso de la memoria RAM define si el *smart IoT gateway* tiene que lidiar con el uso intensivo de memoria y cómo el escenario de carga alta afecta el rendimiento de la misma. El uso de memoria representa un aspecto relevante lo que sugiere que el *smart IoT gateway* podría alojar más aplicaciones de uso intensivo de memoria, destinadas a funcionalidades específicas requeridas por dominios de aplicación y no cubiertas actualmente por el *smart IoT gateway*, por ejemplo, la compresión de datos. La Figura 4.16, ilustra los resultados del consumo porcentual de memoria RAM al ejecutar las funcionalidades del gateway.

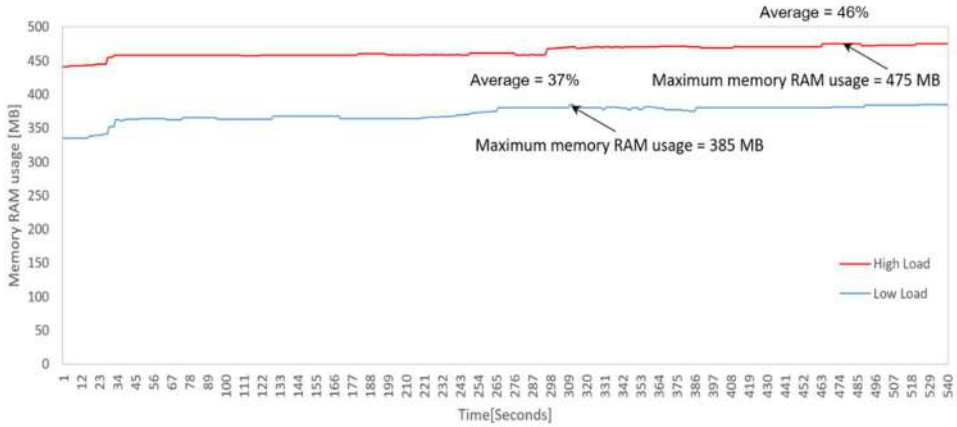


Figura 4.16: Uso de la memoria RAM en los escenarios de carga baja y alta evaluados

Es posible observar, que en el escenario de carga alta la cantidad máxima de memoria utilizada es baja (475 Mbytes) y no supera el 50% de utilización del total de memoria (1GB). Mientras que en un escenario de carga baja se requiere una cantidad máxima de memoria de aproximadamente 385 MBytes para ejecutar dicha funcionalidad. A diferencia de lo que se encontró durante el análisis de la CPU, existe una diferencia mínima (9%) entre el escenario de carga baja y alta; afirmando nuevamente el rango de casos a los que puede atender la implementación del *smart IoT gateway*. Adicionalmente, en los dos escenarios evaluados existe una tendencia plana para toda la duración de las pruebas.

Teniendo en cuenta los resultados de CPU y Memoria en su conjunto, podemos afirmar que la amplia gama de servicios proporcionada por el *smart IoT gateway* (conversión de protocolos, transformación de datos, almacenamiento de datos, procesamiento de datos y manejo de eventos) puede ser desplegada en dispositivos con recursos limitados con un nivel eficiente de adaptabilidad. Así se presenta como una solución viable para el desarrollo de soluciones IoT donde la interoperabilidad es explotada.

4.6.4 Consumo de energía

Finalmente, parte de la evaluación del *smart IoT gateway* cubre un aspecto relacionado con los requisitos del consumo de energía. En particular, para el suministro de energía se empleó una batería de litio (*Hub Power Supply*) de 3800mAh a 5.0 V. Además, se instaló en la RPI2 el sensor de corriente INA219 (Adafruit Industries, 2019), con el objetivo de evaluar el nivel de consumo promedio que demanda la ejecución de la funcionalidad del *smart IoT gateway*. Se desarrolló un *script* en Python que proporciona el acceso al dispositivo INA210 utilizando la librería *Adafruit CircuitPython* (Adafruit Industries, 2018). A fin de hacer evidente el descenso de la batería en los dos escenarios de carga de trabajo, se realizó una evaluación durante un periodo de análisis de 30 minutos de duración. Para ello se configuró un intervalo de 1 segundo entre cada medición. La Figura 4.17, muestra los resultados en términos porcentuales obtenidos durante todo el periodo de análisis. Se puede desprender que se experimenta una reducción de energía aproximada de 12% en escenarios de carga baja a casi el 21% en escenarios de carga alta (Figura 4.17). En particular, el menor nivel de consumo que se obtiene en el escenario de carga baja es de 11% con una demanda promedio 418 mA. En el caso del escenario de carga alta el menor consumo de corriente alcanza el 19% con una demanda promedio de consumo de corriente de 722 mA, lo que representa un incremento del 8% en relación al mínimo obtenido en el

escenario de carga baja (418mA). En cuanto a los incrementos del consumo, los resultados en los dos escenarios describen un comportamiento aproximadamente lineal. Con base en este análisis la vida útil de la RPi2 en los diferentes escenarios de carga varía de aproximadamente 2,5 a más de 4.5 h considerando la batería empleada en el prototipo (3800mAh). Cabe indicar que en la mayoría de los escenarios de aplicación la placa de desarrollo utilizada tiene conexión eléctrica continua.

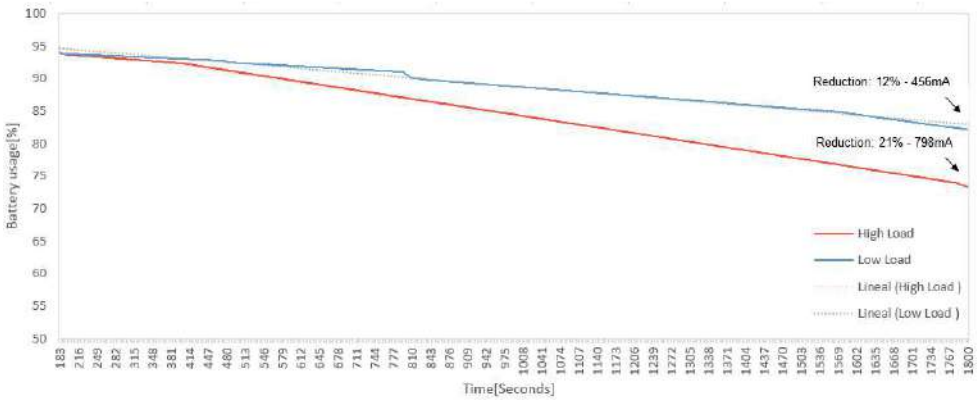


Figura 4.17: Consumo de la batería en los escenarios de carga baja y alta evaluados

4.7 Trabajos relacionados

Un conjunto de capacidades mínimas que deben incluir un *gateway* en IoT se encuentran en (ITU-T, 2018). En particular, un *gateway* tiene que actuar como *proxy*, que interconecta los sensores y aplicaciones.

Las capacidades descritas son: (i) reenvío de mensajes entre diferentes dispositivos que estén conectados al *gateway*, esto implica abordar la interoperabilidad entre dispositivos a través de la conversión de protocolos; (ii) capacidad de procesamiento local, que se refiere a las necesidades de agregación de datos desde dispositivos, análisis de datos, transformación de datos entre diferentes formatos según lo requieran los dispositivos y aplicaciones, y generación de metadatos relacionados con el dispositivo según corresponda y; (iii) capacidad de descubrimiento de dispositivos, necesarias para descubrir los recursos de los dispositivos, recopilar información y controlar los dispositivos. El *smart IoT gateway* es capaz de cumplir con todas las propiedades mencionadas anteriormente.

Existen diferentes trabajos relacionados con el diseño e implementación de *gateways* para IoT. Para cada una de estas soluciones, consideramos las siguientes características claves, que abarcan las propiedades mencionadas anteriormente:

- Interoperabilidad: Muestra la capacidad del *gateway* para interactuar con tecnologías de comunicación múltiple (ZigBee, BLE, LoRa, Wi-Fi, 6LowPAN, etc.) y para permitir una notación sintáctica de los datos de los sensores.
- Soporte de protocolos IoT optimizados: Muestra la capacidad del *gateway* para permitir la transferencia de datos preservando los recursos restringidos de los dispositivos IoT.
- Intercambio de información bidireccional: Muestra la capacidad del *gateway* para la recopilación y difusión de datos y el control de dispositivos.
- Funcionalidades: Muestra la capacidad del *gateway* para llevar a cabo funciones de procesamiento y almacenamiento de datos, y manejo de notificaciones, etc.
- Priorización en la entrega de notificaciones: Muestra la capacidad del *gateway* para priorizar la entrega de notificaciones a los usuarios finales.
- Ejecución local de aplicaciones IoT: Muestra la capacidad del *gateway* para permitir que las aplicaciones se ejecuten localmente.

Un primer conjunto de investigaciones propone el diseño e implementación de *gateways* para solucionar la interoperabilidad entre protocolos y tecnologías de comunicación específicos.

- Zhu *et al.*, (2010) proponen un "IoT Gateway" que proporciona la funcionalidad de la red troncal entre las redes de sensores y las aplicaciones a través de Internet. Más en detalle, el *IoT gateway* se caracteriza por permitir la conversión entre protocolos IEEE 802.15.4 / Zigbee y GPRS con el objetivo de facilitar la transmisión de datos de las redes de sensores inalámbricas y las redes de comunicación móviles. Las propiedades de interoperabilidad son conceptos similares a nuestra propuesta, sin embargo, nuestra propuesta admite tecnologías de comunicación múltiple mediante el uso de diferentes estándares e interfaces de comunicación (WiFi, ZigBee, 6LowPAN, BLE, LoRA, etc). Así como también el

acceso a dispositivos heterogéneos a través de la adaptación de protocolos IoT de capa aplicación como CoAP y MQTT. Además, aunque los autores proporcionan una descripción completa de la arquitectura, faltan por completo los detalles sobre el *software* utilizado para la construcción del prototipo del IoT gateway. Adicionalmente una evaluación del rendimiento está ausente.

- Guoqiang *et al.*, (2013) proponen un *smart IoT gateway* que presenta tres ventajas importantes: puede comunicarse con diferentes redes de comunicación (Ethernet, 3G / LTE o bus RS485) que operan con diferentes protocolos, tienen interfaces externas unificadas (SPI, UART, USB, Ethernet (RJ45) y tiene un protocolo flexible para traducir diferentes datos de los sensores a un formato uniforme. Las capacidades que ofrece este *gateway:* conversión de protocolos y transformación de datos presentan un buen punto de partida para abordar la interoperabilidad. Sin embargo, resulta limitado el número de interfaces y tecnologías de comunicación admitidas por el mismo. Adicionalmente, no incluye las capacidades de procesamiento local y descubrimiento de dispositivos. Además, la implementación de mecanismos de QoS en la transmisión de datos, también en este caso, quedan para futuras implementaciones. (Guoqiang *et al.*, 2013)

- Una investigación similar es propuesta por Bimschas *et al.*,(2010) a través de la implementación de un *middleware*, que habilita en el *gateway* la ejecución de código de aplicaciones para ofrecer las funcionalidades de: conversión de protocolos, almacenamiento en caché y descubrimiento de dispositivos. Los autores a fin de preservar los recursos limitados de las redes de sensores proponen usar el protocolo CoAP como protocolo de aplicación para la transferencia eficiente de datos entre las redes de sensores y el gateway, en lugar de utilizar protocolos estandarizados, pero menos eficientes en IoT como por ejemplo el protocolo HTTP. Esta arquitectura satisface parcialmente el requisito de soporte de protocolos IoT optimizados para las comunicaciones M2M, al explotar los beneficios del protocolo CoAP. Sin embargo, resulta limitado en el número de redes admitidas y la capacidad de procesamiento. Además, aunque los autores proporcionan evaluaciones de rendimiento, faltan por completo los detalles sobre el software y hardware utilizado para el prototipo.

Otras contribuciones proponen *gateways* que funcionan solo para dominios de aplicación específicos. Por ejemplo,

- A. Rahmani *et al.*, (2015) describen un Gateway denominado "Smart e-Salud" que actúa como un puente para sensores médicos y dispositivos de automatización de edificios de hogares para redes basadas en IP y plataformas de computación en la nube. Esta arquitectura satisface la mayoría de los requisitos que cumple el *smart IoT gateway,* sin embargo, gracias al uso de protocolos eficientes en la comunicación M2M entre los dispositivos finales y el *smart IoT gateway* en nuestra propuesta, podemos mejorar el rendimiento de las redes inalámbricas a la vez que se preserva los recursos limitados de los dispositivos IoT, que en muchos casos para satisfacer la movilidad pueden estar alimentados por baterías externas utilizando recursos mínimos para conservar la energía. Adicionalmente, el *Smart e-Salud* no soporta tecnologías emergentes como LoRA. La evaluación del rendimiento y capacidad del *Smart e-Salud* no se realiza y los autores no incluyen ningún mecanismo de QoS en la entrega de notificaciones. Los autores afirman que "la arquitectura permite el descubrimiento de dispositivos". Sin embargo, no existe evidencia que demuestre esta declaración.

- Un *Smart Home Gateway* para el hogar basado en OSGI es propuesto por Kim *et al.*, (2012), el cual permite la interconexión e interoperabilidad de diferentes protocolos de red doméstica: x10, Insteon, ZigBee y UPnP. Las capacidades del descubrimiento de dispositivos, así como la transformación de datos, son conceptos similares a nuestra propuesta, sin embargo, ofrecemos capacidades que se están volviendo cruciales en los escenarios IoT, como el procesamiento y entrega de notificación. El *Smart Home Gateway* presenta una arquitectura cuya adaptabilidad en nodos con recursos limitados como los SoCs puede ser cuestionable. El *Smart Home Gateway* propuesto no incluye ningún mecanismo de seguridad y QoS para la entrega de notificaciones. Además, la utilización de protocolos de comunicación IoT y la evaluación del rendimiento de la propuesta, también en este caso, está ausente.

- Desai *et al.*, (2015) proponen un *Semantic Gateway as Service* (SGS) para proporcionar la interoperabilidad entre protocolos de mensajería XMPP, CoAP y MQTT a través de una arquitectura de *proxy* multiprotocolo. Más en detalle, el SGS se caracteriza por permitir la interoperabilidad a nivel de protocolos de aplicación. Esta arquitectura

resulta similar al *smart IoT gateway* respecto al uso de protocolos IoT livianos y seguros para la comunicación M2M entre los dispositivos IoT y el gateway. Sin embargo, SGS no provee servicios de procesamiento de datos, análisis de datos y entrega de notificaciones. Adicionalmente, la evaluación del rendimiento de SGS está ausente.

De la misma manera para la construcción de *gateways IoT* existen propuestas de código abierto como: *Kura*,(2015), *Iotivity*,(2017) y *Agile*, (2017) las cuales son impulsadas por las iniciativas M2M Eclipse.

- *Kura* es un *framework* abierto basado en Java OSGI que proporciona una plataforma para crear *gateways* IoT. Este *framework* ofrece un conjunto de APIs para simplificar la administración de las configuraciones de red (Ethernet, Wi-Fi y módems celulares), la comunicación con plataformas de integración IoT y la administración remota del *gateway*. Uno de sus principales objetivos es la integración de diferentes protocolos y estándares en un entorno heterogéneo. Sin embargo, utilizan *bundles* en Java, que limitan las capacidades del *gateway* a un solo lenguaje de programación (Rykowski & Wilusz, 2014). Adicionalmente, no admite la utilización de protocolos IoT para la transferencia de datos.

- *Iotivity*, es un *framework* de código abierto, cuyo protocolo clave para la comunicación dispositivo a dispositivo (D2D, por sus siglas en inglés) es el protocolo de aplicación restringido CoAP. *Iotivity* proporciona funciones de descubrimiento, conexión y mensajería entre ellos. Sin embargo, resulta limitado en el número de redes admitidas (Wi-Fi, Ethernet, BLE, ZigBee).

- *Agile* es una puerta de enlace modular y adaptable para dispositivos IoT. La modularidad en el nivel de hardware proporciona soporte para varias tecnologías de redes IoT inalámbricas y por cable (p.ej., KNX, ZWave, ZigBee, Bluetooth Low Energy, etc.), y permite la creación rápida de prototipos de soluciones de IoT para varios dominios (p.ej., domótica, monitorización del entorno, etc.). En el nivel de software, los diferentes componentes habilitan las funciones locales de recopilación, administración de datos y administración de dispositivos, así como también un editor de flujo trabajo visual para crear aplicaciones de IoT. Las características de interoperabilidad, así como

la arquitectura multiprotocolo, son conceptos similares a nuestra propuesta. Sin embargo, el soporte de tecnologías emergentes como LoRA y 6LowPAN falta. Adicionalmente, no admite la entrega de notificaciones ni la priorización de las mismas.

En la Tabla 4.5 proporcionamos un resumen del cumplimiento de las características evaluadas para cada *gateway* propuesto. Ofrecemos un cumplimiento de propuestas de tres niveles, donde un círculo de color negro significa que la solución propuesta cumple con la característica esperada, mientras que círculo de color blanco significan que la solución solo proporciona un cumplimiento parcial de la característica esperada. El incumplimiento de una característica está representado por la ausencia de cualquier círculo. Adicionalmente, se utiliza las siglas NA para indicar que la característica no aplica a la solución evaluada. Este situación se da en las propuestas *Kura*,(2015), *Iotivity*,(2017), *Agile*, (2017) de las cuales no se tiene evidencia en la literatura de la evaluación del rendimiento de estos *gateways*. Sin embargo, son relevantes para este análisis.

Tabla 4.5: Comparación del *smart IoT gateway* con propuestas alternativas

Gateway	Interoperabilidad		Protocolos IoT (MQTT y CoAP)	Intercambio de Información bidireccional	Funcionalidades (Procesamiento, almacenamiento, manejo de notificaciones	Priorización en la entrega de notificaciones	Ejecución local de aplicaciones IoT	Prototipo (harware/software)	Evaluación del rendimiento(CPU, memoria, consumo de energía)
	Técnica	Sintáctica							
Zhu *et al.*, (2010)	○								
Bimschas *et al.*,(2010)	○		○		○				○
Kim *et al.*, (2012)	○	●							
Guoqiang *et al.*, (2013)	○	●							
A. Rahmani *et al.*, (2015)	○	○			○	○		●	
Desai *et al.*, (2015)	○		○		○			●	
Kura,(2015)	○				○			●	NA
Iotivity,(2017)	○		○		○			●	NA
AGILE (2017)	○		○		○		○	●	NA
(Yacchirema & Palau, 2017)	●	●	●	●	●	●	●	●	●

4.8 Conclusiones

En este capítulo, una primera versión embrionaria de la arquitectura denominada *smart IoT gateway* ha sido implementada y descrita en detalle tanto a nivel de *hardware* como de *software*. El prototipo *smart IoT gateway* permite alcanzar la interoperabilidad técnica y sintáctica entre dispositivos, enfocada en dar respuesta a la heterogeneidad inherentes de las tecnologías, protocolos y formato de datos subyacentes de los dispositivos.

El *smart IoT gateway* permite recopilar y reenviar datos provenientes de dispositivos IoT que operan en diferentes redes (ZigBee, Wi-Fi, BLE, LoRa y 6LowPAN) mediante la implementación de varios adaptadores y el establecimiento de un formato de datos uniforme basado en JSON; además permite enviar mensajes de control y notificaciones a los actuadores y usuarios finales, respectivamente, como consecuencia del procesamiento de datos ejecutado con base en reglas definidas.

La funcionalidad y utilidad del *smart IoT gateway* ha sido verificada a través de un caso de estudio AAL, no obstante, puede aplicarse a varios dominios debido a que permite llevar a cabo la secuencia de acciones que se realizan a nivel de las capas de la arquitectura básica de una solución IoT (recopilación de datos, procesamiento y envío de notificaciones). El *smart IoT gateway* ha sido implementado en un dispositivo con recursos limitados, RPI2, los resultados obtenidos, a través de *testbeds* evaluados sobre escenarios de carga alta y baja, validan el prototipo construido que puede ser implementado en este tipo de dispositivos limitados sin hacer un uso excesivo de recursos de hardware en términos de CPU y memoria, así como también del consumo de energía. Siendo este último superable mediante el uso de baterías más eficientes.

Capítulo 5

Arquitectura de interconexión de dispositivos heterogéneos con plataformas IoT

«El aprendizaje es experiencia todo lo demás es información»

Albert Einstein (1879-1955)

5.1 Introducción

Son varias las implicaciones de lograr la interoperabilidad entre todos los elementos del ecosistema IoT (p.ej., plataformas, sistemas y dispositivos), siendo remarcables los beneficios económicos potenciales que se podrían alcanzar con este reto, que se espera que lleguen hasta un 60% para el 2020 (Mckinsey Global Institute, 2015). Sin embargo, alcanzar la interoperabilidad entre todos sus elementos no es sencillo. Dando respuesta a esta necesidad, varias organizaciones de desarrollo de estándares (p.ej., oneM2M, ITU), proyectos de investigación (p.ej., INTER-IoT) y consorcios industriales (p.ej., AllJoyn) han dirigido sus esfuerzos activamente hacia actividades para alcanzar la interoperabilidad en IoT a diferentes niveles (dispositivos, redes, middleware, plataformas, etc.). En particular, la iniciativa global oneM2M, es un organismo de estandarización global para comunicaciones M2M e IoT lanzado en 2012, por siete de los principales organismos de desarrollo de estándares de Europa (ETSI), Japón (ARIB y TTC), Estados Unidos (ATIS y TIA), Corea (TTA) y China (CCSA), los cuales han unido sus esfuerzos para minimizar la fragmentación actual de las soluciones propietarias presentes en el mercado IoT y en las comunicaciones M2M a través

de la definición de especificaciones de interconexión de aplicación amplia e independientes de las tecnologías de acceso de red y transmisión subyacentes (Swetina *et al.*, 2014). Estas especificaciones de interconexión buscan asegurar que los dispositivos de IoT interactúen sin problemas entre ellos y con otros elementos (p.ej., plataformas IoT) a escala global para facilitar el desarrollo de servicios y aplicaciones de IoT horizontales. En particular, oneM2M ha trabaja en la especificación de entidades de aplicación, denominadas entidades *proxy* de interconexión (IPE, por sus siglas en inglés, *Interworking proxy entity*) que permite que elementos *legacy* (es decir, no compatibles con oneM2M), en adelante NoDN, interactúen con otros que cumplen con los estándares que promueven (oneM2M, 2017).

La implementación del *smart IoT gateway* desarrollada en el marco de la presente tesis, extiende sus funcionales para permitir la interconexión entre dispositivos IoT y la plataforma oneM2M en una primera instancia, y su posterior extensibilidad para la integración con la plataforma FIWARE; para ello, integra una IPE, dando lugar a un *smart IoT gateway-IPE*.

El propósito de la *IPE* es proporcionar un mecanismo que facilite las interacciones transparentes entre dispositivos heterogéneos y las plataformas IoT integradas, con el objetivo de exponer los datos existentes de estos dispositivos para su uso en nuevos sistemas y servicios IoT. Utilizamos el *smart IoT gateway-IPE* y la plataforma de código abierto Eclipse OM2M y el *Context Broker Orion* de oneM2M y FIWARE, respectivamente, como caso de prueba para nuestro diseño de interconexión.

En capítulo describe como la arquitectura de interconexión propuesta se adecua a los requerimientos de la arquitectura global. A continuación, se realiza una revisión de los trabajos relacionados con la implementación de IPEs basados en el estándar oneM2M. Posteriormente, se detalla la arquitectura de interconexión propuesta. Luego, se lleva el fundamento a un caso práctico por medio de un caso de uso centrado en el sector de transporte y logística (INTER-LogP). Finalmente, se describe las principales conclusiones de este capítulo.

Este capítulo está basado en las publicaciones (D Yacchirema *et al.*, 2018; Diana Yacchirema; Gonzalez-Usach, *et al.*, 2018).

5.2 Relación de la arquitectura de interconexión con la arquitectura global

Similar a la arquitectura del *smart IoT gateway*, esta implementación de la arquitectura de interconexión basada en oneM2M al ser una extensión de la versión *smart IoT gateway* se adecua de la misma forma (consultar Tabla 4.1) a los requerimientos (R1-R10) de la arquitectura global. Se diferencia en que la arquitectura de interconexión además da cumplimiento al requerimiento R11 (Integración de los dispositivos con plataformas IoT estándar) que no fue abordado en la versión *smart IoT gateway*, a través de la implementación de un proxy de interconexión basado en oneM2M, que facilita la integración de dispositivos *legacy* no compatibles con las plataformas adoptadas.

Para la implementación *software* de la arquitectura se emplea un enfoque combinado basado en la computación *fog y cloud*. Los principios nativos del *cloud* se aplican para el despliegue de las plataformas IoT con las que los dispositivos se integrarán. A través de este enfoque de implementación se favorece la escalabilidad de la arquitectura, mientras se mantiene el procesamiento de los datos al borde de la red para permitir la interacción entre dispositivos. Además, en esta arquitectura se ha optado por alojar la lógica de la aplicación a nivel *cloud*

En esta arquitectura las plataformas IoT, se encargan del almacenamiento permanente de los datos y entrega de la información a las entidades externas (servicios, aplicaciones y usuarios finales) que lo requieran, así como de la seguridad y autenticación previa.

5.3 Trabajos relacionados

Apoyar la interoperabilidad de los dispositivos IoT utilizando estándares internacionales ha sido una investigación de particular relevancia en IoT. Pocos estudios han explotado las especificaciones técnicas de interconexión proporcionadas por el estándar oneM2M, para interoperar dispositivos IoT e integrarlos con plataformas estándar a través de IPEs. En este sentido, Yun *et al.*, (2016) desarrolló un IPE para la interconexión de sistemas oneM2M con productos de consumo IoT heredados de *Nest*, *Jawbone* y *Withings*. La IPE propuesta traduce los mensajes de oneM2M en el protocolo de destino de los servidores que almacenan los datos de los dispositivos en lugar de traducir directamente el mensaje de protocolo de los dispositivos IoT. Por lo tanto, esta solución requiere que los servidores web donde se alojan cada una de las aplicaciones de los dispositivos

estén operativos, de lo contrario esta solución no funciona. Los autores Wu *et al.*, (2017) describen una arquitectura de integración basada en una IPE para abordar la interconexión de plataformas específicas de IoT (es decir, plataformas AllJoyn e loTivity) con el sistema oneM2M. Esta IPE actúa como middleware, que admite la asignación de funciones de administración de dispositivos entre estas plataformas. La arquitectura se valida mediante dos casos de prueba de interconexión. Aunque estos diseños de interconexión se divulgan, no hay evidencia de que estos diseños proporcionen la comunicación bidireccional entre dispositivos. Finalmente, Kim *et al.*, (2016) demostró cómo los procedimientos de interconexión provistos por IPE podrían aplicarse en condiciones reales como las ciudades inteligentes para interconectar múltiples plataformas de servicios de IoT oneM2M. A través de las experiencias y las lecciones aprendidas, los autores destacaron que la principal ventaja de la interconexión basada en oneM2M radica en fomentar la comunicación e interacción entre dispositivos IoT, redes de dispositivos y aplicaciones basadas en el *cloud* a escala global.

5.4 OneM2M y especificaciones de interconexión

Hacia una mejor comprensión, en esta sección, presentamos brevemente el estándar oneM2M y las especificaciones de interconexión de oneM2M basadas en IPE.

5.4.1 Estándar oneM2M

La Figura 5.1, presenta el modelo de la arquitectura de referencia de oneM2M. Considerando un escenario de configuración cuando se implementan sistemas oneM2M, la arquitectura oneM2M divide los entornos M2M/ IoT en dos dominios: dominio de infraestructura y dominio de campo. El dominio de infraestructura consta de un nodo de infraestructura (IN), que se asigna físicamente al servidor de servicio M2M ubicado en el lado de la red de transmisión.

El dominio de campo consta de los nodos intermediarios (MN, por sus siglas en inglés *middle node*), nodos de servicio de Aplicación (ASN, por sus siglas en inglés *Application Service Nodes*) y nodos dedicados de aplicación (AND, por sus siglas en inglés *application dedicated Nodes*), que pueden implementarse físicamente como *gateways,* dispositivos M2M y sensores*,* respectivamente.

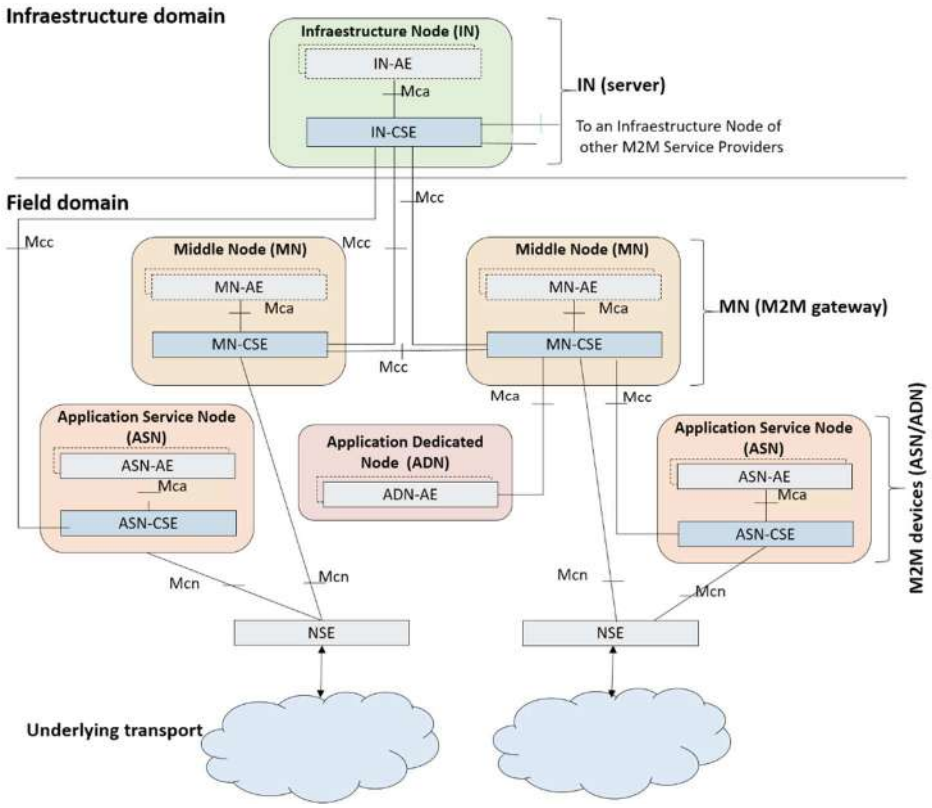


Figura 5.1: Arquitectura funcional oneM2M

Cada nodo puede estar compuesto de dos tipos de entidades lógicas implementadas como funciones de software:

- Una entidad de servicios comunes (CSE) que consta de servicios M2M comunes como administración de datos, administración de dispositivos, servicios de subscripción/notificación, entre otros. El CSE se implementa en los nodos MN, ASN y en cada IN (OneM2M, 2018). Según el tipo de nodo que aloja el CSE, éste se puede clasificar como un IN-CSE, MN-CSE o ASN-CSE.
- Una entidad de aplicación (AE) que contiene la lógica de aplicación de las soluciones IoT y a través de las cuales se accede a los datos de un sistema, por ejemplo, una aplicación para la monitorización de la calidad

- del sueño, una aplicación para la monitorización de la humedad del suelo, etc

Además, oneM2M define una entidad de servicio de red (NSE) que involucra servicios de red básicos como el transporte y la conectividad que deben utilizar los CSE.

La conexión y el intercambio de información entre estas entidades se realiza a través de puntos de referencia: Mca, Mcc y Mcn. El punto de referencia Mca expone los servicios incluidos en el CSE al nodo AE que se ejecuta en los dispositivos. El punto de referencia de Mcc permite a un CSE utilizar los servicios incluidos en otro CSE. El punto de referencia de Mcn permite al CSE utilizar los servicios admitidos por el NSE.

oneM2M adopta un modelo de información basado en recursos. Por lo tanto, los servicios y datos que soporta el sistema oneM2M son representados como recursos. Cada recurso es identificado de forma única mediante un identificador uniforme de recursos (URI), y las interacciones con los recursos son compatibles con las cuatro operaciones básicas CRUD (crear, recuperar, actualizar y eliminar). Todos los recursos de oneM2M pueden ser manipulados a través de APIs RESTful. Estos recursos se organizan en una estructura jerárquica. La dirección de acceso al recurso se representa cono una dirección jerárquica desde la raíz hasta el recurso.

La última versión de oneM2M *Release 2* se publicó en 2018. Esta versión tiene como objetivo proporcionar nuevas funcionalidades y capacidades para expandir el ecosistema de IoT. El estándar actualizado incluye seguridad mejorada, interoperabilidad semántica, características para la habilitación de dominios doméstico e industrial, e interconexión con dispositivos de conectividad IoT impulsada por la industria como *Open Mobile Alliance LWM2M, Open Connectivity Foundation (OCF) y AllSeen Alliance*. Estas actualizaciones y su estado en detalle se pueden encontrar en (oneM2M, 2018). En particular, las especificaciones de interconexión actualizadas incorporan mejoras basadas en la experiencia de implementaciones tempranas para lograr una interoperabilidad de alto nivel al respaldar la interconexión del ecosistema oneM2M con soluciones NoDN a través una IPE.

5.4.2 Interconexión a través del IPE

oneM2M define tres formas de interconectar soluciones NoDN con oneM2M a través del IPE:

1. **Interconexión con mapeo completo del modelo de datos NoDN a oneM2M:** Consiste en traducir todo el modelo de datos NoDN al modelo de datos genérico utilizado en oneM2M (basado en el uso de contenedores). En este caso, la IPE debe incluir toda la lógica de *interworking* de protocolo relacionada. Dependiendo de la complejidad del modelo de datos NoDN, puede implicar que el IPE cree un conjunto complejo de recursos (a partir de los recursos básicos oneM2M) en la CSE, o un mapeo directo simple de la comunicación a través de los contenedores.
2. **Interconexión utilizando contenedores para el transporte transparente de datos y comandos NoDN a través de Mca**: Consiste en empaquetar los datos y comandos codificados NoDN en una lista de contenedores oneM2M. En consecuencia, el CSE o AE necesitan conocer las reglas de codificación de protocolo específicas de la solución NoDN para decodificar el contenido dentro de los contenedores.
3. **Interconexión mediante un mecanismo de reasignación:** Por lo general, esto se admite a través de un *gateway* que sea capaz de mapear operaciones entre soluciones NoDN a soluciones oneM2M y viceversa.

En este trabajo, se seleccionó la primera forma como enfoque de interconexión para la implementación de la arquitectura, debido a que ofrece una solución única para permitir las comunicaciones entre diferentes protocolos. Además, el modelo de datos NoDN determina su representación (es decir, los nombres, tipos de datos y estructuras de los recursos) en el sistema oneM2M. Como consecuencia este enfoque facilita la interoperabilidad de protocolos de comunicación, el intercambio de información sintáctica, el uso y el intercambio de datos entre diferentes soluciones y despliegues.

5.5 Arquitectura de interconexión

Para apoyar la interoperabilidad de dispositivos NoDN con la plataforma oneM2M hemos extendido la funcionalidad de la implementación *smart IoT gateway* propuesta en el capítulo 4 mediante la integración de un *proxy de interconexión*. En particular, se ha implementado un IPE diseñado para incorporar la variante de *interworking* con mapeo completo del modelo de datos de los dispositivos NoDN al modelo de datos admitido por la plataforma oneM2M. La intención es

que el IPE proporcione un conjunto de funcionalidades útiles para establecer la interacción entre la plataforma oneM2M y los dispositivos heterogéneos conectados al *smart IoT gateway*.

La arquitectura de interconexión propuesta se puede observar en la Figura 5.2. Se trata de un esquema de alto nivel en el que un nodo intermedio (MN) está compuesto por la IPE y una MN-CSE. La IPE se define para la interconexión de los dispositivos NoDN con la plataforma oneM2M. El IPE está compuesto por dos componentes principales: un *bridge* de interconexión y una entidad IoT, para convertir el dispositivo NoDN en un recurso oneM2M. La IPE se registra en el MN-CSE con el objetivo de alojar el servicio de *internetworking*, que permite la sincronización de los dispositivos NoDN con las instancias de recursos que representan en la plataforma oneM2M. La MN-CSE y el IN-CSE se registran mutuamente para intercambiar la información. Un sistema, plataforma, aplicación, servicio puede acceder a los datos almacenados en el IN-CSE a través de una IN-AE. oneM2M sigue el enfoque de separar los dispositivos NoDN de los dispositivos oneM2M. En consecuencia, los dispositivos IoT conectados esta vez al *smart IoT gateway-IPE* comprenden los dispositivos NoDN y la entidad IoT se utiliza para conectarse con estos dispositivos.

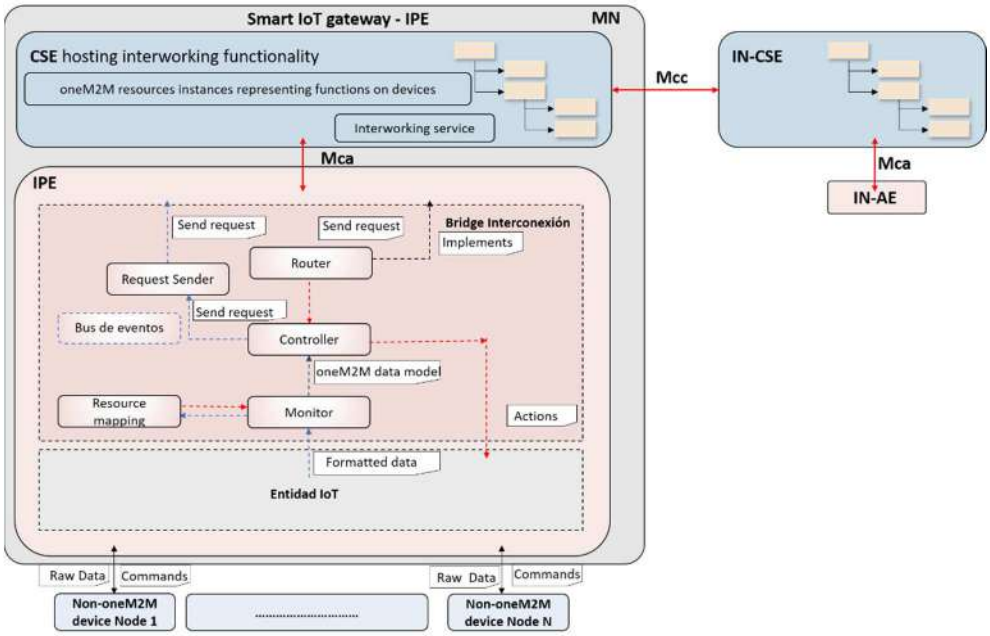


Figura 5.2: Arquitectura de *interworking* de dispositivos NoDN con oneM2M basada en IPE

Los módulos funcionales del IPE se describen brevemente a continuación:

- **Entidad IoT:** es responsable de la interoperabilidad entre dispositivos y de la gestión local de los datos de estos dispositivos. Por lo tanto, está representada por los componentes funcionales de la implementación *smart IoT gateway* (adaptador de protocolo, transformación de datos, procesamiento de datos, almacenamiento de datos, manejo de eventos y *message broker*).

- **Bridge de interconexión**: facilita el entendimiento común de los datos recopilados desde los dispositivos NoDN. Además, el *bridge de interconexión* administra el acceso y extrae el conocimiento de los dispositivos NoDN al describir las instancias de recursos en el sistema oneM2M. Su principal función es mapear el modelo de datos de los dispositivos NoDN en el árbol de recursos de oneM2M.

Como se observa en la Figura 5.2 el *bridge de interconexión* está constituido por un conjunto de sub-módulos interrelacionados que se describen a continuación:

  - Monitor: identifica y recupera los datos de cada dispositivo NoDN expuesto a la plataforma oneM2M y envía la solicitud de inserción de estos datos en la plataforma oneM2M al controlador de plataforma.

  - *message mapper:* permite transformar el formato de datos común definido en la arquitectura al formato de recursos definido en oneM2M (basado en contenedores).

  - Controlador de plataforma*:* realiza dos tareas: Por un lado, implementa las funciones necesarias para conectarse a la plataforma oneM2M (p.ej., maneja la creación de recursos por cada dispositivo). Por otro lado, cuando se envía un mensaje a la plataforma oneM2M, el controlador de plataforma compone la solicitud adecuada utilizando el modelo de datos proporcionado por el *message mapper* y lo envía al *request sender***.** De manera similar, cuando la plataforma oneM2M envía acciones de control, el controlador de plataforma utiliza esta información para enviar a la entidad IoT, quién a través del manejador de eventos y adaptador de protocolos envían las acciones de control al dispositivo IoT.

  - *Request sender:* está diseñado para crear y enviar solicitudes a la plataforma oneM2M vía el punto de acceso Mca. Además, proporciona la respuesta de estas solicitudes.

- *Enrutador:* define una ruta única para manejar todas las solicitudes dirigidas desde la plataforma al IPE en un controlador de recursos simple y envía la solicitud al método correspondiente del controlador de plataforma.

- *Bus de eventos*: admite la intercomunicación entre los componentes del IPE basada en eventos utilizando un mecanismo de publicación/subscripción. Este componente permite que los componentes internos sean informados sobre el estado actualizado. Las líneas discontinuas de color rojo y azul representan estas corrientes de comunicación.

5.6 Verificación de la arquitectura de interconexión

En este trabajo, se implementa un *testbed* con el objetivo de evaluar la utilidad y viabilidad de la arquitectura de interconexión basada en el estándar oneM2M. En particular, primero se describe un caso de uso real. Luego, se detalla los procedimientos de interconexión para el caso de uso. Posteriormente, se describe la implementación y los experimentos de interconexión de la arquitectura del *testbed*.

Con base en las experiencias y en las lecciones aprendidas de la interconexión basado en oneM2M (J. Kim *et al.*, 2016), los experimentos de interconexión para la verificación de la arquitectura incluyen:

- Procedimientos de registro de sensores y actuadores a una plataforma de servidor IoT compatible con oneM2M.
- Registro del smart *IoT gateway-IPE* (legacy) a una plataforma de servidor IoT compatible con oneM2M.
- Aplicación de transporte inteligente que reciba notificaciones sobre los dispositivos registrados e introduzca comandos de control para actuar sobre el entorno.

5.6.1 Caso de estudio: INTER-LogP

Con el fin de fomentar la cooperación de varias plataformas IoT a través de la interoperabilidad sin fisuras de dispositivos IoT heterogéneos, el caso de uso seleccionado para este *testbed* de interconexión es INTER-LogP, que se deriva del proyecto INTER-IoT.

El caso de uso de INTER-LogP (Inter-IoT, 2018) ilustra la necesidad de lograr una interoperabilidad perfecta de diferentes plataformas IoT heterogéneas, orientadas al transporte y logística portuaria en diferentes niveles o capas: dispositivo, redes, middleware, aplicaciones y servicios. En este trabajo, enfocamos el caso de uso para lograr la interoperabilidad en el nivel del dispositivo.

El alcance del INTER-LogP incluye varios escenarios. Nos centramos en el escenario de *control de acceso, tráfico y asistencia operativa*, cuyo objetivo es proporcionar servicios para controlar el acceso, supervisar el tráfico y ayudar a las operaciones en el puerto a través de la interoperabilidad de diferentes plataformas que puede dirigirse a resolver varios problemas. Un problema importante en las terminales portuarias de contenedores es el alto nivel de tráfico y congestión en la entrada de las puertas de las terminales en las horas punta. Esto se debe a la falta de coordinación entre los operadores de terminales y las empresas de transporte y se agrava por el continuo y rápido aumento de contenedores en el puerto. Por ejemplo, el agente de carga informa al transportista por carretera sobre la fecha estimada para recoger o entregar las mercancías en el contenedor, pero la terminal de contenedores generalmente no tiene conocimiento de la fecha y la hora en que el camión llega a su puerta. Esta falta de información impide una planificación óptima de las operaciones de la terminal portuaria y favorece una llegada masiva de camiones al final de los tiempos de cierre, en lugar de tener un flujo más regular durante el tiempo de operación. Por consiguiente, En las horas pico hay largas colas en la puerta de la terminal portuaria y dentro del patio de contenedores, lo que afecta la calidad, la seguridad y la sincronización de las operaciones de manejo de contenedores. Este problema de ineficiencia causa largos tiempos de espera para los transportistas en la terminal y se traduce en un menor rendimiento de las operaciones de la terminal, pérdida de tiempo y recursos económicos y más contaminación.

Para dar solución a este problema desde la interoperabilidad de dispositivos modelamos el camión como uno de los principales activos que la arquitectura monitorizará, con el objetivo de compartir información útil con las plataformas IoT portuarias y de transporte bajo ciertas reglas predefinidas. Los principales actores, y plataformas IoT involucradas en el caso de uso son:

- Terminal de contenedores: *Terminal IoT Platform* (TIP).
- Autoridad Portuaria: *Port IoT Platform* (PIP).
- Empresas de transporte: *Haulier IoT Platform* (HIP).

Como se observa en la Figura 5.3, el *testbed* de interconexión identifica varias entidades físicas de transporte (es decir camiones) donde se despliegan varios dispositivos IoT (Ver Tabla 5.1) que integran sensores y actuadores en varios lugares. Los sensores capturan varias mediciones en tiempo real e informan al *smart IoT gateway-IPE* utilizando diferentes tecnologías de comunicación. El *smart IoT gateway-IPE* aloja un nodo intermedio que a su vez integra una IPE que proporciona una interfaz con los dispositivos NoDN y una interfaz con la MN-CSE, y una MN-CSE que proporciona una interfaz con la plataforma servidor IoT oneM2M en adelante servidor oneM2M. Las lecturas de los sensores se propagan desde el *smart IoT gateway-IPE* al servidor oneM2M a través del punto de referencia Mcc. El servidor oneM2M aloja una IN-CSE que admite un conjunto de servicios para permitir que las plataformas IoT involucradas (TIP, PIP e HIP) y la aplicación inteligente *"Smart Truck"* monitoricen al camión. Algunos servicios incluyen el registro, la gestión de datos, subscripción/notificación, etc. Las lecturas de los sensores se propagan a las plataformas registradas en el servidor oneM2M utilizando los servicios de notificación/subscripción oneM2M sobre el punto de referencia Mca.

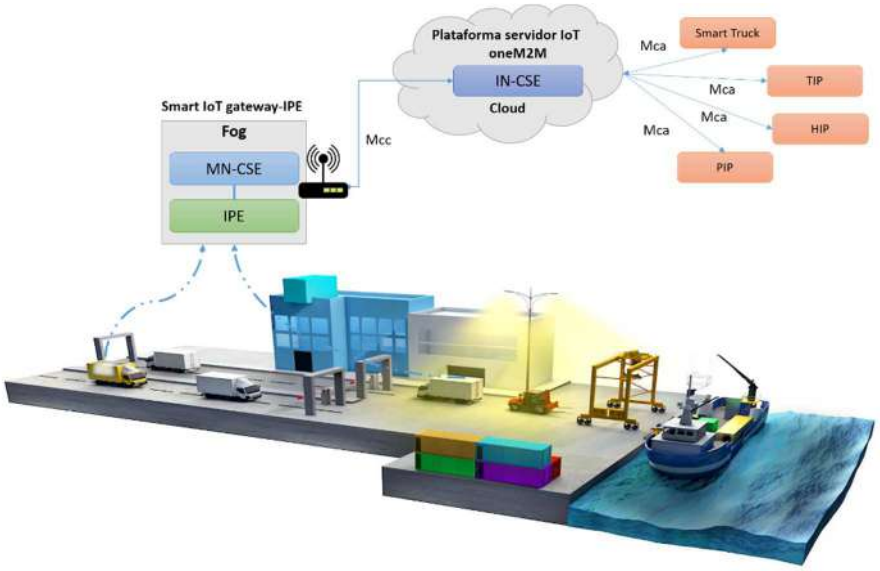


Figura 5.3: *Testbed* de interconexión

La aplicación inteligente "*Smart Truck*" y cada una de las plataformas IoT involucradas se interconecta directamente con el servidor oneM2M para recibir las lecturas de los sensores asociadas a los dispositivos NoDN expuestos por el *smart*

IoT gateway-IPE. Además, la aplicación *"Smart Truck"* controla de forma remota los actuadores en el camión.

5.6.2 Procedimientos de interconexión entre los dispositivos y la plataforma oneM2M

Como se describió en la sección anterior el caso de uso requiere que se realicen varios procedimientos como el registro, la creación de la estructura de recursos, la inserción de datos en las instancias de recursos y la subscripción/notificación. Estos procedimientos son habilitadores clave para permitir la interconexión entre los dispositivos y la plataforma oneM2M. El registro es el proceso en el que la IPE se registra en el MN-CSE que a su vez se registra en el IN-CSE, mientras que la creación de la estructura es un proceso para crear recursos en un CSE que están vinculado a los dispositivos IoT. Los procedimientos de subscripción/notificación permiten la monitorización de los cambios en las instancias de recursos.

La Figura 5.4 muestra el flujo de información de alto nivel del registro, creación de la estructura de recursos, y subscripción/notificación.

Como punto de partida, para un mejor entendimiento de los procedimientos de interconexión del *testbed*, se creó un tipo de recurso <AE> llamado "*truck*" en el *smart IoT gateway-IPE* (MN-CSE) a través del IPE, que modela al activo físico "camión" sobre la cual se crea el árbol de recursos oneM2M. Además, en esta entidad se registró dos características principales del camión: la placa del camión y el sistema de refrigerado (si/no).

- **Paso 1:** El IPE en el nodo intermedio (es decir en el *smart IoT gateway-IPE*) se registra en el MN-CSE que a su vez se registra en el IN-CSE. A través de este registro el *smart IoT gateway-IPE* y el servidor oneM2M se conocen y están listos para intercambiar la información entre sí.

- **Paso 2:** Como la aplicación inteligente (IN-AE) *"Smart Truck"* y las plataformas IoT (TIP, PIP e HIP) está interesadas en toda la información proveniente del *MN-CSE*, se subscriben a la IN-CSE para estar notificadas de cualquier actualización en las lecturas de los sensores.
- **Paso 3:** Los dispositivos IoT conectados al MN son descubiertos por el IPE a través de la entidad IoT que establece la comunicación con ellos

mediante un adaptador específico de acuerdo con la tecnología de comunicación utilizada.

- **Paso 4:** La IPE registra los sensores y actuadores de los dispositivos IoT en MN-CSE y crea un tipo de recurso <contenedor> para cada uno de ellos, a través del *bridge* de interconexión. El MN-CSE a su vez los anuncia al IN-CSE.
- **Paso 5-6:** Cada vez que los sensores envíen datos al MN, la IPE recibe estos datos a través de la entidad IoT vía el adaptador de protocolo. La entidad IoT transforma los datos recibidos al formato común definido en el sistema (es decir, JSON) a través del módulo de transformación de datos que se lo envía al *bridge* de interconexión, y a los módulos de almacenamiento y procesamiento vía el *message broker*. Si el procesamiento da como resultado una acción, la entidad IoT a través del adaptador de protocolo envía los comandos al actuador respectivo. Nótese que los actuadores también pueden ser controlados a partir de la aplicación *web* "Smart Truck".
- **Paso 7:** Como ya se ha comentado, el *bridge* de interconexión está constituido por 6 sub-módulos: un bus de eventos, un monitor, un controlador de plataforma, un *message mapper*, un *request sender* y un enrutador. El *bridge* de interconexión al recibir los datos de los sensores a través del bus de eventos, envía el mensaje al módulo monitor. El monitor recupera los datos en formato JSON y envía la solicitud de inserción de estos datos en la MN-CSE al controlador de plataforma.
- **Paso 8:** El controlador de plataforma envía los datos al módulo *message mapper* para el análisis de mensajes y la conversión de los mismos en el formato de recursos oneM2M.
- **Paso 9:** El *message mapper* extrae la información necesaria del mensaje y construye los mensajes de solicitud primitiva oneM2M correspondientes (Figura 5.5), que se entrega devuelta al módulo controlador de plataforma.
- **Paso 10:** El controlador de plataforma envía al *request sender* una solicitud de creación de un tipo de recurso < contentInstance> con los datos formateados en primitivas oneM2M.
- **Paso 11:** El *request sender* ejecuta la solicitud de inserción vía el punto de acceso Mca.

- **Paso 12:** Las lecturas de los sensores insertadas y actualizadas por el IPE dan como resultado notificaciones que se envían a las plataformas IoT suscritas (TIP, PIP e HIP) y a la aplicación "Smart Truck".
- **Paso 13:** Para establecer la comunicación bidireccional, cuando se requiere un cambio de estado en los actuadores desde la aplicación "*Smart Truck"* el IN-CSE envía a la IPE los comandos de acción.
- **Paso 14-15:** La IPE recibe las solicitudes de control vía el enrutador, el cual envía la solicitud de actualización de estado al controlador de plataforma.
- **Paso 16:** Finalmente, la IPE ejecuta la acción de control en el actuador a través del adaptador respectivo.

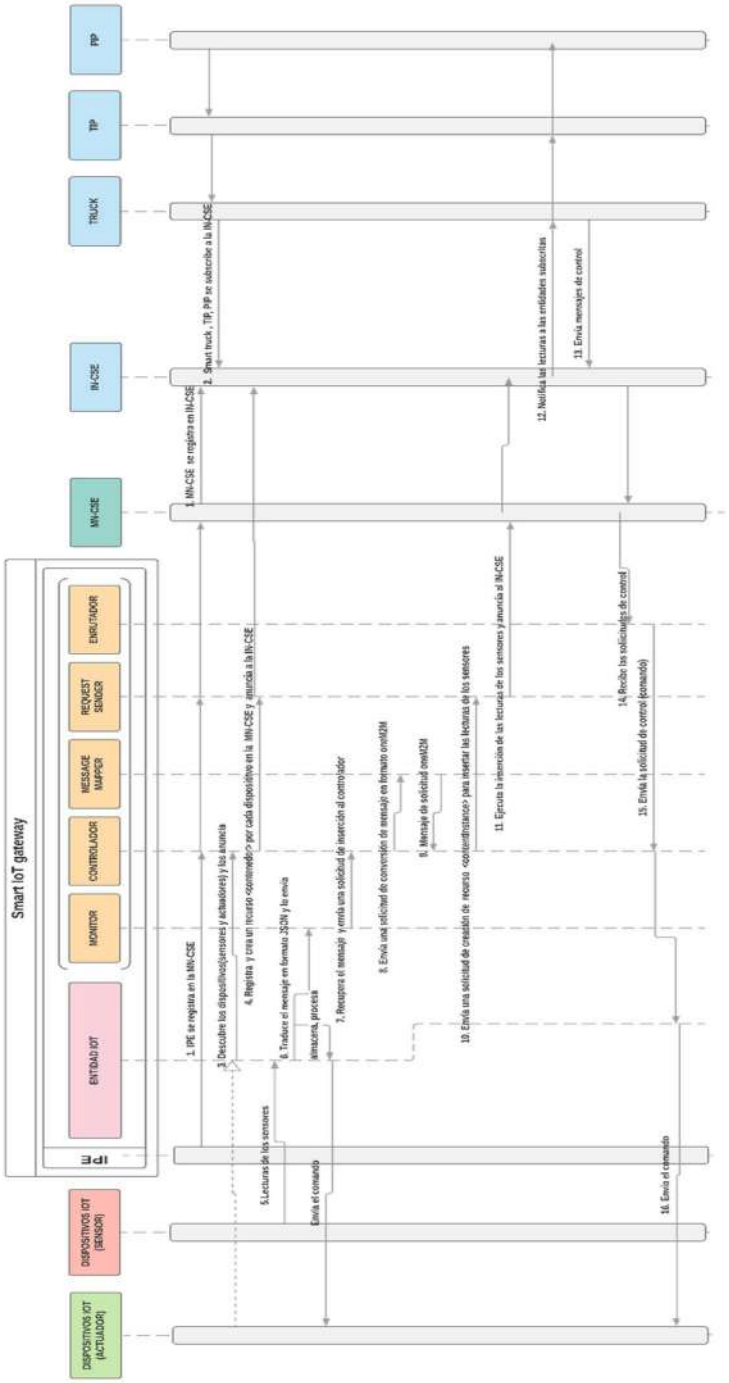


Figura 5.4: Flujo de información de los procedimientos de interconexión

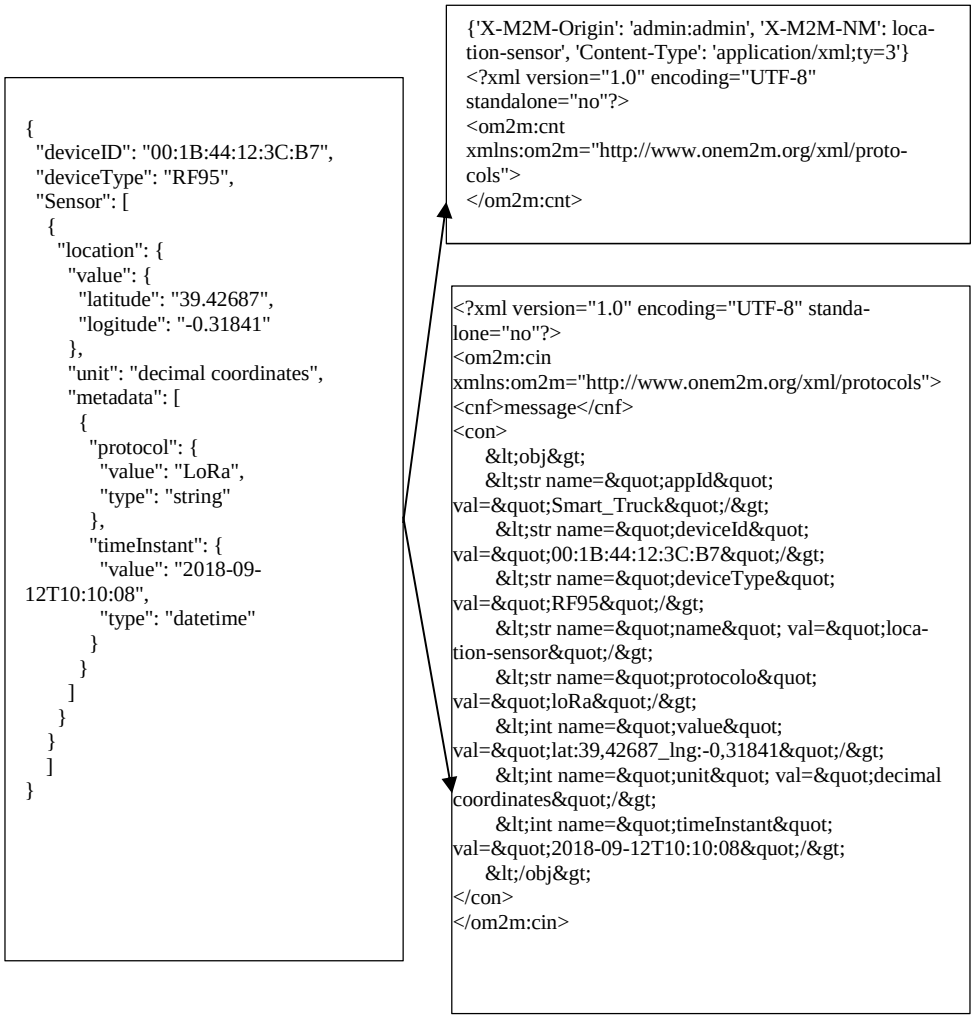


Figura 5.5: Transformación de datos en mensajes de solicitud primitiva oneM2M

5.6.3 Implementación y servicios proporcionados a partir de la interconexión

En este apartado primero ilustramos la implementación de la arquitectura de interconexión y de la aplicación "*Smart Truck*" utilizada como parte de este *testbed*. Posteriormente indicamos los posibles servicios que se podrían ofrecer gracias al uso de la arquitectura de interconexión.

5.6.3.1 Implementación

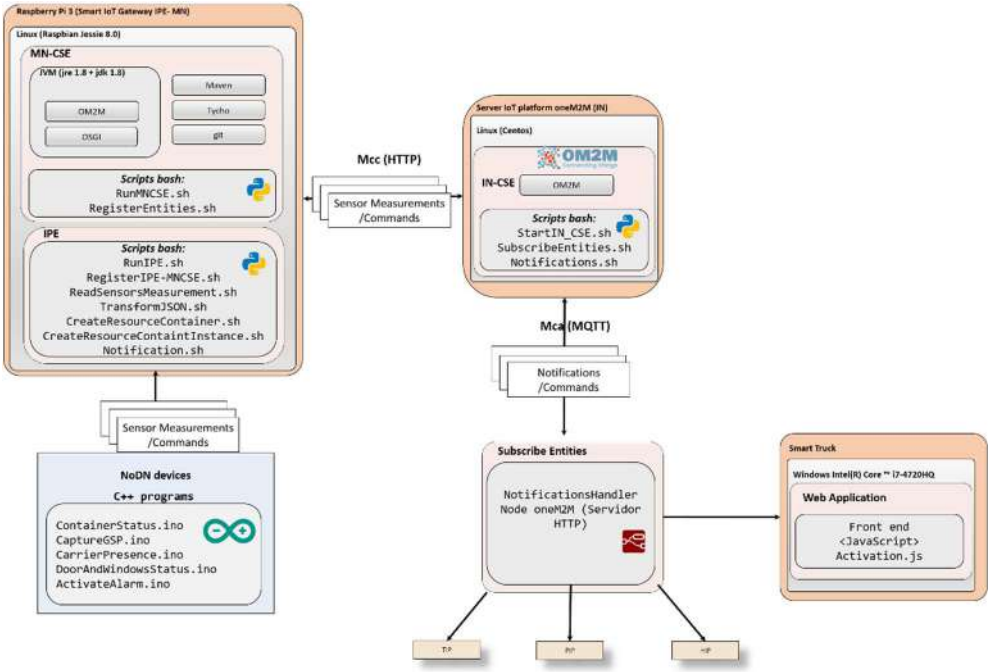


Figura 5.6: Diagrama funcional del *testbed* implementado

En la Figura 5.6 se presenta el diagrama de los programas de software desarrollados y las herramientas y plataformas utilizadas para el experimento de interconexión. En tal sentido, se configuró y desplegó un total de cinco dispositivos prototipo IoT individuales (un dispositivo IoT integra un sensor o un actuador) instalados en dos automóviles que han sido utilizados para realizar las pruebas. La Tabla 5.1 muestra los dispositivos empleados, así como la información que cada uno ellos proporcionan a las plataformas IoT involucradas. Los sensores y

actuadores fueron configurados a través de un conjunto de programas desarrollados en el entorno de programación Arduino ("Plataforma Arduino," 2016) basado en el lenguaje de programación C++.

Tabla 5.1: Dispositivos IoT empleados en el caso de uso

Dispositivo		Información	Tecnología de comunicación	Plataformas IoT
GPS NEO-6 sensor		Localización GPS	LoRA	TIP, PIP
Ultrasónico JSN-SR04T sensor		Estado del contenedor por proximidad(cargado o vacío)	ZigBee	PIP
PIR motion sensor		Presencia o ausencia del transportista en la cabina del vehículo	Wi-Fi-MQTT	TIP
MC-38 wired door/windows sensor magnetic switch		Apertura o cierre de puertas y ventanas del camión	Bluetooth	TIP

Buzzer YI-44 (actuador)		Zumbador	6LowPAN-CoAP	TIP

La estructura y configuración del *gateway* están basadas en el desarrollo previo realizado para el caso de uso AAL detallado en el capítulo 4 (Sección 4.5.2). En este caso se empleó una versión actualizada de la plataforma de desarrollo (Raspberry Pi 3 b+). El MN-CSE ha sido implementado utilizando la solución MN-CSE de Eclipse OM2M (Eclipse Foundation, 2016) compatible con oneM2M. Debido a que la solución OM2M (MN-CSE) está desarrollado en Java y se ejecuta sobre una capa OSGI basada en Eclipse *Equinox*, fue necesario para su ejecución la instalación de la herramienta *Maven y git* para automatizar el proceso de compilación y para clonar el repositorio de la solución, respectivamente. Siguiendo con la filosofía de utilizar estándares abiertos, se desarrolló un conjunto de funciones ejecutadas a través de *scripts bash en Python* que permiten realizar todas las funcionalidades de cada uno de los bloques del IPE.

El servidor oneM2M (IN-CSE) se ha implementado utilizando la solución IN-CSE también de la plataforma de código abierto Eclipse OM2M (Eclipse Foundation, 2016). OM2M (IN-CSE) se ejecutó en un servidor *cloud* privado para garantizar el acceso a los datos de los dispositivos solo a las plataformas involucradas en el caso de uso. El *broker* MQTT que reside en la plataforma OM2M permite que el IN-CSE admitan la interfaz Mcc y Mca vinculadas al protocolo MQTT y HTTP. La interfaz Mcc que interconecta el MN-CSE con el IN-CSE se implementó con mensajes oneM2M vinculados al protocolo HTTP y la interfaz Mca que interconecta al IN-CSE con cada una de las plataformas IoT involucradas en el caso de uso y la aplicación "*Smart Truck*" se implementó con mensajes oneM2M vinculados al protocolo MQTT.

Para la recepción de notificaciones se crearon y configuraron 4 recursos de subscripción en *Node-red* (JS Foundation, 2017) empleando los nodos *Notifications-Handler* oM2M previamente desarrollados por el laboratorio de análisis y arquitectura de sistemas (Francia) y la universidad Nacional Chia Tung (Taiwan)

(LAAS-CNRS;NCTU, 2018). *Node-red* es una herramienta de programación de código abierto gráfica basada en flujos que se utiliza principalmente para aplicaciones controladas por eventos, como IoT. Los nodos *NotificationsHandler* actúan como un servidor HTTP y permiten recibir notificaciones sobre los diferentes recursos a los que las plataformas IoT y la aplicación "*Smart Truck*" se han suscrito una vez que se crea una nueva instancia de datos.

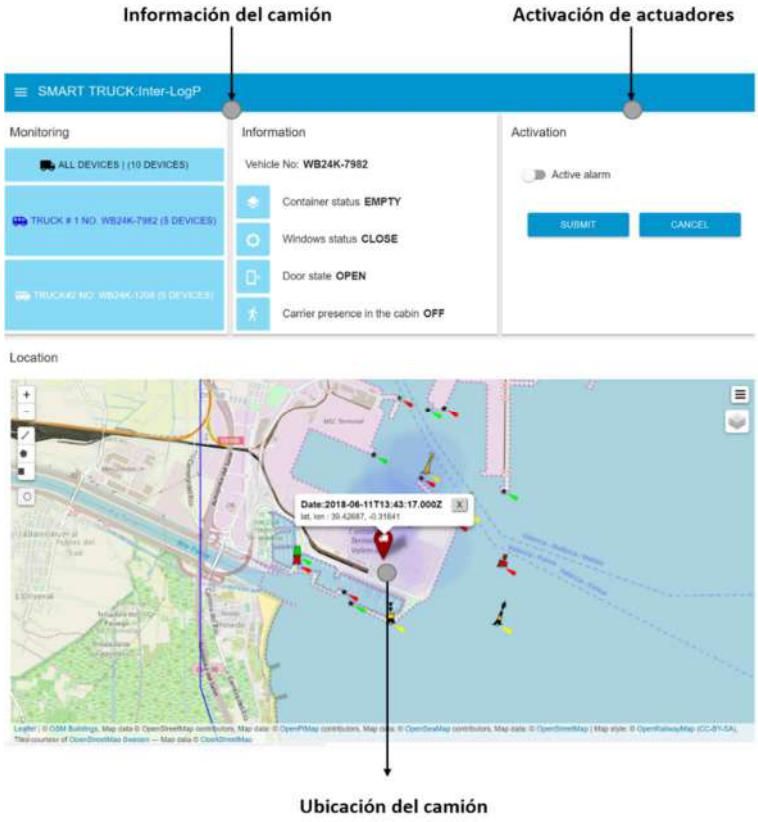


Figura 5.7: Aplicación Web "Smart truck"

Como se observa en la Figura 5.7, la aplicación "*Smart Truck*" permite visualizar los datos de los sensores implementados en los camiones (representados por los automóviles) en una interfaz gráfica de usuario (*front end*) llamada *Dashboard*, con el objetivo de obtener una visión general rápida de la localización del camión, así como también de su entorno. La interfaz también se implementó en *node-red* basada en JavaScript. En la interfaz, se habilitó una función llamada "activation" (a

través de botones) con el objetivo de enviar peticiones de control (*control messages*) a los actuadores registrados en la plataforma OM2M.

5.6.3.2 Servicios

Con esta implementación de la arquitectura se plantea 5 posibles servicios que la arquitectura podría ofrecer para apoyar al escenario control de acceso, tráfico y asistencia operativa, previos acuerdos establecidos entre los principales actores involucrados en el caso de uso.

- **Control de acceso a las instalaciones portuarias**

  Actualmente, los camiones para acceder al puerto deben tener una orden de transporte válida y un horario de cita. Mediante el uso de la arquitectura de interconexión propuesta, el *smart IoT gateway-IPE* propaga automáticamente la ubicación del sensor GPS del camión y la placa del camión a la plataforma del servidor oneM2M. Por lo tanto, en base a esta información, el PIP puede verificar rápidamente la validez de la orden de transporte, mientras que el TIP puede hacer una verificación cruzada para validar la cita. Una vez que el camión llega al puerto y se identifique por medio del sistema de lector de placa de la PIP, la puerta del puerto se podría activar automáticamente, gracias a la información proporcionada por la arquitectura de interconexión a través de la plataforma oneM2M. De la misma manera, cuando el camión llega a la terminal de contenedores, el TIP podría permitir que el camión acceda automáticamente a las instalaciones de la terminal para entregar y/o recoger contenedores. Esto permitirá evitar colas en las puertas de acceso al puerto y la terminal.

- **Control de los camiones en las instalaciones portuarias**

  Por razones de seguridad, las autoridades portuarias deben monitorizar los camiones durante su estancia en las instalaciones portuarias. En este sentido, los sensores PIR instalados en la cabina del camión proporcionar información sobre la presencia o ausencia del transportista en dicha cabina. De la misma manera, los sensores magnéticos instalados en las puertas y ventanas ayudar a detectar la apertura de las mismas. Toda esta información, junto con la ubicación GPS de los camiones provistos a través de la plataforma de interconexión vía la plataforma oneM2M, puede ayudar a la PIP a determinar si el transportista se encuentra dentro de un área insegura o no autorizada. En el caso de que se detecte que el transportista se mueve en lugares no autorizados o peligrosos, se pueden realizar dos acciones simultáneamente:

(i) la arquitectura de interconexión puede enviar avisos y mensajes de información a una aplicación móvil de teléfono celular ofrecida a los transportistas y suscrita al *smart IoT gateway-IPE*, así como activar una alarma en el camión, y (ii) las operaciones de las autoridades portuarias pueden ser detenidas automáticamente por el personal del terminal.

- **Servicio de orientación**

Con frecuencia, algunos camiones no encuentran inmediatamente la ruta correcta para la recolección de contenedores y/o puntos de entrega. El transportista puede recurrir al servicio de guía de GPS en su teléfono celular como una aplicación móvil ofrecida por las autoridades portuarias al compartir la ubicación GPS y la placa del camión a través de nuestra arquitectura de interconexión. La aplicación móvil guía al transportista del camión directamente al contenedor o punto de recogida. La transición desde el área de la terminal al puerto sería transparente para el usuario y no requeriría cambiar a una aplicación diferente. Este servicio estará activo mientras el camión se encuentre dentro de las instalaciones portuarias.

- **Inspección mejorada del servicio de contenedores vacíos**

El objetivo de este servicio es automatizar la inspección de contenedores permitiendo una inspección más efectiva y eficiente de contenedores vacíos. Los camiones salen del puerto a través del carril vacío en caso de que lleven un contenedor vacío. Con los procedimientos actuales, los contenedores se seleccionan al azar para su inspección. Como una mejora para este proceso, el sensor de carga instalado en el contenedor puede proporcionar información de estado del contenedor a través de nuestra arquitectura. Por lo tanto, todas las plataformas que registran esta información en sus sistemas pueden recibir esta información de la plataforma oneM2M. Así que una vez que un camión llega a la puerta del puerto en el carril vacío, el TIP o PIP puede informar a la policía fronteriza si el contenedor debe revisarse o no. En el caso de que no sea necesario, la salida será automática.

- **Iluminación dinámica**

Actualmente, el sistema de iluminación dinámica (DLS, por sus siglas en inglés, *Dynamic real-time lighting system*) (Valenciaport, 2015) de TIP en el terminal del puerto de contenedores es capaz de reducir significativamente el consumo de energía de la luminaria de una manera inteligente y eficiente durante la noche. Sin embargo, en el caso de uso, DLS debe conocer la posición GPS

de cada vehículo en el área de la terminal para poder aplicar el modo de iluminación de bajo consumo. Si no, el área de la terminal estará completamente iluminada por razones de seguridad. Por lo tanto, solo está activo si no hay camiones operando en la terminal de contenedores, lo cual es una situación rara. El DLS podría conocer esta información a través de la ubicación GPS del camión compartida por nuestra arquitectura de interconexión. Como resultado, el DLS puede funcionar a su máximo potencial, siendo el modo de bajo consumo activo toda la noche, logrando ahorros de energía de hasta el 78% (según los resultados iniciales proporcionados por la terminal portuaria) en comparación con los resultados anteriores al envío de la posición GPS de los camiones.

Estos servicios verifican que la implementación de nuestra arquitectura de interconexión propuesta en el escenario de control de acceso, tráfico y asistencia operativa puede lograr varios beneficios, incluida la minimización de colas y tiempos de espera en la entrada del terminal, el aumento del control y la seguridad en el área del puerto, la regulación de los flujos de tráfico dentro del terminal y la optimización de las operaciones portuarias.

5.6.3.3 Resultados preliminares

Las colas en la entrada del puerto de Valencia son una situación frecuente y muchas veces llegan a un máximo de 1 o 2 horas de espera (De acuerdo a los datos de las autoridades portuarias). Esta situación podría minimizarse o incluso evitarse completamente utilizando la arquitectura de interconexión propuesta. Aparte de la pérdida de tiempo, esta situación conlleva un consumo de combustible promedio adicional por camión y por hora de 88% (es decir, 24,42 l / h), así como 0.56 litros adicionales por hora si se recurre al aire acondicionado (principalmente durante el verano), y 2 litros adicionales por hora para el transporte de contenedores refrigerados (consulte la Tabla 5.2). Este último representa al 25% de los camiones de transporte que operan en el puerto de Valencia.

Tabla 5.2: Consumo de combustible promedio adicional por camión por hora

Indicadores	Litros extras por hora
Consumo medio de combustible	24,42l
Aire acondicionado	0,56l
Refrigeración	2l

La Figura 5.8, muestra el consumo adicional de combustible (en litros) de un camión durante un viaje debido a la existencia de colas de espera o atascos de tráfico.

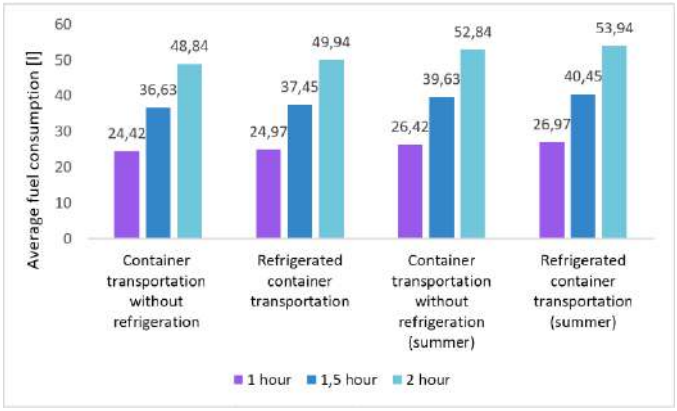


Figura 5.8: Consumo medio adicional de combustible de un camión por viaje

Sobre la base de los resultados de la Figura 5.8, y considerando que un camión realiza dos entregas de carga diarias, estimamos que las compañías transportistas podrían ahorrar hasta 140,24 € (con un costo promedio de combustible de € 1,30 / l) en costos de combustible por camión cada día.

5.7 Extensibilidad de la arquitectura de interconexión

Se ha verificado que la arquitectura puede ser utilizada para la integración de dispositivos heterogéneos con otra plataforma como es el caso de la plataforma FIWARE. A continuación, se describe brevemente la plataforma FIWARE y los procedimientos de interconexión establecidos para la integración e interacción de los dispositivos IoT con dicha plataforma.

5.7.1 FIWARE

Es una plataforma IoT de código abierto financiada por la Unión Europea bajo el programa *Future Internet Public Private Partnership (FI-PPP)* diseñada para acelerar el desarrollo de soluciones IoT en el ecosistema de Internet del futuro. FIWARE proporciona capacidades *cloud* basada en Open Stack (Corradi *et al.*, 2014), que

contiene una rica biblioteca de componentes denominados habilitadores genéricos (GEs, por sus siglas en inglés *Generic Enablers*). En particular, los GEs se consideran componentes de *software* que ofrecen varias funcionalidades junto con protocolos e interfaces para la operación y comunicación. Los GEs son proporcionados como servicios SaaS (Chang, 2014), dichos servicios reutizables implementan funciones de propósito general que se pueden acceder a través de interfaces (APIs) estándar y abiertas que facilitan tareas como la integración de sensores y actuadores y otros dispositivos, el análisis y procesamiento de datos y la implementación de interfaces avanzadas para interactuar con los usuarios.

Los GEs están organizados en siete capítulos tecnológicos diferentes, que proporcionan capacidades muy diversas. Así por ejemplo el capítulo *cloud* proporciona capacidades de cómputo, almacenamiento y recursos para el despliegue de aplicaciones y servicios en entornos FIWARE; en el capítulo de gestión de datos y contexto se proporciona capacidades para acceder, procesar y analizar grandes cantidades de datos; en el capítulo IoT se proporcionan las capacidades para conectar los dispositivos físicos para que sean accesibles a las aplicaciones; en el capítulo entrega de datos servicios y aplicaciones se proporciona herramientas de inteligencia de negocio y desarrollo de interfaces, en el capítulo interfaces a redes y dispositivos se proporciona interfaces estandarizadas para conectar los servicios de la plataforma, en el capítulo interfaz de usuario basada en web avanzada provee formas de interacción para mejorar la experiencia de Internet del futuro como la realidad aumentada o 3D. Finalmente, el capítulo seguridad proporciona mecanismo de seguridad y privacidad para la entrega de servicios de la plataforma.

Desde su lanzamiento en 2012 hasta la actualidad se han realizado varias versiones de la arquitectura (R1 a R6). La arquitectura que se muestra en la Figura 5.9 toma como referencia la R4, y es el resultado de la combinación de los GEs de los capítulos gestión de datos y contexto e IoT. Se trata de una arquitectura simple que permite el despliegue de diversos escenarios IoT, en los cuales varios dispositivos podrían ser integrados a la plataforma FIWARE.

FIWARE promueve y emplea el estándar FIWARE NGSI (del inglés *Next Generation Service Interfaces*), basado en la especificación NGSI 9/10 de OMA(Open Mobile Alliance, 2012). FIWARE NGSI describe como recolectar, administrar y publicar información de contexto proporcionando elementos que permiten explotar estos datos independientemente de la fuente de datos. Iniciativas como OASC (por sus siglas en inglés, *Open & Agile Smart Cities*) utilizan FIWARE

NGSI como el estándar de facto para el desarrollo de soluciones de ciudades inteligentes.

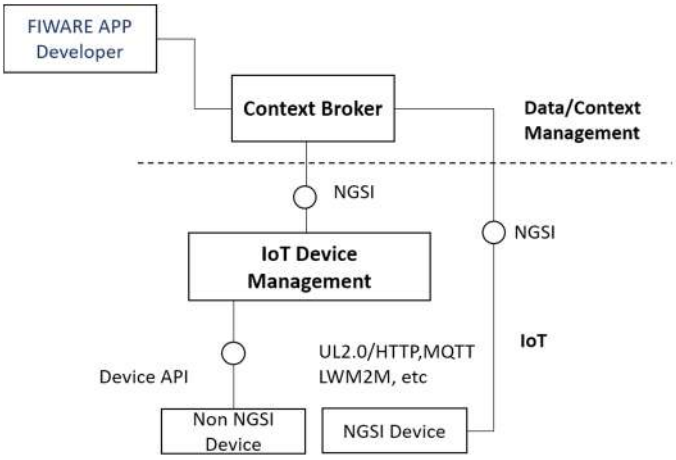


Figura 5.9: Arquitectura IoT de FIWARE

Como se observa en la Figura 5.9, de manera similar a oneM2M, en FIWARE se distinguen dos tipos de dispositivos: compatibles con la plataforma (NGSI Device) y los dispositivos *legacy* (Non NGSI Device).

El primer tipo de dispositivos son todos aquellos dispositivos que implementan el protocolo de comunicaciones OMA NGSI por lo que pueden comunicarse con el *GE context broker* directamente, mientras el segundo tipo de dispositivos se comunican a través del *GE IoT Device Manager,* que corresponde al capítulo IoT.

A continuación, se describe brevemente cada uno de los GEs de la arquitectura FIWARE mostrados en la Figura 5.9.

- **Backend Device Management GE** realiza dos funciones principales. Por un lado, la asignación de entidades de contexto OMA NGSI a los dispositivos IoT que consiste en crear una entidad de contexto en formato NGSI para cada uno de los recursos de los dispositivos IoT. Por otro lado, maneja la traducción de los protocolos IoT hacia /desde el protocolo OMA NGSI De acuerdo a las especificaciones de FIWARE, los protocolos específicos admitidos son HTTP Ultralight, MQTT, LWM2M.

- ***Context broker GE:*** correspondiente al capítulo de gestión de datos y contexto, es el principal y único componente obligatorio para desarrollar aplicaciones o servicios inteligentes basados en FIWARE, el cual permite actualizar y acceder a la información de contexto, y representa la interfaz natural para los desarrolladores de FIWARE. El GE *context broker* implementa la funcionalidad de administración de contexto a través de las interfaces OMA NGSI 9/10:
  - **NGSI-9:** para obtener información sobre la disponibilidad de información de contexto.
  - **NGSI-10:** para intercambiar información de contexto.

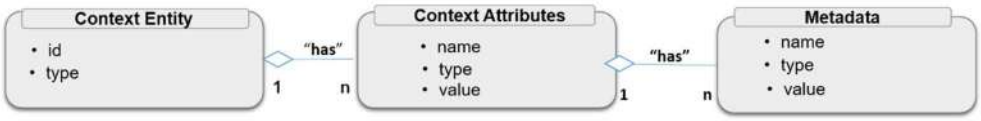


Figura 5.10: Modelo de información de contexto FIWARE NGSI

A diferencia de oneM2M que representa la información mediante recursos basados en contenedores, en FIWARE la información de contexto se estructura siguiendo el esquema que se muestra en la Figura 5.10. Las entidades de contexto son los elementos clave en el modelo de información FIWARE NGSI y representan a una "*thing*" que puede ser cualquier objeto físico o virtual. Por otro lado, los atributos de contexto son las propiedades que caracterizan a una entidad de contexto particular, mientras que los metadatos son opcionales y se pueden utilizar para describir las propiedades del valor del atributo.

5.7.2 Procedimientos de interconexión entre los dispositivos y la plataforma FIWARE

Para integrar los dispositivos IoT a la plataforma FIWARE se siguen los mismos procedimientos de interconexión en cuanto a funcionalidad. En este caso el servidor de plataforma IoT está representado por el *context broker* cuya implementación de código abierta es *Orion*, que implementa las APIs estándar OMA NGSI, que en el caso de uso INTER-LogP permiten a las plataformas consultar

sobre el contexto (que representa la información de los sensores) y subscribirse a cambios en el mismo, que se recibirán a través de notificaciones.

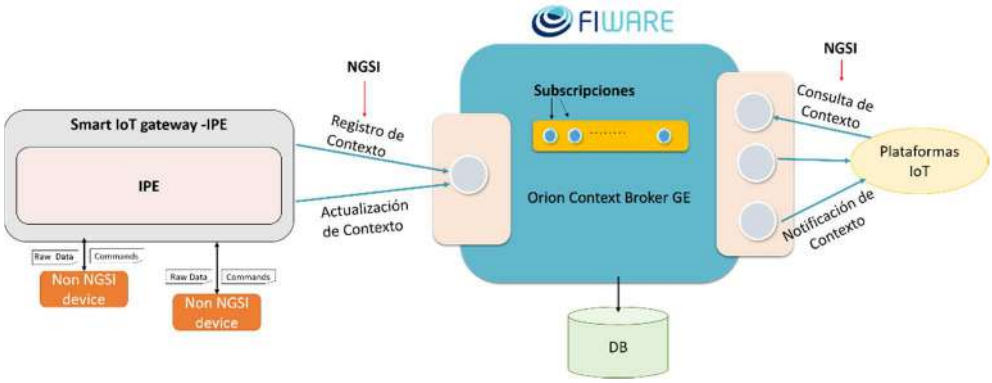


Figura 5.11: Esquema de interconexión con la plataforma FIWARE


```
{
 "deviceID": "00:1B:44:12:3C:B7",
 "deviceType": "RF95",
 "Sensor": [
  {
   "location": {
    "value": {
     "latitude": "39.42687",
     "logitude": "-0.31841"
    },
    "unit": "decimal coordinates",
    "metadata": [
     {
      "protocol": {
       "value": "LoRa",
       "type": "string"
      },
      "timeInstant": {
       "value": "2018-09-12T10:10:08",
       "type": "datetime"
      }
     }
    ]
   }
  }
 ]
}
```

```
{
 "id": "00:1B:44:12:3C:B7",
 "category": {
  "value": [
   "sensor"
  ]
 },
 "modelName": {
  "value": "RF95"
 },
 "controlledProperty": {
  "value": [
   "location"
  ]
 },
 "type": "Point",
 "coordinates": [
  "39.42687",
  "-0.31841"
 ],
 "timeInstant": {
  "value": "2018-09-12T10:10:08"
 },
 "supportedProtocol": {
  "value": "LoRa"
 }
}
```


Figura 5.12: Transformación de datos en formato JSON-LD

Por su parte, el *smart IoT gateway-IPE* representa al *Manager IoT device* de la arquitectura Fi-ware de referencia (Consultar Figura 5.9) y es el encargado de ejecutar todos los procedimientos de interconexión. En este caso, el *smart IoT gateway-IPE* a través del IPE*: (i)* se registra en el *context broker* (*ii*) crea una entidad cuyos atributos corresponden a cada uno de los sensores y actuadores y (iii) ejecuta los comandos procedentes del *Context Broker* en los dispositivos IoT a través del adaptador correspondiente. Para ilustrar lo explicado se muestra en la Figura 5.11 el esquema de interconexión del caso de uso, utilizando la plataforma FIWARE.

En cuanto al formato de datos, FIWARE maneja diferentes formato de datos (JSON, JSON-LD) dependiendo de la versión de la plataforma que se esté utilizando por tanto el IPE para integrar los dispositivos IoT a la plataforma FIWARE, debe realizar la transformación del formato de datos de los dispositivos a estos formatos. La Figura 5.12 muestra un ejemplo de la transformación al formato JSON-LD que ocurre internamente en el IPE.

En la publicación (D. C. Yacchirema, Sarabia-Jácome, *et al.*, 2018) se presenta un sistema inteligente para la monitorización del sueño mediante la integración de IoT y Big Data Analytics donde se muestra otro ejemplo representativo de integración de los dispositivos IoT con la plataforma FIWARE.

5.8 Conclusiones

La falta de interoperabilidad entre elementos *legacy* y plataformas estándar es una barrera importante para el desarrollo eficiente y el despliegue de soluciones horizontales de IoT que puedan interoperar entre sí. Para afrontar este desafío, siete grandes SDOs formaron la iniciativa global oneM2M para la estandarización de M2M e IoT. En este capítulo, se ha presentado el diseño e implementación de una arquitectura de interconexión para integrar diferentes dispositivos heterogéneos y las plataformas IoT (oneM2M y FIWARE). La propuesta implementa un IPE (*interworking proxy*) basado en las especificaciones de interconexión de oneM2M que permite la conversión del modelo de datos de los dispositivos no compatibles con las plataformas al modelo de datos oneM2M (basado en contenedores) y NGSI (basado en entidades) admitido por las plataformas oneM2M y FIWARE, respectivamente.

La propuesta se enfoca en alcanzar la interoperabilidad de diferentes protocolos y tecnologías de comunicación, el intercambio de información entre diferentes dispositivos heterogéneos y la plataformas integradas, con el objetivo de compartir la información proveniente de los dispositivos con partes externas (por ejemplo, plataformas de IoT, servicios y aplicaciones) interesados en usar dicha información a través de una plataforma estándar, así como de habilitar el envío de mensajes de control a los dispositivos de manera ubicua.

La arquitectura de interoperabilidad ha sido implementada en un *smart IoT gateway-IPE* embebido en un dispositivo SoC. El caso de uso Inter-LogP derivado del proyecto INTER-IoT verifica la viabilidad de la arquitectura propuesta a través de la implementación de un *testbed* enfocado a un escenario de aplicación real de IoT centrado en el transporte y la logística. Para el despliegue se han utilizado sensores heterogéneos, el *smart IoT gateway-IPE* instalado en automóviles y las implementaciones *OM2M* y *Orion Context broker* de las plataformas oneM2M y FIWARE, respectivamente. Los resultados preliminares de los beneficios de cinco servicios de interconexión que explotan la información compartida a través de la arquitectura propuesta generan mejoras en la cadena de valor incluida la minimización de colas y tiempos de espera en la entrada del terminal, el aumento del control y la seguridad en el área del puerto, la regulación de los flujos de tráfico dentro del terminal y la optimización de las operaciones portuarias. Además, describen una reducción promedio de € 1,30 / l del gasto de combustible por camión cada día. Mejoras que se obtienen gracias a la arquitectura de interconexión propuesta.

Capítulo 6

Sistema de detección de caídas para personas mayores utilizando IoT y Big data

« Si tuviera la suerte de alcanzar alguno de mis ideales, sería en nombre de toda la humanidad »

Nikola Tesla (1856-1943)

6.1 Introducción

La esperanza de vida ha aumentado en 5 años desde 2000 debido a los avances en el campo de la medicina. Según la Organización Mundial de la Salud (OMS), para 2050, la población actual de personas de edad avanzada (8,5%) aumentará, representando el 20% de la población mundial (W. He *et al.*, 2016). Sobre la base de estas tendencias, muchos países están adoptando políticas de envejecimiento saludable con el objetivo de ayudar a las personas mayores a llevar una vida activa e independiente (Bousquet *et al.*, 2015). En particular, garantizar un envejecimiento activo y saludable (AHA) de las personas mayores es uno de los mayores desafíos, pero también una gran oportunidad para la sociedad en las próximas décadas. La noción de AHA se ha caracterizado últimamente como un concepto amplio, que busca mejorar la calidad de vida (QoL por sus siglas en inglés, *Quality of life*) de las personas mayores a medida que envejecen, optimizando las oportunidades de salud, participación y seguridad (W. He *et al.*, 2016). En ese sentido, los problemas de salud de las personas mayores son cada vez más urgentes (D. C. Yacchirema, Sarabia-Jácome, *et al.*, 2018) y las caídas son los accidentes más comunes, cuya gravedad puede requerir a menudo atención médica. Las caídas

representan un riesgo de salud pública importante en todo el mundo para las personas mayores. Una caída no asistida a tiempo puede causar un deterioro funcional en las personas mayores y una disminución significativa en su movilidad, independencia y calidad de vida. Según la OMS, aproximadamente el 30% de las personas mayores de 65 años sufren accidentalmente una o más caídas por año, y para las personas mayores de 80 años, esta tasa aumenta hasta el 50%. Esta cifra es más alarmante si se considera que las caídas a menudo ocurren en ambientes interiores y están relacionadas con las actividades normales de la vida diaria (ADL, por sus siglas en inglés *normal activities of daily living*).

Una consecuencia grave de sufrir una caída es la "*long-lie*", que consiste en permanecer en el suelo durante largos períodos de tiempo (por una hora o más) luego de una caída hasta que llegue la ayuda (Robie, 2010), es decir representa una caída prolongada. La caída prolongada puede llevar a complicaciones de salud graves, como deshidratación, neumonía e hipotermia, que en muchos casos pueden causar la muerte dentro de los 6 meses posteriores a una caída. Por lo tanto, una caída no asistida en el tiempo en una persona mayor puede afectar negativamente su calidad de vida e independencia (Robie, 2010). En este contexto, el desarrollo de sistemas de IoT en tiempo real que contribuyan a detectar de manera eficiente las caídas y alertar a los servicios de emergencia en el menor tiempo posible es una necesidad social para reducir la ocurrencia de *long-lie*.

En este sentido, en este capítulo a más de evaluar la arquitectura de interoperabilidad propuesta en el Capítulo 3, se pretende brindar una solución para la detección de caídas con tres aportes importantes:

- Se presenta el diseño e implementación de un sistema inteligente de IoT que utiliza el algoritmo de aprendizaje supervisado *ensemble* para detectar las caídas en las personas mayores, denominado "IoTE-Fall". A través de IoTE-Fall basado en *fog computing,* tanto las caídas como las ADLs se pueden detectar en tiempo real. Al aprovechar *fog computing* en el borde de la red, el sistema propuesto ofrece muchos servicios avanzados como la interoperabilidad, transformación de datos, almacenamiento de datos, análisis de datos y notificación de alertas de emergencia. Además, para lograr un análisis en tiempo real de los datos de movimiento de las personas mayores, IoTE-Fall distribuye las tareas de análisis de datos entre un dispositivo *fog* (smart IoT gateway) y un servidor *cloud*.

- Previo a la implementación y para determinar el algoritmo que presenta el mejor rendimiento, se evalúan 4 de los algoritmos más utilizados en la

literatura para la detección de caídas (árboles de decisión, *ensemble*, *Deepnet* y regresión logística) en términos del AUC ROC, tiempo de procesamiento y tiempo de validación. *Ensemble* se identifica como el algoritmo con mayor rendimiento.

- El sistema propuesto se evalúa en términos de exactitud, precisión, sensibilidad y especificidad para demostrar la fiabilidad en la detección de caídas. Los experimentos se realizan con personas mayores voluntarias utilizando un dispositivo portátil para capturar los datos de movimiento y un dispositivo *fog* para el análisis de los datos. Se realiza un estudio comparativo de los resultados alcanzados por IoTE-Fall con otros estudios de investigación relacionados en la literatura.

Este capítulo comienza realizando una revisión del estado del arte y trabajos relacionados con la implementación de sistemas para la detección de caídas que hacen uso de tecnologías emergentes como IoT y algoritmos de *machine learning*. Posteriormente, se describe en detalle las etapas y arquitectura del sistema IoTE-Fall. Luego, se presentan los resultados experimentales y evaluaciones realizadas. Además, se realiza una comparación cualitativa con las principales soluciones de detección de caídas propuestas en la literatura. Finalmente, se destacan las principales conclusiones.

Este capítulo está basado en las publicaciones (Diana Yacchirema, Suárez de Puga, *et al.*, 2018; D. Yacchirema et al., 2019)

6.2 Estado del arte y trabajos relacionados

En la actualidad, se han propuesto varias soluciones para la detección de caídas en personas mayores. Estas soluciones se clasifican en tres tipos de acuerdo con la tecnología de sensores utilizada (Chaccour *et al.*, 2017): sistemas no portátiles (NWS, por sus siglas en inglés *non-wearable-based systems*), sistemas portátiles (WS, por sus siglas en *inglés wearable-based systems*) y sistemas basados en fusión o híbridos (FS, por sus siglas en inglés *fusion or hybrid-based systems*).

En particular, se ha demostrado que muchos sistemas NWS que utilizan dispositivos basados en la visión como (Min et al., 2018; Ma et al., 2014;Yang et al., 2016; Mastorakis & Makris, 2014; Kwolek & Kepski, 2014; Wang et al., 2017; Sehairi et al., 2018) son potentes y robustos para detectar caídas. Sin embargo, las principales desventajas de estos sistemas son su alto costo, tanto en la inversión en equipos como en los recursos computacionales para el procesamiento

de imágenes, y la consiguiente falta de privacidad para las personas mayores, ya que estos sistemas requieren que las cámaras se distribuyan estratégicamente en el ambiente interior en el que viven.

Para superar estas limitaciones, se han propuesto diversos sistemas WS, que emplean sensores de inercia, como acelerómetros y giroscopios, que generalmente se adhieren al cuerpo de las personas mayores para reconocer el movimiento cuando ocurre una caída.

Algunos de estos estudios (Álvarez de la Concepción et al., 2017; Fortino & Gravina, 2015; Aguiar et al., 2014) aprovechan el acelerómetro incorporado en el teléfono inteligente y el giroscopio (Kau & Chen, 2015; J. He et al., 2017) para monitorizar continuamente el movimiento de las personas mayores. Además, otros estudios combinan el uso de sensores de teléfonos inteligentes con sensores portátiles externos ubicados en diversas partes del cuerpo, como la muñeca, la cintura, el pecho, el tobillo (Santoyo-Ramón *et al.*, 2018), el hombro y el pie (Mao *et al.*, 2017), para obtener un conjunto de datos más completo de información sobre el movimiento de los sujetos. Sin embargo, la monitorización y la recopilación continua de datos, en algunos casos junto con la ejecución de aplicaciones de algoritmos de detección de caídas en el teléfono inteligente, produce un consumo de energía relativamente alto, lo que limita a que el sistema esté activo solo por cortos períodos de tiempo.

Por lo tanto, para hacer frente a esta situación, los dispositivos portátiles con acelerómetros se utilizan cada vez más en los sistemas WS porque ofrecen bajo consumo de energía, bajo costo, bajo peso, facilidad de operación, pequeño tamaño, la posibilidad de ser ubicados en varias partes del cuerpo y, lo más importante, la portabilidad. Como resultado, uno de los métodos más utilizados para la detección de caídas es el uso de un acelerómetro triaxial junto con un algoritmo basado en el umbral, que ha sido utilizado por algunos trabajos representativos (Dias et al., 2016; Phu et al., 2015; Santiago et al., 2017; Malheiros et al., 2017). Estos trabajos detectan una caída cuando la aceleración proveniente del acelerómetro triaxial incorporado en un dispositivo portátil está fuera del umbral establecido. Una de las mayores ventajas de usar los métodos basados en umbrales es la menor complejidad y el costo de cálculo en comparación con otros métodos. Sin embargo, encontrar un valor apropiado para el umbral que permita detectar todo tipo de caídas sin confundirse con algunas actividades normales de la vida diaria ha resultado ser un problema complicado.

Recientemente, se han propuesto sistemas WS basados en técnicas de aprendizaje automático (ML, por sus siglas en inglés *Machine learning*) para tratar este problema y mejorar la precisión en la detección de caídas. ML es una técnica en informática que implica la inferencia estadística de modelos a partir de datos para realizar predicciones automatizadas. ML construye un modelo a partir de datos de entrenamiento para predecir o resolver el problema dado (Alpaydin, 2009). En varios trabajos, los autores de los sistemas WS se centran en el uso de algunos algoritmos ML para tratar la detección de caídas y aumentar la precisión. Por ejemplo, el algoritmo de máquina de vectores de soporte no lineal (SVM, por sus siglas en inglés *non-linear support vector machine*) ha sido utilizado por varios trabajos (Mezghani et al., 2017; Pierleoni et al., 2015; Aziz et al., 2017) para extraer las características y obtener un significado del movimiento del cuerpo humano. Mezghani et al., (2017) monitorizaron los datos de caída y la orientación a través de un acelerómetro conectado a un *smart textile*. Del mismo modo, Pierleoni et al., (2015) también detectó la orientación de la caídas a través de la combinada de un acelerómetro, giroscopio y magnetómetro de 3 ejes. Dado que estos sistemas necesitan realizar dos extracciones de las características: el primero para identificar el pico de la caída y el segundo para detectar la orientación de la misma, requieren más procesamiento en comparación con los algoritmos que realizan una única extracción, por lo que es posible que la implementación de esta funcionalidad en dispositivos limitados no sea compatible.

Aziz et al., (2017) encontraron que el algoritmo SVM es el que presenta la mejor precisión en la detección de caídas al evaluar cinco algoritmos de aprendizaje diferentes (regresión logística, *Naïve Bayes*, árboles de decisión, K-vecino más cercano y SVM) en términos de tres medidas de rendimiento: sensibilidad, especificidad, y tasa de falsos positivos. El sistema se probó incorporando acelerómetros triaxiales ubicados en la cintura de voluntarios jóvenes. El algoritmo SVM logró la combinación más alta de sensibilidad (96%) y especificidad (96%) para distinguir las caídas de las ADL. En otro trabajo comparativo entre los algoritmos de aprendizaje: SVM, *k-nearest neighbor* (k-NN), y *multi-layer perceptron* (MLP) propuesto por Nguyen et al., (2018), los autores encontraron que MPL evaluado en un conjunto de datos abierto fue el algoritmo que obtuvo mejores resultados en la detección de caídas, con una sensibilidad, especificidad y una precisión de más del 98%. Sin embargo, la validación proporcionada por la sensibilidad, especificidad y precisión puede no ser suficiente cuando el objetivo es reducir la ocurrencia del síndrome "*long-lie*".

Un estudio similar realizado por Özdemir & Barshan, (2014) también comparó el rendimiento y la complejidad de cómputo lograda por seis algoritmos de aprendizaje automático en la distinción de caídas de las ADLs. Los algoritmos k-NN, método de mínimos cuadrados (LSM), máquinas de vectores de soporte (SVM), toma de decisiones bayesiana (BDM), distorsión de tiempo dinámico (DTW) y red neuronal artificial (ANN) se evaluaron en términos de sensibilidad, especificidad, precisión, tiempo de entrenamiento y tiempo de prueba. Los autores utilizaron los datos de tres sensores triaxiales (acelerómetro, giroscopio y magnetómetro) colocados en seis posiciones diferentes de voluntarios participantes para evaluar el rendimiento del algoritmo. En este caso, encontraron que el algoritmo k-NN y LSM proporcionaron una sensibilidad, especificidad y una precisión superior al 99%. Al asistir al tiempo de cómputo en ambas fases, entrenamiento y validación, LSM proporcionó resultados prometedores en comparación con k-NN, obteniendo un mínimo de 2. 2 ms de retraso en la fase de entrenamiento y hasta 32,7 ms en la prueba. Mientras que k-NN proporcionó 318.2 ms y 76.6 ms en la fase de entrenamiento y validación, respectivamente.

Santoyo-Ramón et al., (2018) realizó un estudio de evaluación sistemática entre SVM, k-NN, árboles de decisión y algoritmo nativo de Bayes, con cuatro sensores ubicados en diferentes posiciones del cuerpo como el pecho, la cintura, el tobillo y el muslo para discriminar entre caídas y ADL utilizando el análisis de varianza (ANOVA). Por un lado, los autores demostraron que el pecho y la cintura son las dos posiciones más adecuadas para ubicar los sensores y maximizar la efectividad del sistema, con la posición de la cintura como la alternativa más ergonómica. Por otro lado, los autores encontraron que el algoritmo SVM con un sensor colocado en la cintura logró una sensibilidad, especificidad superior al 93%. Sin embargo, el procesamiento de datos se lleva a cabo en el teléfono inteligente y puede ser un punto de falla en el sistema debido a la limitación inherente de la batería.

Al extraer series de tiempo del movimiento humano recuperado por un acelerómetro triaxial colocado en el tronco humano, Tong et al., (2013) utilizó el método oculto basado en el modelo de Markov (HMM) para detectar y predecir caídas. Los resultados del experimento muestran una tasa de éxito ideal en la detección de caídas (100% de sensibilidad y 100% de especificidad). Sin embargo, esta solución utiliza conjuntos de datos correspondientes a actividades simuladas de personas jóvenes tanto para el entrenamiento como para la validación del modelo.

Otro estudio propuesto por Aguiar et al., (2014) usó un acelerómetro incorporado en el teléfono inteligente para monitorizar continuamente los datos del movimiento de las personas mayores. Estos datos se utilizaron para validar tres clasificadores de aprendizaje diferentes: árboles de decisión, k-NN y *Naive Bayes*. Los resultados muestran que el algoritmo basado en el árbol de decisión presentó el mejor rendimiento, con valores de sensibilidad y especificidad más equilibrados en comparación con los otros algoritmos probados. Sin embargo, como resultado del consumo de energía relativamente alto de los teléfonos inteligentes, este sistema solo podría estar activo durante un corto período de tiempo.

Finalmente, Wang et al., (2017) desarrollaron el sistema *WiFall* que emplea los algoritmos SVM y de *ensemble* para clasificar tanto las ADL como las caídas. Los autores encontraron que el algoritmo *ensemble* mejora la precisión de la clasificación hasta en un 6% (en uno de los escenarios experimentales) en comparación con SVM. Sin embargo, estos resultados revelan que el porcentaje de falsas alarmas está por encima del 12%.

En este trabajo, hemos seguido un enfoque similar mediante el uso de un sensor portátil ubicado en la cintura de las personas mayores y un algoritmo *ensemble* para la detección de caídas, pero diferimos de los trabajos anteriores en la forma en que se construye el sistema. Primero, los datos de los movimientos de las personas mayores en el ambiente interior son capturados por un acelerómetro 3D integrado en un dispositivo portátil 6LowPAN. En segundo lugar, la caída de un anciano es detectada por el algoritmo *ensemble* que consta de 10 árboles de decisión que se crean y entrenan en el *cloud* pero que se instancian en el borde de la red donde tiene lugar la fase de detección. Para el entrenamiento y creación del modelo, se construyó un conjunto de datos denominado *Motion-DT* a partir del conocimiento histórico de un conjunto de datos de acceso público que contiene registros de caídas habituales y ADL de personas mayores. La técnica de ventanas deslizantes junto con el área de magnitud de señal normalizada (SMA) se utilizó para extraer los datos del movimiento que se corresponden con las caídas y las ADLs. Posteriormente, cuatro algoritmos de aprendizaje automático (clasificadores): árboles de decisión, *ensemble*, regresión logística y *Deepnets* se evalúan en términos de requisitos computacionales (tiempo de entrenamiento y tiempo de validación*)* y parámetros de optimización (AUC ROC), para seleccionar el algoritmo de clasificación más adecuado para el reconocimiento de caídas mediante la validación cruzada de *5-fold*. En tercer lugar, la efectividad del sistema propuesto para distinguir las ADL y las caídas detectadas se verifica mediante

experimentos realizados por voluntarios adultos mayores en términos de precisión, exactitud, sensibilidad y especificidad.

Finalmente, una de las principales innovaciones del sistema propuesto es que el proceso de detección de caídas se lleva a cabo en *smart IoT gateway* que proporciona capacidades de computación *fog* que permiten (i) reducir la ocurrencia del síndrome "*long-lie*" al detectar las caídas más cerca de donde los datos son producidos, en lugar de enviar todos los datos a través de largas rutas al *cloud* para su procesamiento; (ii) enviar notificaciones con información del tipo de caída y ubicación de la casa de la persona mayor a profesionales de la salud a través de un protocolo de IoT ligero y seguro que también proporciona niveles de calidad de servicio (QoS); y (iii) la transformación de los datos provenientes de los sensores en un formato de datos uniforme e interoperabilidad del sistema con plataformas AAL.

6.3 Relación de la arquitectura del sistema con la arquitectura global

El sistema de detección de caídas que presentamos en este capítulo apunta a demostrar otro ejemplo de la utilidad de la arquitectura global definida en el capítulo 3, a través de la implementación de la funcionalidad de análisis de datos en el *smart IoT gateway*. Una vez más se verifica que el *smart IoT gateway* puede implementarse de muchas maneras para adaptarse mejor a las necesidades de las aplicaciones IoT. Al ser el *smart IoT gateway* uno de los componentes principales del sistema, a través de la implementación subyacente del mismo se cumple una serie de requerimientos (R1-R10) de la arquitectura global (consultar Tabla 4.1 en el capítulo 4).

Uno de los requerimientos clave y adicionales que no se han abordado hasta ahora en la versión *smart IoT gateway* es el análisis de datos. En la búsqueda de cumplir con este requerimiento se instancia localmente en el *smart IoT gateway* un modelo de datos basado en *Big data,* el cual a partir del análisis de los datos generados por el sensor de movimiento integrado en un dispositivo portátil permite la detección de caídas en tiempo real. Para un análisis profundo de la información, varios modelos *Big data* se instancian en un servidor alojado en el *cloud*, empleando varios algoritmos de *machine learning* y herramientas de *big data*. Los modelos de datos se crean a partir de datos históricos provenientes de un conjunto de datos (*dataset)* abierto que contienen datos de caídas y actividades de personas adultas mayores.

Los modelos se evaluados en función de varias métricas estadísticas para determinar el algoritmo que presenta el mejor rendimiento en la detección de caídas.

Para la implementación *software* de la arquitectura se emplea un enfoque combinado basado en la computación *fog y cloud*. A través de este enfoque, se favorece el análisis distribuido de los datos, en el cual gran parte de la información es analizada y accionable en tiempo real y otro parte se puede utilizar para ejecutar análisis en períodos más largos. El *smart IoT gateway* actúa como un nodo *fog* ubicado cerca del dispositivo IoT con el que interactúa.

6.4 Sistema de detección de caídas utilizando IoT y Big Data

El diagrama de bloques que se muestra en la Figura 6.1 define las etapas del sistema de detección de caídas identificadas en esta investigación: extracción de características, capacitación y validación, y pruebas. Las ventanas deslizantes y las técnicas de SMA se utilizan para extraer las características que describen la señal sin procesar de los movimientos de las personas mayores de un conjunto de datos de acceso público. Estas características se almacenan en un nuevo conjunto de datos, *Motion-DT*, que se utiliza en el esquema de validación cruzada *k-fold* para entrenar y validar cuatro algoritmos de aprendizaje automático para encontrar el algoritmo más óptimo para la detección de caídas. El modelo seleccionado se verifica luego con las mediciones de aceleración obtenidas de un sensor acelerómetro MEMs con el fin de obtener el rendimiento real del sistema de detección de caídas.

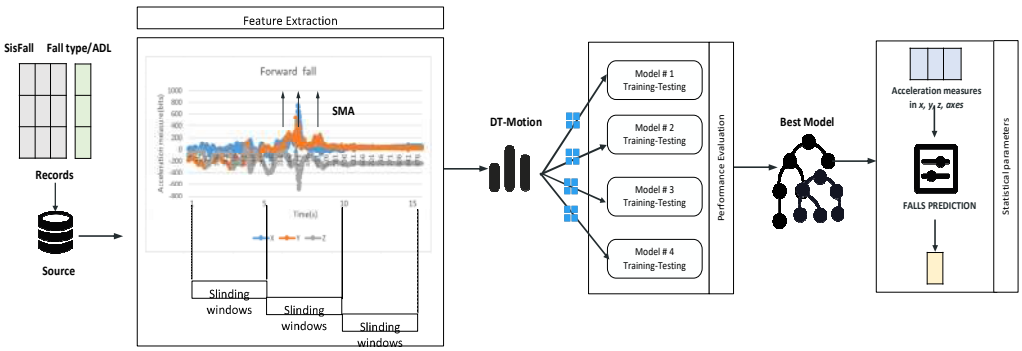


Figura 6.1: Etapas del sistema IoTE-Fall

6.4.1 Extracción de características

Un enfoque adecuado para entrenar y validar los sistemas de detección de caídas es utilizar el conocimiento histórico proveniente de conjuntos de datos que consisten en diferentes eventos pasados (caídas reales o simuladas, ADL o ambos) realizados por participantes que usan diferentes sensores ubicados en varias partes del cuerpo. El conocimiento histórico es esencial para entender qué comportamiento se espera. Por ejemplo, usar el conocimiento del comportamiento de los patrones de movimiento inesperados que se han producido cuando un adulto ha sufrido una caída permite predecir situaciones de riesgo cuando ocurren patrones similares de comportamiento. Sin embargo, el conocimiento histórico debe procesarse previamente para extraer las características que representan la señal sin procesar (*raw data*) de la caída como sea posible.

6.4.1.1 Conocimiento histórico de SisFall

En esta sección, nos centramos en identificar los períodos de actividad del usuario en un conjunto de datos de acceso público, SisFall de Sucerquia et al., (2017) con el fin de extraer las características más representativas tanto de las caídas como de las ADLs**.** El conjunto de datos de SisFall contiene registros de 38 participantes, de los cuales 15 son personas mayores de entre 60 y 75 años. Las señales se capturaron en experimentos de laboratorio utilizando tres sensores inerciales (2 acelerómetros: ADXL345 e ITG3200, y 1 giroscopio: MMA8451Q) integrados en un dispositivo de desarrollo propio. Las señales de aceleración y giroscopio se adquirieron con una frecuencia de muestreo de alrededor de 200 Hz. Los participantes ancianos realizaron 19 tipos de ADL y 15 tipos de caídas. Una explicación más detallada sobre el conjunto de datos SisFall se puede encontrar en (Sucerquia *et al.*, 2017).

Varios estudios encontraron que existe una variación clínicamente importante en el tipo de caída que puede sufrir una persona mayor (Rubenstein, 2006). En este estudio, nos centramos en los tipos más comunes de caídas que pueden causar graves riesgos para la salud de las personas mayores. Según un estudio realizado por (Youn *et al.*, 2017), las caídas más severas en personas mayores ocurren en una dirección hacia adelante cuando las personas caminan. Por otro lado, la caída hacia atrás es la caída más recurrente y puede causar graves daños a la salud humana, como una fractura en la muñeca o una fractura vertebral. Además, la caída lateral también es peligrosa porque las personas mayores pueden sufrir una fractura de cadera (Nevitt, MC, Cummings, 2018). Por lo tanto,

las caídas incluidas en esta investigación fueron caída hacia adelante (FF), caída hacia atrás (BF) y caída lateral (LF).

Además, tres ADL también se incluyeron en esta investigación, a saber, caminar (WA), subir escaleras (SCA) y sentarse (SA), para distinguir las caídas de estas actividades.

La colección del conjunto de datos correspondiente al experimento de una caída o ADL se considera un *trial*. Los *trials* utilizados del conjunto de datos de SisFall con respecto a estos tres tipos de caídas y ADL se muestran en la Tabla 6.1.

Tabla 6.1: Caídas y actividades de SisFall utilizadas en este trabajo

Código	Descripción	Trials	Duración
FF	Caída hacia adelante	5	15 s
BF	Caída hacia atrás	5	15 s
LF	Caída lateral	5	15 s
WA	Caminar	5	15 s
SCA	Subir escaleras	5	15 s
SA	Sentarse	5	15 s

6.4.1.2 Ventanas deslizantes junto con SMA

La técnica de ventanas deslizantes junto con SMA se aplicó a los *trials* seleccionados para obtener las características que representan los cambios de aceleración (picos) producidos por las caídas y las ADL. Para ello, se llevó a cabo el siguiente proceso:

(i) Inicialmente, con el objetivo de ubicar el segmento donde la señal de aceleración en los tres ejes sufre la mayor variación, se aplicó como técnica de división el método de la ventana deslizante a cada uno de los *trials*. Específicamente, se empleó una ventana de longitud fija de 5 segundos. Dicha longitud fue seleccionada debido al hecho de que las caídas ocurren en un período de tiempo relativamente corto. Esta ventana de longitud fija utilizada para deslizarse a través de toda la secuencia de cada *trial* es coherente con otras investigaciones relacionadas con la detección de caídas. Por ejemplo, Yoshida et al., (2005), Kangas et al., (2007), Shan & Yuan, (2010) emplearon las ventanas deslizantes 0.4 s, 0.6 s y 0.5 s, respectivamente. De hecho, según un estudio de Lombardi et al., (2009), una ventana de longitud de aproximadamente 0,5 s parece ser el mejor compromiso entre la efectividad, la complejidad y el bajo consumo de energía para verificar los valores de aceleración dentro de un evento de caída.

(ii) Posteriormente, el segmento con la mayor variación de movimiento en los tres ejes se extrae mediante el cálculo de SMA utilizando la ecuación 1. Donde *w* es la longitud de la ventana; las variables *xi, yi y zi* se refieren a la señal de aceleración de cada eje. La SMA es calculada en cada segmento mediante la sumatoria de las sumas de la magnitud de la aceleración en los tres ejes. SMA ha sido reportado por (Karantonis *et al.*, 2006) (A. M. Khan *et al.*, 2008) como una medida adecuada para distinguir entre periodos de trabajo y periodos de descanso en los acelerómetros.

$$SMA = \frac{1}{w}\sum_{i=1}^{w}(|x_i| + |y_i| + |z_i|) \quad (1)$$

Finalmente, como resultado de este proceso se obtiene el conjunto de datos denominado *Motion-DT* que contiene la señal de la aceleración con mayor variación junto con la tipología de las diferentes caídas y ADL. Para llevar a cabo este proceso se desarrolló un programa en Matlab.

6.4.2 Selección del algoritmo de aprendizaje supervisado

6.4.2.1 Algoritmos de aprendizaje supervisado

En el aprendizaje supervisado, los algoritmos trabajan con datos etiquetados, intentando encontrar una función, que dadas variables de entrada (*input data*), les asigne la etiqueta de salida adecuada, es decir predice el valor de la salida. Los algoritmos supervisados también denominados algoritmos de clasificación se utilizan en problemas en los cuales se conoce a priori el número de clases y las características presentes de cada clase (Bhavsar & Ganatra, 2012). En esta investigación, cada una de las caídas y ADL representan las clases y los datos de la aceleración en los ejes *x, y, z* representan las características.

Con el objetivo de seleccionar el algoritmo de aprendizaje supervisado más adecuado (el mejor modelo basado en estadísticas) para ser utilizado por IoTE-Fall en la detección de caídas, se evaluó un total de cuatro algoritmos de clasificación diferentes que incluyen árboles de decisión, *ensemble*, regresión logística y *Deepnet*. Los cuales se describen brevemente a continuación:

- **Regresión logística**

Es uno de los algoritmos de aprendizaje supervisado que se utiliza cada vez más en el campo de la biomedicina. Además, se ha utilizado para desarrollar modelos

predictivos de caída en trabajos propuestos por Aziz, Klenk, et al., (2017), K. Wang et al., (2017) y Lindholm et al., (2015). Este algoritmo calcula, para cada clase objetivo, una probabilidad modelada como un valor de función logística, cuyos parámetros son una combinación lineal de sus entradas. En particular, los campos de entrada $(X_1, X_{2,\ldots} X_k)$ se combinan linealmente utilizando coeficientes de regresión logística $(b_{0,i} , b_{1,i} , b_{2,i} , \ldots b_{k,i})$ para predecir un campo de salida $fi(X)$que es la probabilidad (ecuación 2) para cada una de las clases *i* del campo objetivo:

$$pi = + \frac{1}{1 + e^{-fi(x)}} \quad (2)$$

Donde:

$$fi(X) = (b_{0,i} + b_{1,i}X_1 + b_{2,i}X_2 \ldots . + b_{k,i}X_k)$$

La regresión logística intenta aprender los coeficientes k de la función lineal, $fi(X)$, utilizando técnicas de estimación de máxima verosimilitud. En particular, la forma de la curva que se utiliza es un ajuste natural para dividir los datos en grupos. La regresión logística funciona mejor en los casos en que las características son aproximadamente lineales y el problema se puede resolver de manera lineal. Esto se debe principalmente al hecho de que la regresión logística genera límites de decisión lineales para separar las clases objetivas.

- **Deepnet**

Este algoritmo de aprendizaje supervisado también se ha aplicado con éxito a la detección de caídas en trabajos propuestos por Fan et al., (2017), Jokanovic et al., (2016), Jankowski et al., (2015), Jokanović & Amin, (2018) y Shojaei-Hashemi et al., (2018). Deepnets se compone de un conjunto de capas interconectadas en las que los valores de entrada dan lugar a los valores de salida a través de una serie de nodos. Cada nodo es esencialmente una función que transforma las entidades de entrada en una colección de valores con sus pesos correspondientes. Entre la capa de entrada y la capa de salida, puede haber una o varias capas ocultas.

- **Árboles de decisión**

Este algoritmo se usa ampliamente para la clasificación de datos y su efectividad ha sido demostrada para abordar muchos problemas, incluida la detección de caídas en trabajos propuestos por Aguiar et al., (2014), Santoyo-Ramón et al., (2018), Aziz, Klenk, et al., (2017), Leu et al., (2017) y D. Yacchirema et al., (2018).

Los árboles de decisión están compuestos por nodos y ramas que crean un modelo de decisiones utilizando el algoritmo divide y vencerás. En otras palabras, el modelo de árbol de decisión consiste en una serie de rutas (nodos ramas) que conducen a decisiones basadas en una predicción (nodos hoja).

- **Ensemble**

El algoritmo *ensemble* ha logrado excelentes resultados y ha demostrado ser muy efectivo en la detección de caídas en trabajos propuestos por Y. Wang et al., (2017), Rougier et al., (2011), Stone & Skubic, (2015) y Yuwono et al., (2012). En particular, este algoritmo resuelve un problema de clasificación a través de un conjunto de árboles de decisión que se combinan para crear un modelo reforzado con un mejor desempeño predictivo que cualquiera de los árboles de decisión individuales. Este algoritmo supervisado es muy utilizado en el mundo de *machine learning* debido a que con menos complejidad, proporciona mejores resultados en una amplia variedad de problemas de aprendizaje automático al utilizar las fortalezas de los árboles de decisión individuales. (Zhang & Ma, 2012).

6.4.2.2 Selección del clasificador

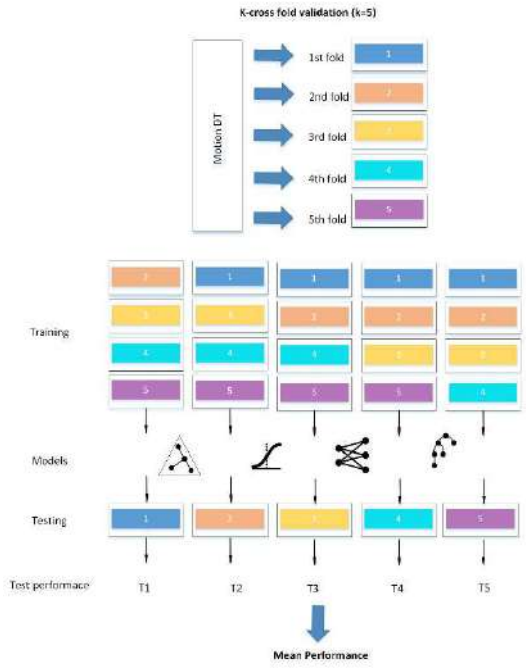


Figura 6.2: Validación cruzada (5-*fold*) aplicada a *Motion-DT* para crear y entrenar cada modelo

Para validar el rendimiento de cada uno de los algoritmos supervisados descritos previamente, se crearon varios modelos de predicción, empleando la técnica de validación cruzada *k-fold*. Concretamente se utilizó la validación cruzada 5-*fold* que consistió en dividir el conjunto de datos disponibles (*Motion-DT)* en k (5) partes iguales; hemos utilizados particiones basadas en los *trials*. Es decir la parte k1 (1st fold) está formada por los trials 1 (de cada caída y ADL), la parte k2 (2st fold) por los trials 2 y así sucesivamente. Así entrenamos el modelo con cuatro partes y validamos con una parte, se repite este procedimiento 5 veces. De esta manera, se crearon 5 modelos para cada algoritmo únicamente con los datos de entrenamiento. Posteriormente, cada modelo creado fue validado con los datos restantes (es decir con los datos de validación). La Figura 6.2 ilustra el proceso descrito.

Cada algoritmo de clasificación se evaluó en función de tres parámetros: área bajo la curva *ROC* (también conocida como AUC), tiempo de entrenamiento y tiempo de validación. Los tiempos de entrenamiento y validación se identifican como el tiempo que toma un algoritmo de clasificación para construir y probar el modelo, respectivamente. Mientras, que AUC representa el rendimiento esperado de cada algoritmo a través de un único valor escalar. Un valor de AUC más grande generalmente implica un mejor rendimiento. El AUC ha demostrado ser estadísticamente consistente y una medida más discriminatoria que la precisión, especialmente en la optimización de un algoritmo (Ling *et al.*, 2003). Además, esta métrica ha sido ampliamente empleada en varios trabajos de investigación (Dai *et al.*, 2010) (Li *et al.*, 2012) (Rougier *et al.*, 2011) para evaluar varios algoritmos de aprendizaje automático.

Para obtener la estimación del rendimiento de cada modelo, se trazaron las curvas ROC y se calcularon los valores de AUC para cada validación. De la misma manera, se calculan los tiempos de entrenamiento y validación. Luego, el resultado de las 5 validaciones en cada clasificador se promedió para obtener las medidas finales de validación cruzada, obteniendo de esta manera el rendimiento esperado por cada algoritmo.

El rendimiento del AUC para todos los clasificadores muestra resultados alentadores para todos los tipos de caídas y ADL después de la aplicación de la técnica de validación cruzada, como se muestra en la Fig. 6.3. Además, observamos que todos los clasificadores presentan un mejor desempeño predictivo para el tipo de caída y ADL FF (> 0.99) y WA (0.97), respectivamente.

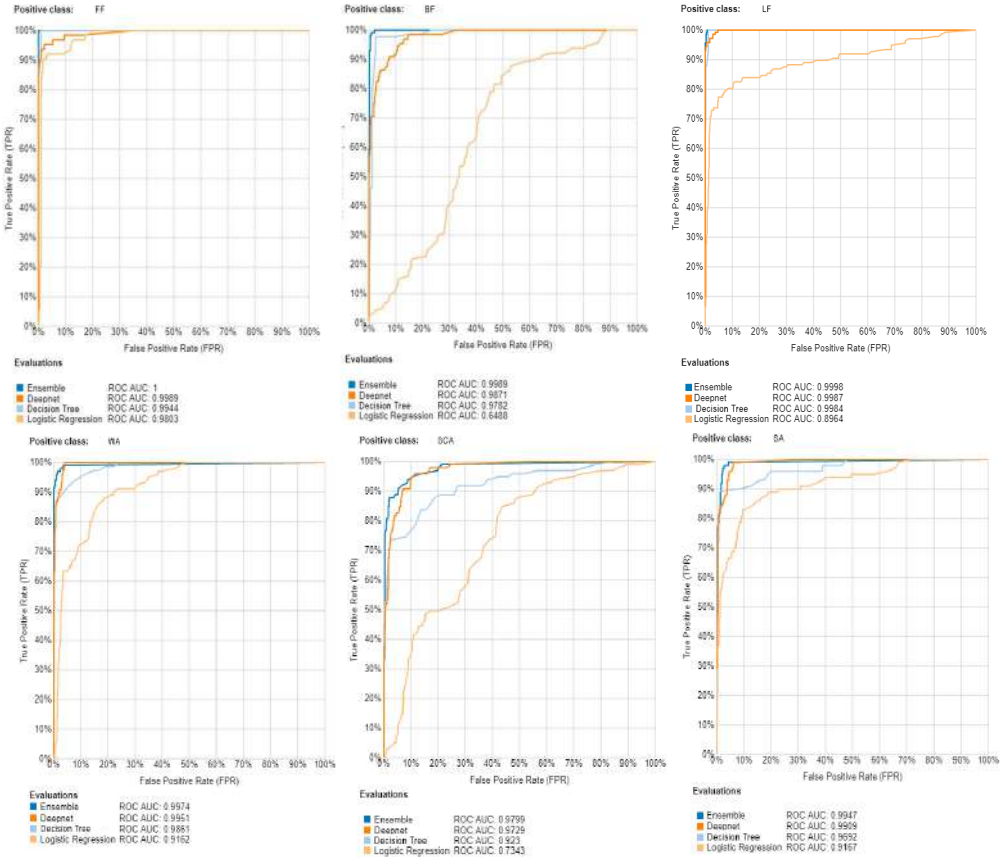


Figura 6.3: Curvas ROC y valores de AUC alcanzados para cada tipo de caída con un promedio de 0.96 (panel superior) y para cada tipo de ADL con un promedio de 0.95 (panel inferior).

La Tabla 6.2 muestra los resultados experimentales, que revelan que los algoritmo árboles de decisión, *ensemble* y *Deepnet* alcanzan un valor de AUC promedio superior a 0.97, que es mayor al valor alcanzado por el algoritmo regresión logística y es significativamente mejor que las estimaciones aleatorias (0.50). Sin embargo, se requieren extensos tiempos de entrenamiento y validación para los algoritmos de regresión logística y *Deepnet*. De acuerdo al tiempo de entrenamiento requerido, los algoritmos se pueden clasificar ascendentemente como árbol de decisiones, regresión logística, *ensemble* y *Deepnet*, con muy poca diferencia entre los tres primeros algoritmos. El clasificador *ensemble* logró el valor más alto de AUC, así como el tiempo de validación más bajo entre los clasificadores. Este

menor tiempo de validación se debe al hecho de que el entrenamiento y la validación de cada uno de los árboles de decisión individuales se realizan en paralelo. A partir de los resultados se seleccionó al algoritmo *ensemble* como el clasificador para aplicarse en el sistema.

Tabla 6.2: Estadísticas de la evaluación de los algoritmos de clasificación

Algoritmo de clasificación	AUC promedio	Tiempo de entrenamiento(s)	Tiempo de validación(s)
Decision Tree	0.9748	4.52	3.52
Ensemble	0.9951	5.75	3.48
Regresión logística	0.8487	5.4	4.80
Deepnet	0.9906	223.6	3.8

6.4.3 Sistema de detección de caídas

El IoTE-Fall propuesto, que se muestra en la Fig. 6.4, consta de cuatro componentes principales: un dispositivo portátil, una red de comunicación inalámbrica, un *smart IoT gateway* y un servidor *cloud*. Cada componente juega un papel importante en la detección de caídas y está asociado a cada uno de los grupos funcionales de la arquitectura para la interoperabilidad de dispositivos físicos en IoT descrita en el capítulo 3.

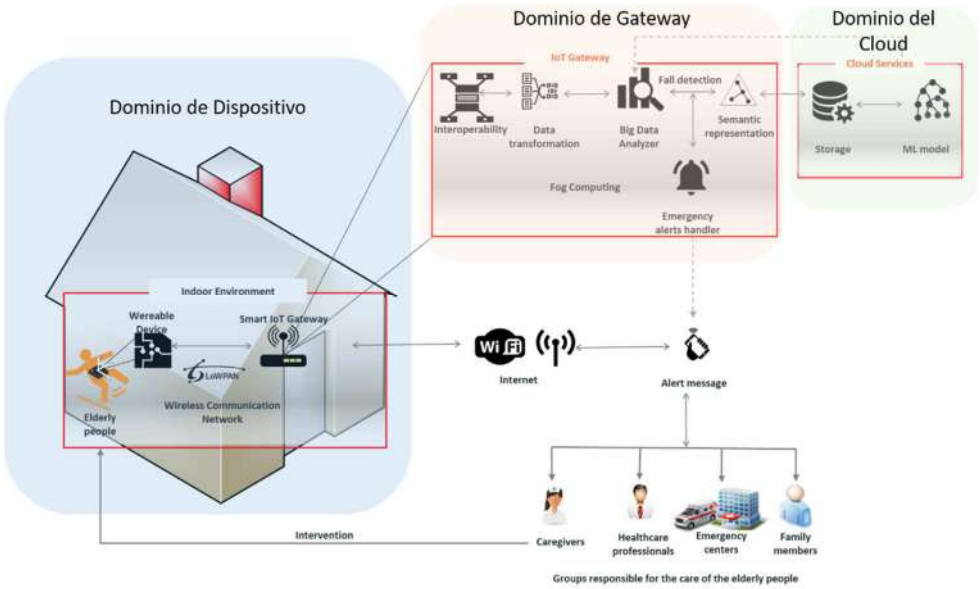


Figura 6.4: Visión general del sistema IoTE-Fall

El dispositivo portátil, integrado con sensores MEMs en movimiento, mide la aceleración (expresada en bits) de los movimientos corporales de las personas mayores y los transmite al *smart IoT gateway* mediante una red de área inalámbrica de baja potencia. El *smart IoT gateway* procesa y analiza los datos recibidos (utilizando el modelo *ensemble* previamente seleccionado) en el borde de la red para detectar rápidamente caídas y actuar en consecuencia enviando mensajes de alerta en tiempo real a los profesionales de la salud involucrados. El servidor *cloud* aloja el modelo ML, el servicio de almacenamiento y la plataforma IoT con la que se integra.

A continuación, se describe la funcionalidad que cada componente aporta al sistema IoTE-Fall, y como estos componentes fueron implementados.

6.4.3.1 Dispositivo Portátil

Se construyó un prototipo del dispositivo portátil a partir de la combinación de tres bloques modulares clave: el microcontrolador STM32, conectado con una tarjeta de expansión basada en el SPSGRF-915 con conectividad de RF por debajo de 1 GHz que funciona a 868 o 915 MHz y una tarjeta de expansión del sensor MEMS (X-NUCLEO-IKS01A2) desarrollado por *ST Microelectronics*. El microcontrolador STM32 está equipado con un procesador ARM de 32 bits Cortex-M3 diseñado para ofrecer un rendimiento muy alto, procesamiento digital de señales con bajo consumo de energía y bajo voltaje. La placa del sensor consta de varios sensores de potencia ultrabaja diminutos. Sin embargo, solo el sensor de movimiento MEMS (LSM6DS0) se usa para recopilar los datos de movimiento de las personas mayores. LSM6DS0 es un acelerómetro de 3D ejes que opera con un rango de aceleración de escala completa (± 2 / ± 4 / ± 8 g). El diagrama esquemático del dispositivo de detección se muestra en la Fig. 4.5 (a), y la estructura de hardware se muestra en la Fig. 4.5 (b).

(a) (b)

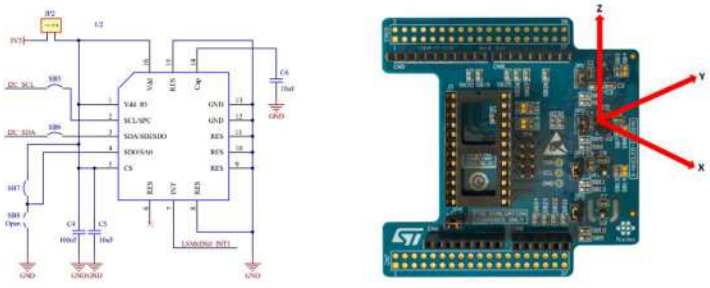


Figura 6.5: Dispositivo portátil (a) Diagrama esquemático; (b) Estructura de hardware

El firmware del dispositivo portátil se basa en "Contiki", que como ya se ha mencionado anteriormente es un sistema operativo de código abierto (OS) desarrollado para redes restringidas. Al utilizar el sistema operativo Contiki, obtenemos compatibilidad total con la pila IoT sobre 6LowPAN, en otras palabras, compatibilidad con los protocolos 6lowPAN, RPL y CoAP.

Para leer las mediciones del acelerómetro, se configuró un servidor CoAP integrado en el dispositivo portátil. Las mediciones se recuperaron utilizando el método CoAP GET en la ruta de recursos respectiva, incluida la dirección IPv6 y el puerto del servidor CoAP como se muestra en la Tabla 6.3.

Tabla 6.3: Ruta de acceso al recurso de aceleración del dispositivo portátil para obtener los datos de movimiento

Parámetro	Descripción
Sensor	LSM6DS0
Resource	Acceleration
Method	GET
Resource path	[coap://[aaaa::b00:f6ff:2d3b:d2c4]:5683/sensors/acceleration

6.4.3.2 Red de comunicaciones inalámbrica

La comunicación inalámbrica entre dispositivos y el *smart IoT gateway* se estableció mediante la tecnología inalámbrica de baja potencia IPv6 (6LowPAN) basada en el estándar IEEE 802.15.4, debido a su eficiencia energética y soporte de interoperabilidad. Construimos e implementamos una red 6LoWPAN compuesta de dos nodos 6lowPAN: el dispositivo portátil previamente descrito y un enrutador de borde 6lowPAN (6LoBR). 6LoBR juega un papel importante en la comunicación dentro y fuera de nuestra red 6LowPAN. El 6LoBR es responsable del intercambio de datos entre el dispositivo portátil y los servicios en la nube y de proporciona capacidades de reenvío y enrutamiento dentro de la red 6LoWPAN. En este trabajo el *smart IoT gateway* desempeña el papel de 6LoBR.

6.4.3.3 Smart IoT gateway y servicios *fog*

El *smart IoT gateway* asistido por *Fog* es el componente clave del sistema de detección de caídas y facilita los servicios de interoperabilidad, almacenamiento local, transformación de datos, análisis de datos, manejo de eventos y representación semántica.

- **Interoperabilidad**

El *Smart IoT gateway* actúa como un puente entre la red 6LowPAN y los servicios en la nube, permitiendo así la conectividad y la comunicación perfecta entre todos los componentes del sistema. Proporciona funciones de conversión de protocolo que incluyen los mecanismos de transición 6LowPAN IPv6 / IPv4 y la traducción de mensajes entre CoAP y los protocolos de transporte de telemetría de Message Queue Server (MQTT).

- **Transformación de datos**

El servicio de transformación de datos recibe los datos del movimiento de las personas mayores (valores de aceleración tridimensional en el eje x, y, z) y los mapea al formato CSV (del inglés, *comma-separated values*). Cada valor de aceleración se almacena localmente para ser utilizado por el servicio de análisis de datos.

- **Análisis de datos**

El análisis de datos en el *smart IoT gateway* asistido por *fog* es uno de los servicios más importantes en este sistema. No solo ayuda a reducir la carga del servidor en el *cloud* sino que también ayuda a extraer información importante que se puede utilizar para tomar decisiones críticas en tiempo real. En particular, este servicio se utiliza para detectar si los valores de la aceleración representan una caída o una ADL. Para ello, el servicio de análisis de datos hace uso del modelo de detección de caídas creado en el *cloud* a partir del algoritmo *ensemble* previamente seleccionado. A continuación, se describe las características del modelo construido para la detección de caídas.

- *Modelo para la detección de caídas*

Dos de las técnicas más utilizadas para la construcción de *ensemble* son *bagging*, y *boosting*. A fin de exhibir un mejor rendimiento de predicción para la construcción del modelo a partir de clasificadores más sencillos y rápidos se ha utilizado la técnica *bagging*. La idea básica de esta técnica es entrenar varios clasificadores a partir de los datos de entrenamiento originales (utilizando replicas *bootstrap)* y después, en la clasificación agregar las predicciones del conjunto de clasificadores base. Cada replica *bootstrap* consisten en una muestra aleatoria (con reemplazo) del conjunto de entrenamiento original. Así se crea un nuevo conjunto de entrenamiento para cada clasificador.

En este sistema, el modelo basado en *ensemble* ha sido configurado para construir 10 clasificadores base (árboles de decisión) que son independientes en términos de decisión, donde cada uno de ellos intenta predecir el campo objetivo del

problema (tipo de caída o ADL). Los modelos base se entrenaron a partir de réplicas *bootstrap* del conjunto de datos *Motion-DT*. Cada árbol de decisión aporta un voto a la clase que predice (FF, BF, LF, WA, SCA), y la clase seleccionada es la clase que más votos obtenga.

La Figura 6.6 ilustra los resultados de la clasificación de uno de los 10 modelos básicos. Este modelo indica la importancia de cada variable (la aceleración medida en los ejes x, y, z) en la capacidad de toma de decisiones (predicción) de los tipos de caídas y de las ADLs. Cada nodo representa una regla de clasificación (es decir, regla IF-THEN) para una variable. Las ramas que conducen desde el nodo indican la (s) ruta (s) que podrían tomarse. Cada camino tiene asociado un nivel de confianza. El grosor de las ramas indica la cantidad de datos de entrenamiento que esta ruta utilizó. El nodo de hoja constituye la decisión basada en una predicción.

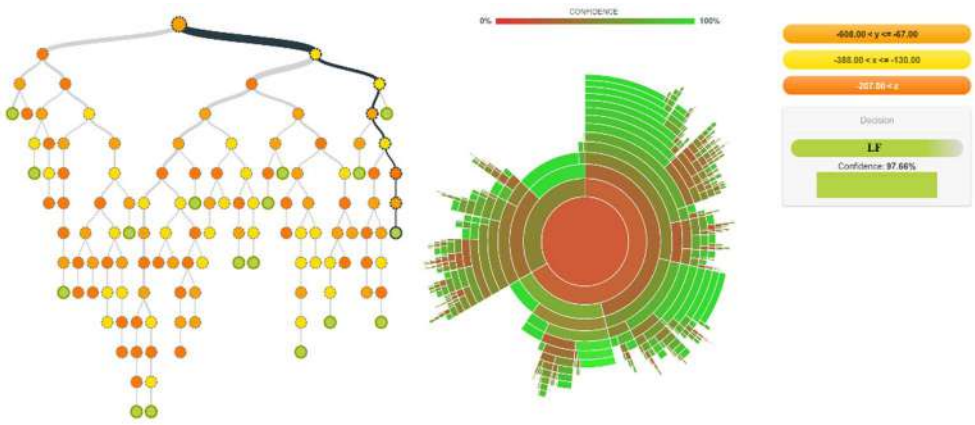


Figura 6.6: Clasificación para uno de los árboles de decisión que forman el *ensemble*: (izquierda) diagrama del árbol de decisión, (centro) grado de confianza alcanzado por el árbol de decisión al predecir la clase LF , (derecha) regla de decisión generada por el árbol de decisión para predecir la clase LF.

- *Detección de las caídas*

Para la detección de caídas, el servicio de análisis de datos en el *smart IoT gateway* crea una instancia local del modelo *ensemble* construido en el *cloud*. Al llevar el procesamiento cerca de la fuente de datos (es decir, cerca al dispositivo portátil 6lowPAN utilizado por las personas mayores), el sistema podría reducir la latencia y los gastos generales en la red, y como resultado, también podría reducir la

ocurrencia de la caída prolongada (*long-lie*). El modelo *ensemble* instanciado predice una caída o ADL utilizando el voto mayoritario a partir de los 10 modelos base, tomando como datos de entrada la información proveniente del servicio de transformación de datos. Si el resultado de la predicción es una caída, el sistema invoca al servicio de alertas de emergencia.

- **Alerta de emergencias**

El servicio de alertas de emergencia hace uso del módulo de manejo de eventos de la arquitectura, para enviar mensajes de alerta del evento de caída, junto con la posición geográfica de la casa de la persona mayor, a los grupos responsables del cuidado de las personas mayores previamente registradas en el sistema. La Figura 6.7 muestra un ejemplo de las notificaciones enviadas. Los datos de las caídas detectadas son enviados al servicio de almacenamiento en el *cloud* para su creación a posterior de un nuevo modelo de predicción a partir de estos eventos de caídas.

Figura 6.7: Ejemplo de notificaciones de detección de caídas

- **Representación semántica de datos**

La disponibilidad, el entendimiento común y la extracción del conocimiento de los datos del sistema son necesarios para que estos datos sean explotados por otros sistemas o servicios. El servicio de representación semántica de datos mapea los datos de cada caída o ADL antes de reenviarlo al *cloud*. El servicio de representación semántica proporciona descripciones de los eventos detectados y descripciones específicas del dominio utilizando los servicios semánticos proporcionados por la plataforma *UniversAAL*. *UniversAAL* es una plataforma de código abierto que proporciona un enfoque estandarizado para facilitar el desa-

rrollo de soluciones de vida asistida por el ambiente (AAL). El núcleo de *UniversAAL* es un *middleware* que contextualiza los datos y su funcionalidad a través de la definición de ontologías. *UniversAAL* proporciona un conjunto de ontologías que cubren los dominios de entornos inteligentes AAL y proporciona servicios semánticos para facilitar el modelado de datos, así como la representación y serialización de los mismos mediante el Lenguaje de ontología web (OWL, por sus siglas en inglés *Web ontology lenguage*) y el marco de descripción de recursos (RDF, por sus siglas en inglés, *Resource Description Framework*), respectivamente.

Para llevar a cabo la funcionalidad del servicio de representación semántica, en primer lugar, se extendió la ontología de la plataforma *UniverSAAL* "*Device*" para modelar los eventos de caída y ADL como se muestra en la Figura 6.8.

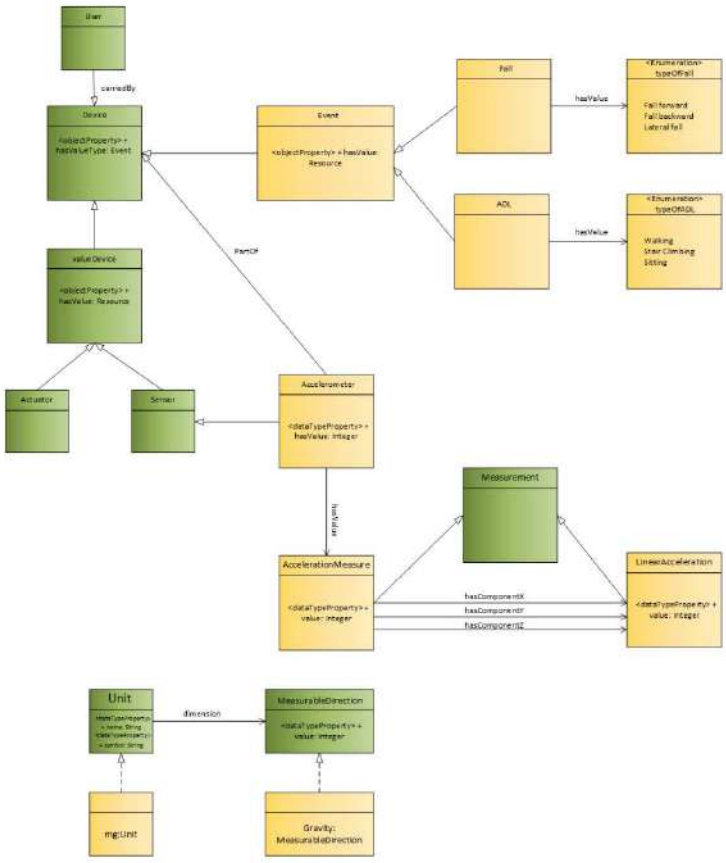


Figura 6.8: Ontología extendida (Device) de UniverSAAL para la representación semántica de los eventos de caídas y actividades

Los cuadros de color verde representan módulos semánticos ya incluidos en la ontología; mientras tanto, los cuadros de color amarillo representan los elementos extendidos. En segundo lugar, se integró el *smart IoT gateway* con la plataforma *UniverSAAL* mediante la implementación de los módulos funcionales del *proxy* de interconexión descrito en el capítulo 5. El *proxy* toma los datos de la aceleración de una caída o ADL, los mapea a un vocabulario ontológico utilizando la ontología extendida y los envía al *cloud* (es decir a la plataforma UniverSAAL). Los consumidores interesados (sistemas u otros servicios) pueden acceder a estos datos con notación semántica a través de las APIs proporcionadas por la plataforma *UniverSAAL*.

- **Implementación del smart IoT gateway**

El *smart IoT gateway* ha sido construido con un microcontrolador STM32 integrado con una tarjeta de expansión basada en el SPSGRF-915, y un Raspberry Pi 3 modelo B (RPI3) como entorno de ejecución, cuyas características son similares a las especificadas en el desarrollo previo en el capítulo 5. De igual manera los servicios de interoperabilidad, transformación de datos, análisis de datos, alertas de emergencia, representación semántica de datos se han implementado en el lenguaje de programación Python que se ejecuta en el sistema operativo instalado en la RPi3.

Para rastrear la ubicación de la casa de las personas mayores, un módulo GPS Ublox NEO-6M de bajo costo se ha conectado a RPI3. Para proporcionar la interoperabilidad, a más de las interfaces de comunicación admitidas por la RPi3, se agregó un adaptador 6LowPAN, cuya estructura y configuración, así como su implementación se ha definido en detalle en la sección 4.5.2.3.1. El adaptador permite recibir los datos de aceleración provenientes del dispositivo portátil. De manera similar al dispositivo portátil, todas las operaciones en el adaptador 6loWPAN se ejecutan en el sistema operativo "Contiki". RPL está configurado en el adaptador 6loWPAN para permitir la comunicación entre los nodos 6LoWPAN y el *smart IoT gateway*. Además, para traducir paquetes provenientes de nodos 6LoWPAN a paquetes IPv6 / IPV4 y viceversa, el adaptador 6loWPAN habilita un interfaz de red virtual para túneles en el RPI3 mediante el uso de la herramienta *tunslip6* que se ejecuta en el *Contiki* OS. Además, para recuperar los valores de aceleración del dispositivo portátil, el adaptador 6loWPAN implementa un cliente CoAP en RPI3 utilizando la biblioteca *aiocoap* de Python 3. Los datos analizados se almacenan temporalmente en la base de datos del *smart IoT Gateway*, que se construye a partir del motor de base de datos MySQL.

En la implementación del *smart IoT gateway* la instancia del modelo ML se implementa también en Python utilizando la API REST proporcionada por BigML. BigML, (2017) es un SaaS (del inglés *software as a service*) para el aprendizaje automático, diseñado para crear modelos predictivos e integrarlos en aplicaciones de software mediante el uso de APIs RESTful. El servicio de notificación de manejo de alertas de emergencia se implementa mediante el *broker* mosquitto MQTT v3. Para transportar los datos correspondientes a las caídas e información de contexto asociada, el servicio de alertas ha sido configurado con el protocolo SSL y con el nivel de QoS 0, a fin de garantizar, una comunicación segura extremo a extremo y la entrega inmediata de las alertas de emergencia ante la detección de una caída, respectivamente.

6.4.3.4 Servidor cloud

El *cloud* consta de varios servicios, incluidos el almacenamiento y el módulo de análisis de datos *cloud* que contiene el modelo ML. Además en este servidor se encuentra instanciado la plataforma *UniverSAAL*. El servicio de almacenamiento conserva la información de las caídas y las ADL provenientes del *smart IoT gateway* a fin de utilizar como conocimiento histórico a posterior para la creación y entrenamiento de un nuevo modelo predictivo. Por su parte, la plataforma UniverSAAL también almacena todos los eventos reenviados a través del *proxy* de interconexión en una base de datos ontológica en formato RDF. Concretamente, la plataforma *UniverSAAL* lo almacena en el historial de contexto *Entrepot* (Che, por sus siglas en inglés *Context History Entrepot*). La consulta a estos datos, se realiza utilizando el lenguaje de consultas *SPARQL* que es un lenguaje de consulta similar a *SQL* para consultar datos representados en formato *RDF*.

Los modelos se crearon y evaluaron en un servidor privado en el *cloud* usando la herramienta de análisis de datos BigML. En el *cloud* se ha utilizado un servidor FUJITSU equipado con 64 GB de memoria y una CPU Intel Xeon E3-1220v5 a 3.00 GHz.

6.4.4 Resultados y evaluaciones

6.4.4.1 Resultados de la detección de caídas

Para la fase pruebas y evaluación del desempeño del sistema propuesto, se realizaron un total de 54 experimentos controlados: 27 experimentos de caídas y 27 experimentos de ADL. Tres sujetos (voluntarios) entre las edades de 40 y 60 años, masa corporal de 68.7 a 84.6 kg, y altura de 1.64 a 1.79 m participaron en los experimentos. Para cumplir la ley de protección de datos GPRD (del inglés,

General Data Protection Regulation)[2] y asegurar el reconocimiento de los participantes, antes de estos experimentos, cada sujeto dio su consentimiento informado por escrito para participar en el experimento y utilizar la información extraída de los mismos para cumplir sus objetivos.

Cada sujeto realizó las caídas y las ADL descritas en la Tabla 6.1, y cada experimento se repitió tres veces. Por lo tanto, cada sujeto realizó un total de 18 simulaciones. El dispositivo portátil se colocó en la cintura de los voluntarios porque se considera el lugar adecuado (alternativa más ergonómica) en el cuerpo humano para obtener un rendimiento óptimo (Gjoreski *et al.*, 2011). Las caídas fueron simuladas sin ningún orden en particular. Inicialmente, se realizaron varias pruebas para ajustar los parámetros del sistema.

Los resultados obtenidos después de los experimentos se analizaron utilizando los parámetros estadísticos: precisión, exactitud, sensibilidad y especificidad (Parker, 2011).

$$Precision = \frac{VP+VN}{VP+VN+FP+FN} \quad (3) \qquad Exactitud = \frac{VP}{VP+FP} \quad (4)$$

$$Sensibilidad = \frac{VP}{VP+FN} \quad (5) \qquad Especificidad = \frac{VN}{VN+FP} \quad (6)$$

Estos parámetros se definen mediante verdaderos positivos (VP), verdaderos negativos (VN), falsos positivos (FP) y falsos negativos (FN) de la siguiente manera:

- VP se define como el éxito del sistema para detectar una caída o ADL cuando esto sucedió.
- VN se define como el éxito del sistema para detectar la ausencia de una caída o ADL cuando este no sucedió.
- FP es el fallo del sistema que detecta una caída o ADL cuando esta, no sucedió.
- FN es el fallo del sistema que no detecta una caída o ADL cuando esto sucedió.

[2] https://eur-lex.europa.eu/eli/reg/2016/679/oj

La Tabla 6.4 muestra los resultados alcanzados por el sistema IoTE-Fall al usar el algoritmo *ensemble* en la detección de caídas y ADLs.

Tabla 6.4: Resumen consolidado de los resultados alcanzados por el sistema IoTE-Fall en la detección de caídas

Clase Objetivo	Precisión	Exactitud	Sensibilidad	Especificidad
FF	99.80%	98.20%	100.00%	100.00%
BF	99.10%	92.40%	99.40%	99.32%
LF	99.70%	98.10%	99.40%	99.85%
WA	98.90%	98.20%	98.60%	99.50%
SCA	97.10%	93.60%	95.80%	98.34%
SA	97.70%	96.80%	74.40%	99.85%
Promedio	**98.72%**	**96.22%**	**94.60%**	**99.48%**

Los valores presentados en la Tabla 6.4 son los promedios de los resultados obtenidos para cada tipo de caída y actividad detectada. Se puede observar que la precisión de la detección de caídas y ADLs del sistema propuesto es del 98.72%. Dado que el porcentaje de error es ligeramente pequeño (1.28 %), las reglas generadas por el modelo pueden aplicarse a la clasificación de nuevos datos (Han et al., 2012) y justifican claramente la decisión de entrenar el modelo con el conocimiento histórico correspondiente a la audiencia prevista (es decir, la población adulta). Además, dado que el sistema está estrechamente vinculado a la asistencia sanitaria, la exactitud es muy importante para reducir las falsas alarmas. El sistema IoTE-Fall alcanza una exactitud en la detección de caídas de 96.23%, lo que indica que el sistema tiene un porcentaje bajo de emitir falsas alarmas de emergencia. Sin embargo, esto se podría gestionar, añadiendo al dispositivo portátil una alarma que se activé cuando el sistema detecte una caída, si está es una falsa alarma, la persona adulta puede presionar un botón para desactivarlo lo cual también sería notificado a todos los responsables del cuidado de la salud. Por otro lado, la sensibilidad promedio alcanzada en la detección de los diferentes tipos de caídas fue de 99.6%, lo que indica que el sistema IoTE-Fall propuesto reconoció casi todas las caídas. El sistema también logró una alta especificidad (99.23%) en la identificación de ADLs. Esto significa que hay un número relativamente pequeño de casos en los que el sistema detecta una actividad cuando la misma no se produjo.

6.4.4.2 Comparación con otros sistemas

Como se indica la sección 6.2, en los últimos años, se han realizado varios estudios de investigación para abordar la detección de caídas en personas mayores.

En comparación, con los sistemas WS que emplean algoritmos de aprendizaje automático para la detección de caídas (que son el enfoque de nuestra investigación). Nuestra propuesta es la única que aprovecha la eficacia del algoritmo de aprendizaje automático *ensemble*, logrando resultados prometedores en la detección de caídas en términos de precisión, sensibilidad y especificidad, como se muestra en la Tabla 6.5.

Tabla 6.5: Comparación de IoTE-Fall con otros trabajos relacionados

Sistema	Algoritmo ML	Precisión	Sensibilidad	Especificidad
D. Yacchirema et al., (2019)	*Ensemble*	98.72%	96.22%	94.60%
Aguiar et al., (2014)	*Decision tree*	92.9%	92.3%	93.2%
Santoyo-Ramón et al., (2018)	SVM	-	93.5%	97.8%
Pierleoni et al., (2015)	SVM	95.18%	90.37%	100%
Nguyen et al., (2018)	MLP	99.05%	98.26%	99.62%
Özdemir & Barshan, (2014)	K-NN	99.91%	100%	99.79%
Tong et al., (2013)	HMM	-	100%	100%

A pesar que el algoritmo *Decision Tree* utilizado en el sistema propuesto por Tong et al., (2013) presenta una sensibilidad (100%) y especificidad (100%) ideal, el conjunto de datos utilizados para validar el sistema no se corresponde a la audiencia objetivo. Por lo tanto, los resultados del sistema aplicado a las personas mayores serán diferentes en condiciones realistas. De hecho, los autores afirman que los experimentos del sistema con personas mayores, quedan para futuros trabajos. Además, este sistema no alerta en caso de una caída, que es un requisito necesario para evitar la *long-lie* y mejorar la calidad de vida de las personas mayores.

Cuando comparamos con el sistema propuesto por Özdemir & Barshan, (2014), obtenemos resultados similares, con una diferencia de 3.78 y 5.19% de sensibili-

dad y especificidad, respectivamente. Mientras que la precisión tiene una diferencia mínima de 1.19%. Sin embargo, al igual que Tong et al., (2013), los ensayos utilizados para probar el sistema corresponden a voluntarios jóvenes con edades entre 21.5 y 27 años. Por lo tanto, la comparación de estos parámetros de rendimiento podría estar sesgada hacia los enfoques anteriores. Además, el sistema muestra algunas desventajas para una implementación real debido a que el sistema utiliza seis unidades de sensores con tres dispositivos triaxiales. La mala usabilidad puede plantear barreras para adoptar esta solución. En contraste con la propuesta de Özdemir & Barshan, (2014), nuestro sistema utiliza solo los datos provenientes de un acelerómetro de eje 3D para la detección de caídas. Por lo tanto, nuestro sistema tiene menos datos para procesar y analizar, lo que lleva a una rápida detección de caídas y, por lo tanto, a reducir la ocurrencia de la *long-lie*.

Similar a nuestra propuesta, el sistema basado en el algoritmo MLP propuesto por Nguyen et al., (2018) utilizó el conjunto de datos SisFall para la detección de caídas. El sistema logró resultados prometedores con una especificidad del 99,62%. En este caso, la precisión es muy cercana al enfoque presentado aquí, con una diferencia insignificante de 0.33% a favor del sistema propuesto por Nguyen et al., (2018). Los autores utilizan el mismo conjunto de datos tanto para entrenar como probar el sistema. Sin embargo, el mismo conjunto de datos utilizado en todas las etapas del sistema podría causar un problema de *overfitting*, por lo que el sistema no podrá generalizar lo suficiente con los nuevos datos recopilados. Además, el detector de caídas no envía notificaciones cuando se produce una caída. Por el contrario, en nuestra propuesta aplicamos la técnica de validación cruzada 5-*fold* para evitar el problema del *overfitting*. Además, empleamos un conjunto de datos diferente tanto para la etapa de entrenamiento como de pruebas que se corresponde solo con la población objetivo.

Los resultados también revelan que IoTE-Fall basado en *ensemble* proporciona un mejor desempeño que los otros cuatro trabajos relacionados analizados (Y. Wang *et al.*, 2017) (Aguiar *et al.*, 2014) (Santoyo-Ramón *et al.*, 2018) and (Pierleoni *et al.*, 2015) en términos de precisión y sensibilidad, con una diferencia en la especificidad de 5,4% con la propuesta de Pierleoni et al., (2015). Sin embargo, también hay otras diferencias importantes entre estos sistemas propuestos e IoTE-Fall. Pierleoni et al., (2015) empleó tres sensores triaxiales integrados en un dispositivo portátil para la detección de caídas. La fusión de datos de estos sensores puede tener un impacto en la duración de la batería. Además, los autores realizaron dos extracciones para la detección de caídas: una para identificar

el pico y otra para detectar la orientación de la caída, lo que requiere más tiempo de procesamiento y, por lo tanto, puede no ser compatible con aplicaciones en tiempo real en las que se requiere que el procesamiento sea rápido con el objetivo de actuar en consecuencia ante la detección de una caída. Los sistemas propuestos por Santoyo-Ramón et al., (2018) y Aguiar et al., (2014) usan un teléfono inteligente como entorno de ejecución para el procesamiento de los datos de las caídas. El procesamiento en teléfonos inteligentes puede acortar la duración de la batería del dispositivo y evitar el funcionamiento prolongado del sistema. El sistema propuesto por Santoyo-Ramón et al., (2018), en este caso, tampoco incluyen ningún mecanismo de notificación de alertas de emergencia. IoTE-Fall, en comparación con el sistema propuesto por Aguiar et al., (2014) también ha demostrado que varios árboles de decisión aumentan la precisión del sistema en la detección de caídas gracias a la efectividad del voto de pluralidad.

Además, en comparación con todos los sistemas analizados en este estudio, la interoperabilidad semántica implementada en nuestro sistema facilita el intercambio de datos de caídas detectadas en un formato uniforme con cualquier sistema autorizado.

6.5 Conclusiones

El Internet de las cosas es un nuevo paradigma que puede ayudar a la población adulta a mejorar su calidad de vida al facilitar una forma de atención generalizada y más personalizada. En este capítulo, se ha presentado el diseño y evaluación del sistema IoTE-Fall para validar en un entorno real las bondades de la arquitectura de interoperabilidad propuesta en el capítulo 3.

Para la implementación del sistema se evaluó cuatro algoritmos (árboles de decisión*, ensamble*, regresión logística y *Deepnets*). Las validaciones han sido realizadas mediante la creación de modelos Big Data empleando el conocimiento histórico de caídas y ADLs de un conjunto de datos de acceso público y aplicando la validación cruzada *5-fold*. Estas evaluaciones se han centrado en medir el rendimiento, los requisitos computacionales y el área bajo la curva (AUC ROC). Los resultados revelan que el algoritmo *ensemble* presenta una tendencia dominante respecto a los otros algoritmos con un AUC promedio de 0.995, un tiempo de entrenamiento de 5.75 s, y un tiempo de validación de 3.48 s. Como consecuencia, el sistema IoTE-Fall se implementó utilizando el algoritmo *ensemble y* consta de un dispositivo portátil, una red de comunicaciones inalámbrica de baja potencia, un *smart IoT gateway* y un servidor *cloud*.

Varios experimentos controlados se llevaron a cabo para obtener los datos del movimiento de las personas mayores. Estos valores se recopilan con un sensor acelerómetro 3D integrado en un dispositivo portátil basado en 6LowPAN. El dispositivo se colocó en la cintura de las personas mayores y ofrece una solución adecuada para ser utilizado por una persona mayor en un ambiente interior.

IoTE-Fall realiza el procesamiento de detección de caídas en el *smart IoT gateway* que proporciona varios servicios *fog* (interoperabilidad, transformación de datos, almacenamiento local, entrega de notificaciones, y representación semántica) complementarios al borde de la red para ofrecer una solución IoT completa. IoTE-Fall alerta de forma remota a los profesionales de la salud, a los centros de emergencia, a los cuidadores y a los familiares de las personas mayores en caso de que ocurra un evento de caída utilizando mecanismos de QoS.

El rendimiento del sistema propuesto se evaluó para reconocer tres tipos de caídas: caída hacia adelante, caída hacia atrás y caída lateral, y tres tipos de ADL: caminar, subir escaleras y sentarse. La precisión promedio de reconocimiento (98.72%), la exactitud (96.22%), la sensibilidad (94.60%) y la especificidad (99.48%) demostraron que el sistema propuesto tiene una alta tasa de éxito, tanto en la detección de caídas como en detección de ADL.

También se presentó un análisis comparativo que revela las ventajas de IoTE-Fall sobre otras propuestas de investigación de la misma naturaleza. Estas ventajas implican no solo altos niveles de precisión, exactitud, sensibilidad y especificidad, iguales o superiores a los resultados obtenidos por otros sistemas, sino también un valor agregado adicional con los diferentes servicios *fog* incluidos en el *smart IoT gateway*. Por ejemplo, el servicio de interoperabilidad semántica proporciona una solución de IoT holística e interoperable y permite que el sistema IoTE-Fall coexista y se integre con otras aplicaciones o servicios externos. También, el servicio del sistema de alarma ayuda a los profesionales de la salud a tomar decisiones efectivas y en el tiempo, informando sobre el tipo de caída y la ubicación de la casa de la persona mayor. Además, gracias al procesamiento de los datos del movimiento de las personas mayores en el borde de la red, nuestra solución también podría ayudar a evitar la *long-lie* y, en consecuencia, mejorar la calidad de vida de las personas mayores. Estos servicios no están contemplados en muchos de los estudios analizados.

Por último, el sistema IoTE-Fall permite el uso de tecnologías emergentes como 6LowPAN.

Capítulo 7

Conclusiones y líneas de trabajo futuras

« Los momentos finales de una experiencia determinan el recuerdo que conservaremos de la misma.

»

Daniel Kahneman (1934)

En esta tesis se han identificado y estudiado los principales problemas que surgen debido a la heterogeneidad de los diferentes dispositivos en IoT. Tras ello, se ha diseñado, implementado y evaluado una arquitectura para la interoperabilidad de dispositivos físicos en IoT y su integración con plataformas IoT estándar que trata de solventar el problema de la interoperabilidad a nivel técnico y, sintáctico. La arquitectura se ha puesto a prueba en dos casos de uso y escenarios diferentes.

A continuación, se presentan las principales conclusiones que se han obtenido a lo largo de la investigación derivadas de cada uno de los capítulos precedentes. Los cuales, a su vez, están basados en las publicaciones científicas logradas como productos de esta investigación. Además, se destaca el futuro trabajo que se puede desarrollar a partir de esta investigación.

7.1 Conclusiones generales

- IoT es una realidad que se proyecta con mayor potencial de crecimiento para los próximos años, ya que como red omnipresente permite una visión que implica un ecosistema global hiperconectado de una gran cantidad de dispositivos IoT identificables; equipados con capacidades de detección, computación, activación y comunicación, que actualmente ya superan en gran medida a los usuarios de la Web.

- La interoperabilidad en IoT es uno de los problemas más acuciantes y una necesidad de primer orden para la adopción de esta tecnología. El valor de IoT reside en los datos, y es a través de la interconexión de dispositivos heterogéneos de distinta naturaleza y perfil técnico que se alcanzará este valor, permitiendo el desarrollo de aplicaciones disruptivas gracias a la agregación de datos recopilados de diferentes dispositivos.

- Se ha realizado una extensa revisión bibliográfica de las características esenciales de IoT, los diferentes componentes y los dominios de aplicación. Además, se ha destacado las arquitecturas de referencia más representativas para el despliegue de las aplicaciones IoT en entornos reales. También se explican los diferentes tipos de interoperabilidad en IoT, considerando los patrones de comunicación, y detallando las tecnologías de conectividad más representativas y protocolos IoT para proponer una nueva solución a la interoperabilidad de dispositivos.

- Las contribuciones de esta tesis se abordan a través de dos ramas de investigación interrelacionadas que pueden considerarse conjuntamente como una solución que facilita el desarrollo de soluciones IoT.

    - La arquitectura de interoperabilidad de dispositivos en IoT, en el capítulo 3.
    - La arquitectura para la integración de dispositivos IoT con plataformas IoT estándar, en el capítulo 5.

- Para el diseño de la arquitectura de interoperabilidad de dispositivos se ha considerado el modelo de referencia arquitectónico IoT-ARM y la arquitectura funcional M2M.La arquitectura de interoperabilidad de dispositivos en IoT aborda el problema de la heterogeneidad de los protocolos de comunicación y del formato de datos que caracterizan a los dispositivos IoT. Si bien, actualmente algunas tecnologías de IoT han alcanzado cierto grado de madurez y estandarización, todavía hay una alta fragmentación en los protocolos de comunicación.

- La arquitectura se puede implementar como un *smart IoT gateway* que implementa diferentes adaptadores que envuelven controladores, *bridges* y protocolos IoT para comunicarse con redes de sensores cableadas e inalámbricas basadas en diferentes tecnologías para permitir la interoperabilidad técnica. Las comunicaciones entre dispositivos y el *smart IoT gateway* se realiza a través de protocolos IoT estándar, abiertos y livianos.

- La arquitectura utiliza un formato de datos común basada en una estructura de atributos y metadatos para permitir el intercambio de la información entre dispositivos que contribuye a garantizar la interoperabilidad sintáctica.

- La arquitectura provee servicios locales de procesamiento, almacenamiento y análisis de datos para permitir el desarrollo de servicios y aplicaciones IoT que explotan la interoperabilidad técnica y sintáctica para aplicarse a diferentes entornos y dominios de IoT.

- La funcionalidad de la arquitectura se valida a través de dos casos de estudio enfocados a los dominios AAL, AHA y uno al transporte y la logística. En el primer caso de estudio se ha explotado las bondades del *smart IoT gateway* para implementar un sistema prototipo de monitorización que promueve la vida independiente de las personas mayores.

- Se han realizado mediciones del uso de recursos y se ha verificado que la comunicación entre dispositivos, así como que el intercambio y uso de información entre ellos, a través del *smart IoT gateway* se realice sin fisuras independiente de los protocolos de comunicación y formato de datos subyacentes. Los resultados favorables demuestran que el sistema tiene un rendimiento satisfactorio. Además, validan la arquitectura y confirman su viabilidad.

- Se ha presentado una arquitectura de interconexión para la integración de dispositivos con plataformas IoT estándar, en el que los dispositivos IoT conectados al *smart IoT gateway* pueden interactuar con servicios y aplicaciones externas a través de un *proxy* de interconexión.

- Se ha diseñado un escenario de pruebas que consiste en representar virtualmente un camión como uno de los principales activos para controlar el acceso, supervisar el tráfico y ayudar a las operaciones en el puerto. A lo largo del escenario propuesto, el camión intercambia información con varias plataformas IoT portuarias a través de la arquitectura de interconexión. Cada una de ellas, se subscribe al servidor OM2M compatible con el estándar oneM2M para recibir los datos y para controlar el camión. En el escenario de pruebas también se ha utilizado el habilitador genérico *Orion* de la plataforma FIWARE, a través del cual también se ha verificado la funcionalidad de la arquitectura de interconexión.

- Se ha diseñado un sistema basado en IoT y Big data para detectar las caídas en las personas mayores. Una de las funcionalidades de especial interés implementadas es el análisis de datos al borde de la red basado en *fog computing*. Gracias a la combinación de *fog computing y cloud computing*, el sistema es capaz de crear modelos predictivos en el *cloud* y realizar en tiempo real predicciones de caídas y enviar notificaciones de emergencia a los profesionales de la salud, desde el borde de la red.

- Para seleccionar el mejor algoritmo para la detección de caídas se ha validado varios algoritmos de aprendizaje empleando la técnica *cross-validation* en función de tres parámetros: área debajo de la curva *ROC*, tiempo de entrenamiento y tiempo de validación. Los resultados demuestran que *ensemble* es el algoritmo más adecuado para la detección de caídas. El modelo predictivo basado en *ensemble* ha sido configurado para construir 10 árboles de decisión que en función del voto mayoritario determinan la predicción del tipo de caída.

- Se ha desarrollado un sistema prototipo que consiste de un dispositivo portátil, una red de comunicación inalámbrica, el *smart IoT gateway* donde se implementa el análisis de datos y un servidor *cloud*. Se ha extendido la ontología *device* de la plataforma *UniverSAAL* para modelar los eventos de caídas detectados.

- El sistema se ha validado a través de varios experimentos con voluntarios adultos en términos de precisión, sensibilidad y especificidad. Se ha realizado una evaluación comparativa con trabajos relacionados. Los resul-

tados favorables revelan que el sistema proporciona un mejor desempeño en términos de precisión y sensibilidad. No obstante, presenta una especificidad inferior alrededor del 5% con uno de los trabajos relacionados.

7.2 Trabajos futuros

A lo largo de este trabajo, es posible ampliar el estudio realizado en esta investigación en múltiples direcciones. A continuación, se describen algunas ideas que son consideradas interesantes para posteriores investigaciones, tanto en el ámbito de la interoperabilidad, como en los casos de uso presentados.

- Extender las funcionalidades de conectividad de la arquitectura: Las implementaciones actuales del *smart IoT gateway* incluyen adaptadores para los estándares y protocolos más populares y actuales en IoT que son muy efectivos para implementar redes inalámbricas de corto (ZigBee, BLE, 6LoWPAN, Wi-Fi) y largo alcance (LoRA), y redes cableadas (Ethernet). Las futuras aplicaciones de IoT (p ej., aplicaciones domésticas e industriales) necesitarán características de conectividad como movilidad, ancho de banda mejorado, aumento del alcance del enlace inalámbrico y seguridad de mensajes extremos a extremo. Estas características se lograrán mediante protocolos como Mesh, Thread y NFC. Además, la posibilidad de implementar estas tecnologías en dispositivos restrictivos será sin duda la mejor opción para las futuras aplicaciones de IoT. Se deben desarrollar adaptadores para admitir estos protocolos para implementaciones futuras del *smart IoT gateway*.

- Agregar técnicas de compresión de datos: ningún módulo que proporcione capacidades de compresión de datos se incluyó como componente en la arquitectura del *smart IoT gateway*. Las futuras implementaciones del *smart IoT gateway* incluirán esta capacidad, ya que la compresión de datos ayuda a ahorrar ancho de banda de red y la energía consumida durante la transmisión de datos. Las capacidades de compresión se alcanzarán mediante técnicas de compresión de datos con pérdida o sin pérdida ampliamente utilizados en IoT dependiendo del dominio de aplicación. Siendo, la compresión de datos con pérdida la opción más adecuada para la implementación en dispositivos con recursos limitados debido a que requiere menos potencia de procesamiento que la técnica de compresión

de datos sin pérdida. Se debe desarrollar e integrar este módulo para implementaciones futuras del *smart IoT gateway*.

- Agregar nuevas técnicas de transformación de datos: Debido a que la presente tesis doctoral se ha desarrollado con el financiamiento del Estado ecuatoriano, a través de una beca de la Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación (SENESCYT), dentro del futuro inmediato, se tiene proyectado permitir el control de los dispositivos a través de interfaces como voz o gestos, para implementar aplicaciones AHA reales que sean de beneficio para la sociedad ecuatoriana en especial para las personas con discapacidad de movilidad y habla. Las capacidades de transformación se alcanzarán mediante la implementación de técnicas de transformación de comandos de voz o gestos en peticiones publish/subscribe o REST y el uso de dispositivos de asistencia domiciliaría como Amazon Echo y Alexa. Se debe desarrollar e integrar esta funcionalidad en el módulo de transformación de datos para implementaciones futuras del *smart IoT gateway*.

- Implementar la arquitectura en otros dispositivos SoC: Las implementaciones actuales del *smart IoT gateway* han sido ejecutadas en dispositivos *Raspberry Pi 2 y 3 modelo B* que son uno de los SoC más populares entre los investigadores para la implementación de soluciones IoT debido a su versatilidad y bajo costo. Al extender la conectividad del *smart IoT gateway* las próximas implementaciones del mismo se ejecutarán sobre dispositivos SoC que integren múltiples capacidades de conexión en una misma placa de desarrollo. Un ejemplo representativo es Fipy(PyCOM, 2019) que habilita interfaces de comunicación para brindar acceso a varias redes LowPAN (LPWAN) como Wi-Fi, LoRa, SigFOX, LTE Celular y BLE. Otro SoC que aporta flexibilidad de diseño para ejecutar la funcionalidad del *smart IoT gateway* es Nordic de la serie 455-00001 que integra el módulo BL654 (Laird, 2018) para brindar el acceso a varias redes como Bluetooth Low Energy (BLE) 5.0, Thread (802.15.4) y NFC.

- Al término de la redacción de esta tesis, fue aprobado un grupo de investigación multidisciplinario denominado Analítica de datos e inteligencia artificial aplicado a la ciberseguridad[3], aprobado por la Corporación Ecuatoriana para el Desarrollo de la Investigación y la Academia (CEDIA). El grupo está constituido por docentes de la EPN y de cuatro universidades del Ecuador (ESPE, UNACH, UCE y UC), siendo uno de sus objetivos específicos, el desarrollar una arquitectura de *BigData* como servicio (BDaaS) que pueda ser utilizada bajo demanda por varias aplicaciones, servicios y/o plataformas de las IES del Ecuador que requieran analizar los datos sin tener que incurrir en inversiones de dinero. Para las fases de prueba y validación, se tomará como fuente de datos la arquitectura propuesta en la presente tesis.

Casos de uso

- Sistema de detección de caídas: Consolidar la línea de investigación en AHA y presentar proyectos de investigación aplicada en el marco del Concurso Ecuatoriano de Proyectos I+D+i, CEPRA XIII para desplegar el caso de uso en un entorno real. En la práctica, las caídas se pueden producir por varios factores como un desmayo, frecuencia cardíaca alta, etc. Por lo tanto, para el trabajo futuro se debe avanzar hacia la integración de más sensores de salud, como sensores de ritmo cardíaco, sensor de presión arterial entre otros, así como también hacia la utilización de algoritmos ML no supervisados que permitan establecer asociaciones entre las caídas y los datos de los nuevos sensores integrados, para facilitar aún más la toma de decisiones médicas.

- Transporte y logística: Aprovechar las ontologías y el procesamiento de datos semántico de los desarrollos del proyecto INTER-IoT a fin de apoyar la interoperabilidad a distintos niveles. Además, partiendo de que la posición del camión será monitorizada una vez que se encuentre en las instalaciones portuarias, la integración de un sistema de alertas de emergencia permitiría detectar un accidente o un problema médico con los datos provenientes del camión y del conductor respectivamente, y alertar a las autoridades portuarias.

[3] https://www.cedia.edu.ec/es/grupos-de-trabajo-conformados/ii-convocatoria/analitica-de-datos-e-inteligencia-artificial-aplicado-a-la-ciberseguridad

Bibliografía

Adafruit Industries (2018) *Adafruit_CircuitPython_INA219*. Retrieved from https://github.com/adafruit/Adafruit_CircuitPython_INA219

Adafruit Industries (2019) *Adafruit INA219 Current Sensor Breakout*. Retrieved from https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ina219-current-sensor-breakout.pdf

Agile IoT Project (2017) Agile (Adaptive Gateways for dIverse muLtiple Environments). Retrieved January 1, 2019, from http://agile-iot.eu/about/

Aguiar, B., Rocha, T., Silva, J., *et al.* (2014) Accelerometer-based fall detection for smartphones. In: *2014 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, pp. 1–6. doi:10.1109/MeMeA.2014.6860110

Ahlgren, B., Hidell, M., & Ngai, E. C.-. (2016) Internet of Things for Smart Cities: Interoperability and Open Data. *IEEE Internet Computing*, 20, 52–56. doi:10.1109/MIC.2016.124

Ahmed, N., Rahman, H., & Hussain, M. I. (2016) A comparison of 802.11ah and 802.15.4 for IoT. *ICT Express*, 2, 100–102. doi:https://doi.org/10.1016/j.icte.2016.07.003

Ainsley, A. (2018) *Python Learn Python Programming Language From The Scratch*. USA: CreateSpace Independent Publishing Platform.

Akeela, R., & Elziq, Y. (2017) Design and verification of IEEE 802.11ah for IoT and M2M applications. In: *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 491–496. doi:10.1109/PERCOMW.2017.7917612

Al-Fuqaha, A., Guizani, M., Mohammadi, M., *et al.* (2015a) Internet of Things: A Survey on Enabling Technologies, Protocols and Applications. *IEEE Communications Surveys & Tutorials*, 1–1. doi:10.1109/COMST.2015.2444095

Al-Fuqaha, A., Guizani, M., Mohammadi, M., *et al.* (2015b) Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys and Tutorials*, 17, 2347–2376. doi:10.1109/COMST.2015.2444095

Alliance, L. (2015) A technical overview of LoRa and LoRaWAN. *White Paper*.

Alliance for Internet of Things Innovation (2018) High Level Architecture (HAL) Release 4.0. Retrieved from https://aioti.eu/wp-content/uploads/2018/06/AIOTI-HLA-R4.0.7.1-Final.pdf

Alpaydin, E. (2009) *Introduction to machine learning*. MIT press.

Álvarez de la Concepción, M. Á., Soria Morillo, L. M., Álvarez García, J. A., *et al.* (2017) Mobile activity recognition and fall detection system for elderly people using Ameva algorithm. *Pervasive and Mobile Computing*, 34, 3–13. doi:https://doi.org/10.1016/j.pmcj.2016.05.002

Aruba Networks (2017) IoT Heading for Mass Adoption by 2019 Driven by Better-Than-Expected Business Results. Retrieved June 5, 2018, from https://goo.gl/22UZ8e

ATIS (2015) ATIS Telecom Glossary. Retrieved August 12, 2017, from http://www.atis.org/glossary/definition.aspx?id=4292

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

Aziz, O., Klenk, J., Schwickert, L., *et al.* (2017) Validation of accuracy of SVM-based fall detection system using real-world fall and non-fall datasets. *PLoS One*, 12, e0180318.

Aziz, O., Musngi, M., Park, E. J., *et al.* (2017) A comparison of accuracy of fall detection algorithms (threshold-based vs. machine learning) using waist-mounted tri-axial accelerometer signals from a comprehensive set of falls and non-fall trials. *Medical & Biological Engineering & Computing*, 55, 45–55. doi:10.1007/s11517-016-1504-y

Baccelli, E., Hahm, O., Gunes, M., *et al.* (2013) RIOT OS: Towards an OS for the Internet of Things. In: *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 79–80. doi:10.1109/INFCOMW.2013.6970748

Bahga, A., & Madisetti, V. (2016) *Big data science & analytics: A hands-on approach*. VPT.

Baker, N. (2005) ZigBee and Bluetooth strengths and weaknesses for industrial applications. *Computing & Control Engineering Journal*, 16, 20–25. doi:10.1049/cce:20050204

Banh, M., Mac, H., Nguyen, N., *et al.* (2015) Performance evaluation of multiple RPL routing tree instances for Internet of Things applications. In: *2015 International Conference on Advanced Technologies for Communications (ATC)*, pp. 206–211. doi:10.1109/ATC.2015.7388321

Baronti, P., Pillai, P., Chook, V. W. C., *et al.* (2007) Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards. *Computer Communications*, 30, 1655–1695. doi:https://doi.org/10.1016/j.comcom.2006.12.020

Bauer, M., Boussard, M., Bui, N., *et al.* (2013) *Internet of Things – Architecture IoT-A Deliverable D1.5 – Final architectural reference model for the IoT v3.0*.

Bauer, M., Bui, N., De Loof, J., *et al.* (2013) IoT Reference Model BT - Enabling Things to Talk: Designing IoT solutions with the IoT Architectural Reference Model. In: (eds Bassi, A., Bauer, M., Fiedler, M., et al.), pp. 113–162. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-40403-0_7

Bello, O., Zeadally, S., & Badra, M. (2017) Network layer inter-operation of Device-to-Device communication technologies in Internet of Things (IoT). *Ad Hoc Networks*, 57, 52–62. doi:https://doi.org/10.1016/j.adhoc.2016.06.010

Bhavsar, H., & Ganatra, A. (2012) A comparative study of training algorithms for supervised machine learning. *International Journal of Soft Computing and Engineering (IJSCE)*, 2, 2231–2307.

BigML (2017) Comprehensive Machine Learning Platform. Retrieved August 12, 2018, from https://bigml.com/features

Bimschas, D., Hellbrück, H., & Mietz, R. (2010) Middleware for smart gateways connecting sensornets to the internet. In: *Proceedings of the 5th International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks. ACM.*, pp. 8–14. doi:10.1145/1890784.1890787

Bluetooth (2019) Bluetooth SIG Bluetooth Core Specification 5.0. Retrieved October 10, 2018, from https://www.bluetooth.com/specifications

BLuetooth-SIG (2018) Secure Gateway Kit. Retrieved October 12, 2018, from https://www.bluetooth.com/develop-with-bluetooth/build/developer-kits/secure-gateway-kit

Bluetooth, S. I. G. (2003) Bluetooth specification.

BLueZ Project (2016) Bluez. Retrieved February 18, 2019, from http://www.bluez.org/

Borgia, E. (2014) The Internet of Things vision: Key features, applications and open issues. *Computer Communications*, 54, 1–31. doi:https://doi.org/10.1016/j.comcom.2014.09.008

Bormann, C., Bremen, U., & Sensinode, Z. S. (2012) CoAP : An Application Protocol for Billions of Tiny Internet Nodes.

Bormann, C., Ersue, M., & Keranen, A. (2014) Terminology for Constrained-Node Networks RFC 7228. Retrieved August 12, 2016, from https://tools.ietf.org/html/rfc7228#section-2.3.2

Bousquet, J., Kuh, D., Bewick, M., *et al.* (2015) Operational definition of Active and Healthy Ageing (AHA): A conceptual framework. *The Journal of Nutrition, Health {&} Aging*, 19, 955–960. doi:10.1007/s12603-015-0589-6

Cao, Q., Abdelzaher, T., Stankovic, J., *et al.* (2008) The LiteOS Operating System: Towards Unix-Like Abstractions for Wireless Sensor Networks. In: *2008 International Conference on Information Processing in Sensor Networks (Ipsn 2008)*, pp. 233–244. doi:10.1109/IPSN.2008.54

Caro, N. De, Colitti, W., Steenhaut, K., *et al.* (2013) Comparison of two lightweight protocols for smartphone-based sensing. In: *2013 IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT)*, pp. 1–6. doi:10.1109/SCVT.2013.6735994

CASAGRAS (2009) *Final Report: RFID and the Inclusive Model for the Internet of Things*.

Castellani, A., Loreto, S., Rahman, A., *et al.* (2011) *Best practices for HTTP-CoAP mapping implementation draft-castellani-core-http-mapping-02*. Retrieved from https://tools.ietf.org/html/draft-castellani-core-http-mapping-02

Cerina, L., Notargiacomo, S., Paccanit, M. G., *et al.* (2017) A fog-computing architecture for preventive healthcare and assisted living in smart ambients. In: *2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)*, pp. 1–6. doi:10.1109/RTSI.2017.8065939

CERP-IoT Cluster of European Research Projects on the Internet of Things (2010) Vision and challenges for Realising the Internet of Things. Retrieved from http://www.internet-of-things-research.eu/pdf/IoT_Clusterbook_March_2010.pdf

Chaccour, K., Darazi, R., Hassani, A. H. El, *et al.* (2017) From Fall Detection to Fall Prevention: A Generic Classification of Fall-Related Systems. *IEEE Sensors Journal*, 17, 812–822. doi:10.1109/JSEN.2016.2628099

Chang, V. (2014) The business intelligence as a service in the cloud. *Future Generation Computer Systems*, 37, 512–534.

Chen, S., Xu, H., Liu, D., *et al.* (2014) A Vision of IoT: Applications, Challenges, and Opportunities With China Perspective. *IEEE Internet of Things Journal*, 1, 349–359. doi:10.1109/JIOT.2014.2337336

Chowdhury, A. H., Ikram, M., Cha, H.-S., *et al.* (2009) Route-over vs Mesh-under Routing in 6LoWPAN. In: *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, pp. 1208–1212. ACM.

Colitti, W., Steenhaut, K., Caro, N. De, *et al.* (2011) Evaluation of constrained application protocol for wireless sensor networks. In: *2011 18th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN)*, pp. 1–6. doi:10.1109/LANMAN.2011.6076934

Compton, M., Barnaghi, P., Bermudez, L., *et al.* (2012) The SSN ontology of the W3C semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17, 25–32.

Corradi, A., Fanelli, M., & Foschini, L. (2014) VM consolidation: A real case based on OpenStack Cloud. *Future Generation Computer Systems*, 32, 118–127.

Crosby, G. V, & Vafa, F. (2013) Wireless sensor networks and LTE-A network convergence. In: *38th Annual IEEE Conference on Local Computer Networks*, pp. 731–734. IEEE.

Culler, D., & Chakrabarti, S. (2009) 6LoWPAN: Incorporating IEEE 802.15. 4 into the IP architecture. *White Paper*.

Dai, J., Bai, X., Yang, Z., *et al.* (2010) PerFallD: A pervasive fall detection system using mobile phones. In: *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pp. 292–297. doi:10.1109/PERCOMW.2010.5470652

Darroudi, M. S., & Gomez, C. (2017) Bluetooth Low Energy Mesh Networks: A Survey. *Sensors* . doi:10.3390/s17071467

Delgado, J. (2013) Service Interoperability in the Internet of Things. In: *Internet of Things and Inter-Cooperative Computational Technologies for Collective Intelligence* (eds Bessis, N., Xhafa, F., Varvarigou, D., et al.), pp. 51–87. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-34952-2_3

Department of Defense of United States of America (1988) *Levels of Information Systems Interoperability (LISI)*. Retrieved from http://web.cse.msstate.edu/~hamilton/DODAF/LISI.pdf

Desai, P., Sheth, A., & Anantharam, P. (2015) Semantic Gateway as a Service Architecture for IoT Interoperability. In: *2015 IEEE International Conference on Mobile Services*, pp. 313–319. doi:10.1109/MobServ.2015.51

Diallo, S. Y., Herencia-zapana, H., Padilla, J. J., *et al.* (2011) Understanding Interoperability, 84–91.

Dias, P. V. G. F., Costa, E. D. M., Tcheou, M. P., *et al.* (2016) Fall detection monitoring system with position detection for elderly at indoor environments under supervision. In: *2016 8th IEEE Latin-American Conference on Communications (LATINCOM)*, pp. 1–6. doi:10.1109/LATINCOM.2016.7811576

Dunkels, A., Gronvall, B., & Voigt, T. (2004) Contiki - a lightweight and flexible operating system for tiny networked sensors. In: *29th Annual IEEE International Conference on Local Computer Networks*, pp. 455–462. doi:10.1109/LCN.2004.38

Eclipse Foundation (2016) OM2M. Retrieved March 11, 2019, from https://wiki.eclipse.org/OM2M/Download

Ecma Internacional (2017) *Standard ECMA-404: The JSON Data Interchange Syntax*. Retrieved from https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

EsperTech (2018) Esper EPL reference. Retrieved August 12, 2018, from http://esper.espertech.com/release-5.2.0/esper-reference/html/epl_clauses.html

European Telecommunications Standards Institute (2011) *Technical Specification -Machine-to-Machine communications (M2M); Functional architecture V1.1.1, 102 690*. Retrieved from https://www.etsi.org/deliver/etsi_ts/102600_102699/102690/01.01.01_60/ts_102690v010101p.pdf

Evans, D. (2011) *The Internet of Things: How the Next Evolution of the Internet is Changing Everything. Cisco Internet Business Solutions Group (IBSG)*, Vol. 1.

Fan, Y., Levine, M. D., Wen, G., *et al.* (2017) A deep neural network for real-time detection of falling humans in naturally occurring scenes. *Neurocomputing*, 260, 43–58. doi:https://doi.org/10.1016/j.neucom.2017.02.082

FIESTA-IoT (2015) Federated Interoperable Semantic IoT Testbeds and Applications. Retrieved January 1, 2019, from http://fiesta-iot.eu/

FIWARE (2017) FIWARE-Cepheus CEP. Retrieved August 12, 2017, from http://fiware-cepheus.readthedocs.io/en/latest/cep/index.html

Fortino, G., & Gravina, R. (2015) Fall-MobileGuard: A Smart Real-time Fall Detection System. In: *Proceedings of the 10th EAI International Conference on Body Area Networks*, pp. 44–50. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). doi:10.4108/eai.28-9-2015.2261462

Fortino, G., Guerrieri, A., & Russo, W. (2012) Agent-oriented smart objects development. In: *Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 907–912. doi:10.1109/CSCWD.2012.6221929

Fortino, G., Parisi, D., Pirrone, V., *et al.* (2014) BodyCloud: A SaaS approach for community Body Sensor Networks. *Future Generation Computer Systems*, 35, 62–79. doi:https://doi.org/10.1016/j.future.2013.12.015

Ganzha, M., Paprzycki, M., Pawłowski, W., *et al.* (2017) Semantic interoperability in the Internet of Things: an overview from the INTER-IoT perspective. *Journal of Network and Computer Applications*, 81, 111–124.

Ganzha, M., Paprzycki, M., Pawłowski, W., *et al.* (2018) Towards Semantic Interoperability Between Internet of Things Platforms BT - Integration, Interconnection, and Interoperability of IoT Systems. In: (eds Gravina, R., Palau, C. E., Manso, M., et al.), pp. 103–127. Cham: Springer International Publishing. doi:10.1007/978-3-319-61300-0_6

Gigli, M., & Koo, S. (2011) Services and Applications Categorization. *Advances in Internet of Things*, 1, 27–31. doi:10.4236/ait.2011.12004

Gjoreski, H., Lustrek, M., & Gams, M. (2011) Accelerometer Placement for Posture Recognition and Fall Detection. In: *2011 Seventh International Conference on Intelligent Environments*, pp. 47–54. doi:10.1109/IE.2011.11

Gonzalez-Usach, R., Yacchirema, D., Julian, M., *et al.* (2019) Interoperability in IoT. In: *Handbook of Research on Big Data and the IoT*, pp. 149–173. IGI Global.

Gruber, T. R. (1993) A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5, 199–220. doi:https://doi.org/10.1006/knac.1993.1008

Gubbi, J., Buyya, R., Marusic, S., *et al.* (2013a) Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29, 1645–1660. doi:https://doi.org/10.1016/j.future.2013.01.010

Gubbi, J., Buyya, R., Marusic, S., *et al.* (2013b) Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29, 1645–1660. doi:10.1016/j.future.2013.01.010

Guoqiang, S., Yanming, C., Chao, Z., *et al.* (2013) Design and implementation of a smart IoT gateway. In: *Green Computing and Communications (GreenCom), IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, pp. 720–723. doi:10.1109/GreenCom-iThings-CPSCom.2013.130

H. Tschofenig, J. Arkko, D. Thaler, D. M. (2015) *Architectural Considerations in Smart Object Networking - RFC 7452*. Retrieved from https://tools.ietf.org/pdf/rfc7452.pdf

Haller, S. (2009) Internet of Things: An Integral Part of the Future Internet.

Hanke, S., Mayer, C., Hoeftberger, O., *et al.* (2011) universAAL -- An Open and Consolidated AAL Platform. In: *Ambient Assisted Living: 4. AAL-Kongress 2011, Berlin, Germany, January 25--26, 2011* (eds Wichert, R., & Eberhardt, B.), pp. 127–140. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-18167-2_10

Harden, W. (2017) Inteligent Transportation Systems. Retrieved October 20, 2018, from http://www.walkerharden.com/intelligent-transportation/

He, J., Bai, S., & Wang, X. (2017) An Unobtrusive Fall Detection and Alerting System Based on Kalman Filter and Bayes Network Classifier. *Sensors* . doi:10.3390/s17061393

He, W., Goodkind, D., & Kowal, P. (2016) *An Aging World : 2015 International Population Reports*. *Aging*. doi:P95/09-1

Holler, J., Tsiatsis, V., Mulligan, C., *et al.* (2014) *From Machine-to-machine to the Internet of Things: Introduction to a New Age of Intelligence*. Academic Press, Inc.

Hu, F., Xie, D., & Shen, S. (2013) On the Application of the Internet of Things in the Field of Medical and Health Care. In: *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pp. 2053–2058. doi:10.1109/GreenCom-iThings-CPSCom.2013.384

Hui, J. W., & Culler, D. E. (2008) Extending IP to Low-Power, Wireless Personal Area Networks. *IEEE Internet Computing*, 12, 37–45. doi:10.1109/MIC.2008.79

IEEE Architecture Working Group (2000) *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems:IEEE Std 1471-2000*.

IERC (2011) *IoT Semantic Interoperability:Research Challenges, Best Practices, Recommendations and Next Steps*. Retrieved from http://www.internet-of-things-research.eu/pdf/IERC_Position_Paper_IoT_Semantic_Interoperability_Final.pdf

IETF (2012) *Internet Research Task Force*. Retrieved from https://tools.ietf.org/pdf/draft-lee-iot-problem-statement-05.pdf

Institute of Electrical and Electronics Engineers, Inc., I. S. 802. 15. .-2003 (2003) *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)*. New York.

Inter-IoT (2018) *Interoperability of Heterogeneous IoT Platforms*. Retrieved from http://www.inter-iot-project.eu/wp-content/uploads/2016/02/D2_1_INTER-IoT_Stakeholders-and-market-analysis-report-final-v1.1.pdf

InterIoT (2016) INTER-IoT. Retrieved June 1, 2017, from http://www.inter-iot-project.eu/

International Business Machines Corporation (IBM) (2010) *MQ Telemetry Transport (MQTT) V3.1 Protocol Specification*.

International Organization for Standardization (ISO) (2015) ISO/IEC 2382-1:1993 Information Technology – Vocabulary – Part 1: Fundamental terms. Retrieved from https://www.iso.org/standard/7229.html

International Telecommunication Union (2012) *Next Generation Networks - Frameworks and functional architecture models*. Retrieved from https://www.itu.int/rec/T-REC-Y.2060-201206-I

Internet Engineering Task Force (IETF) (2011) *Request for Comments: 6282 - Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*. Retrieved from https://tools.ietf.org/html/rfc6282

Islam, S. M. R., Kwak, D., Kabir, M. H., *et al.* (2015) The Internet of Things for Health Care: A Comprehensive Survey. *IEEE Access*, 3, 678–708. doi:10.1109/ACCESS.2015.2437951

ITU-T (2018) *Recommendation ITU (T Y.4418): Gateway functional architecture for Internet of things applications*.

Jankowski, S., Szymański, Z., Dziomin, U., *et al.* (2015) Deep learning classifier for fall detection based on IR distance sensor data. In: *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Vol. 2, pp. 723–727. doi:10.1109/IDAACS.2015.7341398

Jokanović, B., & Amin, M. (2018) Fall Detection Using Deep Learning in Range-Doppler Radars. *IEEE Transactions on Aerospace and Electronic Systems*, 54, 180–189. doi:10.1109/TAES.2017.2740098

Jokanovic, B., Amin, M., & Ahmad, F. (2016) Radar fall motion detection using deep learning. In: *2016 IEEE Radar Conference (RadarConf)*, pp. 1–6. doi:10.1109/RADAR.2016.7485147

JS Foundation (2017) Node-RED. Retrieved May 1, 2017, from https://nodered.org/

Kangas, M., Konttila, A., Winblad, I., *et al.* (2007) Determination of simple thresholds for accelerometry-based parameters for fall detection. In: *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 1367–1370. doi:10.1109/IEMBS.2007.4352552

Karantonis, D. M., Narayanan, M. R., Mathie, M., *et al.* (2006) Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. *IEEE Transactions on Information Technology in Biomedicine*, 10, 156–167. doi:10.1109/TITB.2005.856864

Kau, L., & Chen, C. (2015) A Smart Phone-Based Pocket Fall Accident Detection, Positioning, and Rescue System. *IEEE Journal of Biomedical and Health Informatics*, 19, 44–56. doi:10.1109/JBHI.2014.2328593

Kevin Ashton (2009) That "Internet of Things" Thing. Retrieved from https://www.rfidjournal.com/articles/pdf?4986

Khan, A. M., Lee, Y. K., & Kim, T. S. (2008) Accelerometer signal-based human activity recognition using augmented autoregressive model coefficients and artificial neural nets. In: *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 5172–5175. doi:10.1109/IEMBS.2008.4650379

Khan, R., Khan, S. U., Zaheer, R., *et al.* (2012) Future internet: The internet of things architecture, possible applications and key challenges. In: *Proceedings - 10th International Conference on Frontiers of Information Technology*, pp. 257–260. doi:10.1109/FIT.2012.53

Kim, J. E., Boulos, G., Yackovich, J., *et al.* (2012) Seamless Integration of Heterogeneous Devices and Access Control in Smart Homes. In: *Eighth International Conference on Intelligent Environments*, pp. 206–213. doi:10.1109/IE.2012.57

Kim, J., Yun, J., Choi, S. C., *et al.* (2016) Standard-based IoT platforms interworking: implementation, experiences, and lessons learned. *IEEE Communications Magazine*, 54, 48–54. doi:10.1109/MCOM.2016.7514163

Knyazev, N. S., Chechetkin, V. A., & Letavin, D. A. (2017) Comparative analysis of standards for Low-power Wide-area Network. In: *2017 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SINKHROINFO)*, pp. 1–4. doi:10.1109/SINKHROINFO.2017.7997528

Kotsev, A., Schade, S., Craglia, M., *et al.* (2016) Next Generation Air Quality Platform: Openness and Interoperability for the Internet of Things. *Sensors* . doi:10.3390/s16030403

Kraijak, S., & Tuwanut, P. (2015) A survey on internet of things architecture, protocols, possible applications, security, privacy, real-world implementation and future trends. In: *2015 IEEE 16th International Conference on Communication Technology (ICCT)*, pp. 26–31. doi:10.1109/ICCT.2015.7399787

Kula, P. J. (2014) *Raspberry Pi Server Essentials*. Packt Publishing.

Kura (2015). Retrieved December 1, 2015, from http://www.eclipse.org/kura/

Kwolek, B., & Kepski, M. (2014) Human fall detection on embedded platform using depth maps and wireless accelerometer. *Computer Methods and Programs in Biomedicine*, 117, 489–501. doi:https://doi.org/10.1016/j.cmpb.2014.09.005

LAAS-CNRS;NCTU (2018) node-red-contrib-ide-iot. Retrieved March 14, 2019, from https://www.npmjs.com/package/node-red-contrib-ide-iot

Laird (2018) BL654 series. Retrieved March 5, 2019, from https://assets.lairdtech.com/home/brandworld/files/BL654 Datasheet v1_1.pdf

Lappeteläinen, A., Tuupola, J. M., Palin, A., *et al.* (2008) Networked systems, services and information the ultimate digital convergence. In: *1st International NoTA Conference, Helsinki, Finland*.

Lee, I., & Lee, K. (2015) The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons*, 58, 431–440. doi:https://doi.org/10.1016/j.bushor.2015.03.008

Leu, F.-Y., Ko, C.-Y., Lin, Y.-C., *et al.* (2017) Chapter 10 - Fall Detection and Motion Classification by Using Decision Tree on Mobile Phone. In: *Intelligent Data-Centric Systems* (eds Xhafa, F., Leu, F.-Y., & Hung, L.-L. B. T.-S. S. N.), pp. 205–237. Academic Press. doi:https://doi.org/10.1016/B978-0-12-809859-2.00013-9

Levis, P., Madden, S., Polastre, J., *et al.* (2005) TinyOS: An Operating System for Sensor Networks BT - Ambient Intelligence. In: (eds Weber, W., Rabaey, J. M., & Aarts, E.), pp. 115–148. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/3-540-27139-2_7

Levis, P., Patel, N., Culler, D., *et al.* (2004) *Trickle: A selfregulating algorithm for code propagation and maintenance in wireless sensor networks. In Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*.

Li, S., Xu, L.D. & Zhao, S. I. S. F. (2015) *The internet of things: a survey*. Springer US. doi:https://doi.org/10.1007/s10796-014-9492-7

Li, Y., Ho, K. C., & Popescu, M. (2012) A Microphone Array System for Automatic Fall Detection. *IEEE Transactions on Biomedical Engineering*, 59, 1291–1301. doi:10.1109/TBME.2012.2186449

Lindholm, B., Hagell, P., Hansson, O., *et al.* (2015) Prediction of Falls and/or Near Falls in People with Mild Parkinson's Disease. *PLoS ONE*, 10, e0117018. doi:10.1371/journal.pone.0117018

Ling, C. X., Huang, J., Zhang, H., *et al.* (2003) AUC: a statistically consistent and more discriminating measure than accuracy. In: *IJCAI*, Vol. 3, pp. 519–524.

Linux Foundation (2017) IoTivity. Retrieved May 1, 2017, from https://www.iotivity.org/

Lombardi, A., Ferri, M., Rescio, G., *et al.* (2009) Wearable wireless accelerometer with embedded fall-detection logic for multi-sensor ambient assisted living applications. In: *2009 IEEE Sensors*, pp. 1967–1970. doi:10.1109/ICSENS.2009.5398327

LoRa Alliance (2017) *LoRaWAN™ Specification v1.1*. Retrieved from https://lora-alliance.org/resource-hub/lorawantm-specification-v11

Luckham, D. (2002) The power of events, vol. 204. *Reading: Addison-Wesley*.

Lund, D., & Morales, M. (2014) *Worldwide and Regional Internet of Things (IoT) 2014-2020 Forecast : A Virtuous Circle of Proven Value and Demand*.

Ma, X., Wang, H., Xue, B., *et al.* (2014) Depth-Based Human Fall Detection via Shape Features and Improved Extreme Learning Machine. *IEEE Journal of Biomedical and Health Informatics*, 18, 1915–1922. doi:10.1109/JBHI.2014.2304357

Mahdavinejad, M. S., Rezvan, M., Barekatain, M., *et al.* (2018) Machine learning for internet of things data analysis: a survey. *Digital Communications and Networks*, 4, 161–175. doi:https://doi.org/10.1016/j.dcan.2017.10.002

Mahmud, R., Kotagiri, R., & Buyya, R. (2018) Fog Computing: A Taxonomy, Survey and Future Directions BT - Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives. In: (eds Di Martino, B., Li, K.-C., Yang, L. T., et al.), pp. 103–130. Singapore: Springer Singapore. doi:10.1007/978-981-10-5861-5_5

Mainetti, L., Patrono, L., & Vilei, A. (2011) Evolution of wireless sensor networks towards the Internet of Things: A survey. In: *SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks*, pp. 1–6.

Malheiros, L., Nze, G. D. A., & Cardoso, L. X. (2017) Fall detection system and Body positioning with Heart Rate Monitoring. *IEEE Latin America Transactions*, 15, 1021–1026. doi:10.1109/TLA.2017.7932688

Mao, A., Ma, X., He, Y., *et al.* (2017) Highly Portable, Sensor-Based System for Human Fall Monitoring. *Sensors* . doi:10.3390/s17092096

Mastorakis, G., & Makris, D. (2014) Fall detection system using Kinect's infrared sensor. *Journal of Real-Time Image Processing*, 9, 635–646. doi:10.1007/s11554-012-0246-9

McDermott-Wells, P. (2005) What is Bluetooth? *IEEE Potentials*, 23, 33–35. doi:10.1109/MP.2005.1368913

Mckinsey Global Institute (2015) *The Internet of Things : Mapping the Value Beyond the Hype*.

Mezghani, N., Ouakrim, Y., Islam, M. R., *et al.* (2017) Context aware adaptable approach for fall detection bases on smart textile. In: *2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pp. 473–476. doi:10.1109/BHI.2017.7897308

Mihini (2012). Retrieved December 1, 2015, from https://wiki.eclipse.org/Mihini

Min, W., Cui, H., Rao, H., *et al.* (2018) Detection of Human Falls on Furniture Using Scene Analysis Based on Deep Learning and Activity Characteristics. *IEEE Access*, 6, 9324–9335. doi:10.1109/ACCESS.2018.2795239

Minerva, R., Biru, A., & Rotondi, D. (2015) *Towards a definition of the Internet of Things (IoT)*. Retrieved from https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf

Miorandi, D., Sicari, S., Pellegrini, F. De, *et al.* (2012) Internet of things : Vision , applications and research challenges. *Ad Hoc Networks*, 10, 1497–1516. doi:10.1016/j.adhoc.2012.02.016

Molina, B. (2014) Empowering smart cities through interoperable Sensor Network Enablers. In: , pp. 7–12.

Mulligan, G. (2007) The 6LoWPAN architecture. In: *Proceedings of the 4th Workshop on Embedded Networked Sensors*, pp. 78–82. ACM.

Nest (2018) Nest Thermostat. Retrieved October 12, 2018, from https://nest.com/es/

Nevitt, MC, Cummings, S. (2018) Type of Fall and Risk of Hip and Wrist Fractures: The Study of Osteoporotic Fractures. *Journal of the American Geriatrics Society*, 41, 1226–1234. doi:10.1111/j.1532-5415.1993.tb07307.x

Nguyen, L. P., Saleh, M., & Le Bouquin Jeannès, R. (2018) An Efficient Design of a Machine Learning-Based Elderly Fall Detector BT - Internet of Things (IoT) Technologies for HealthCare. In: (eds Ahmed, M. U., Begum, S., & Fasquel, J.-B.), pp. 34–41. Cham: Springer International Publishing.

Nolan, K. E., Guibene, W., & Kelly, M. Y. (2016) An evaluation of low power wide area network technologies for the Internet of Things. In: *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 439–444. doi:10.1109/IWCMC.2016.7577098

Nottingham, M., & Hammer-Lahav, E. (2010) *Defining Well-Known Uniform Resource Identifiers (URIs)*. Retrieved from https://tools.ietf.org/html/rfc5785

Noura, M., Atiquzzaman, M., & Gaedke, M. (2018) Interoperability in Internet of Things: Taxonomies and Open Challenges. *Mobile Networks and Applications*. doi:10.1007/s11036-018-1089-9

oneM2M (2017) Standards for M2M and the Internet of Things: TS-0033 Interworking Framework. Retrieved March 8, 2019, from http://www.onem2m.org/component/rsfiles/files?folder=Release_3_Draft_TS

oneM2M (2018) Standards for M2M and the Internet of Things:Updated Release 2 Technical Specifications. Retrieved March 11, 2019, from http://www.onem2m.org/technical/published-drafts

OneM2M (2018) *TS-0001 Functional Architecture 2.19.0*. Retrieved from http://www.onem2m.org/technical/published-drafts

Open Mobile Alliance (2012) NGSI Context Management. Retrieved January 1, 2019, from http://www.openmobilealliance.org/release/NGSI/V1_0-20120529-A/OMA-TS-NGSI_Context_Management-V1_0-20120529-A.pdf

OpenIoT (2017) OpenIoT- Semantic-based platform. Retrieved May 1, 2017, from http://www.openiot.eu/

Oyegoke, A. (2011) The constructive research approach in project management research. *International Journal of Managing Projects in Business*, 4, 573–595. doi:10.1108/17538371111164029

Özdemir, T. A., & Barshan, B. (2014) Detecting Falls with Wearable Sensors Using Machine Learning Techniques. *Sensors* . doi:10.3390/s140610691

Pace, P., Gravina, R., Aloi, G., *et al.* (2017) IoT platforms interoperability for active and assisted living healthcare services support. In: *2017 Global Internet of Things Summit (GIoTS)*, pp. 1–6. doi:10.1109/GIOTS.2017.8016250

Pantsar, S., Purhonen, A., Ovaska, E., *et al.* (2012) Situation-based and self-adaptive applications for the smart environment. *Journal of Ambient Intelligence and Smart Environments*, 4, 491–516. doi:10.3233/AIS-2012-0179

Parker, C. (2011) An Analysis of Performance Measures for Binary Classifiers. In: *2011 IEEE 11th International Conference on Data Mining*, pp. 517–526. doi:10.1109/ICDM.2011.21

Perera, C., Zaslavsky, A., Christen, P., *et al.* (2014a) Context aware computing for the internet of things: A survey. *IEEE Communications Surveys and Tutorials*, 16, 414–454. doi:10.1109/SURV.2013.042313.00197

Perera, C., Zaslavsky, A., Christen, P., *et al.* (2014b) Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials*, 16, 414–454.

Phu, P. T., Hai, N. T., & Tam, N. T. (2015) A Threshold Algorithm in a Fall Alert System for Elderly People BT - 5th International Conference on Biomedical Engineering in Vietnam. In: (eds Toi, V. Van, & Lien Phuong, T. H.), pp. 347–350. Cham: Springer International Publishing. doi:10.1007/978-3-319-11776-8_85

Pierleoni, P., Belli, A., Palma, L., *et al.* (2015) A High Reliability Wearable Device for Elderly Fall Detection. *IEEE Sensors Journal*, 15, 4544–4553. doi:10.1109/JSEN.2015.2423562

Plataforma Arduino (2016). Retrieved February 12, 2016, from https://www.arduino.cc/

Prabhu, N. (2013) *Design and Construction of an RFID-enabled Infrastructure: The Next Avatar of the Internet*. CRC Press.

PyCOM (2019) FiPy. Retrieved March 5, 2019, from https://pycom.io/product/fipy/

Radatz J, Geraci A, K. F. (1990) *IEEE Standard Glossary of Software Engineering Terminology IEEE Std 610.12-1990*. Retrieved from http://www.mit.jyu.fi/ope/kurssit/TIES462/Materiaalit/IEEE_SoftwareEngGlossary.pdf

Rahmani, A., Thanigaivelan, N. K., Gia, T. N., *et al.* (2015) Smart e-Health Gateway : Bringing Intelligence to Internet-of-Things Based Ubiquitous Healthcare Systems. In: *Consumer Communications and Networking Conference (CCNC), 12th Annual IEEE*, pp. 826–834. doi:10.1109/CCNC.2015.7158084

Ray, P. P. (2018) A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*, 30, 291–319. doi:https://doi.org/10.1016/j.jksuci.2016.10.003

Raza, S., Misra, P., He, Z., *et al.* (2017) Building the Internet of Things with bluetooth smart. *Ad Hoc Networks*, 57, 19–31. doi:https://doi.org/10.1016/j.adhoc.2016.08.012

Razzaque, M. A., Milojevic-Jevric, M., Palade, A., *et al.* (2016) Middleware for Internet of Things: A Survey. *IEEE Internet of Things Journal*, 3, 70–95. doi:10.1109/JIOT.2015.2498900

Robie, K. (2010) Falls in older people: Risk factors and strategies for prevention. *JAMA*, 304, 1958–1959. Retrieved from http://dx.doi.org/10.1001/jama.2010.1599

Rossum, G. van, & Team, P. D. (2018) *Python Tutorial: Release 3.6.4*. usa: 12th Media Services.

Rougier, C., Meunier, J., St-Arnaud, A., *et al.* (2011) Robust Video Surveillance for Fall Detection Based on Human Shape Deformation. *IEEE Transactions on Circuits and Systems for Video Technology*, 21, 611–622. doi:10.1109/TCSVT.2011.2129370

Rozanski, N., & Woods, E. (2012) *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley. Retrieved from https://books.google.nl/books?id=ka4QO9kXQFUC

Rubenstein, L. (2006) *Falls in Older People: Epidemiology, Risk Factors and Strategies for Prevention. Age and Ageing*, Vol. 35 Suppl 2. doi:10.1093/ageing/afl084

Rykowski, J., & Wilusz, D. (2014) Comparison of architectures for service management in IoT and sensor networks by means of OSGi and REST services. In: *2014 Federated Conference on Computer Science and Information Systems*, pp. 1207–1214. doi:10.15439/2014F324

Santiago, J., Cotto, E., Jaimes, L. G., *et al.* (2017) Fall detection system for the elderly. In: *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 1–4. doi:10.1109/CCWC.2017.7868363

Santoyo-Ramón, A. J., Casilari, E., & Cano-García, M. J. (2018) Analysis of a Smartphone-Based Architecture with Multiple Mobility Sensors for Fall Detection with Supervised Learning. *Sensors* . doi:10.3390/s18041155

Sehairi, K., Chouireb, F., & Meunier, J. (2018) Elderly fall detection system based on multiple shape features and motion analysis. In: *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pp. 1–8. doi:10.1109/ISACV.2018.8354084

Semtech (2015) *LoRa™ Modulation Basics - AN1200.22*. Retrieved from https://www.semtech.com/uploads/documents/an1200.22.pdf

Seo, J., Nam, C., Yoon, S., *et al.* (2014) Group-based contention in IEEE 802.11ah networks. In: *2014 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 709–710. doi:10.1109/ICTC.2014.6983265

Shan, S., & Yuan, T. (2010) A wearable pre-impact fall detector using feature selection and Support Vector Machine. In: *IEEE 10th INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING PROCEEDINGS*, pp. 1686–1689. doi:10.1109/ICOSP.2010.5656840

Shi, W., Cao, J., Zhang, Q., *et al.* (2016) Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3, 637–646.

Shojaei-Hashemi, A., Nasiopoulos, P., Little, J. J., *et al.* (2018) Video-based Human Fall Detection in Smart Homes Using Deep Learning. In: *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5. doi:10.1109/ISCAS.2018.8351648

SIG, B. (2018) Specification of the Bluetooth System. Covered Core Package Version: 4.0. Retrieved October 9, 2018, from https://www.bluetooth.com/specifications/bluetooth-core-specification/legacy-specifications

SmartThings (2018) SmartThings is the easy way to turn your home into a smart home. Retrieved from https://www.smartthings.com/getting-started

Sotiropoulos, T., & Livshits, B. (2019) Static Analysis for Asynchronous JavaScript Programs. *arXiv Preprint arXiv:1901.03575*.

Stanford-Clark, A., & Truong, H. L. (2013) Mqtt for sensor networks (mqtt-sn) protocol specification. *International Business Machines (IBM) Corporation Version*, 1.

Stone, E. E., & Skubic, M. (2015) Fall Detection in Homes of Older Adults Using the Microsoft Kinect. *IEEE Journal of Biomedical and Health Informatics*, 19, 290–301. doi:10.1109/JBHI.2014.2312180

Sucerquia, A., López, J. D., & Vargas-Bonilla, J. F. (2017) SisFall: A Fall and Movement Dataset. *Sensors*, 17.

Sun, J., Wang, Z., Wang, H., *et al.* (2007) Research on Routing Protocols Based on ZigBee Network. In: *Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2007)*, Vol. 1, pp. 639–642. doi:10.1109/IIH-MSP.2007.263

Swetina, J., Lu, G., Jacobs, P., *et al.* (2014) Toward a standardized common M2M service layer platform: Introduction to oneM2M. *IEEE Wireless Communications*, 21, 20–26. doi:10.1109/MWC.2014.6845045

SymbIoTe (2015) Symbiosis of smart objects across IoT environments. Retrieved January 1, 2019, from https://www.symbiote-h2020.eu/

Tan, L., & Wang, N. (2010) Future Internet: The Internet of Things. In: *Advanced Computer Theory and Engineering (ICACTE), 3rd International Conference*, pp. 376–380.

Telecommunication Standarization Sector of ITU (2012) *Overview of the Internet of things*. Retrieved from https://www.itu.int/rec/T-REC-Y.2060-201206-I/es

Telecommunication Standarization Sector of ITU (2014) *Next Generation Networks – Frameworks and functional architecture models -Common requirements of the Internet of things (Y.2066)*.

Tolk, D. A. (2004) Composable Mission Spaces and M&S Repositories Applicability of Open Standards. In: *2004 Spring Simulation Interoperability Workshop, Paper 04S-SIW-009*.

Tong, L., Song, Q., Ge, Y., *et al.* (2013) HMM-Based Human Fall Detection and Prediction Method Using Tri-Axial Accelerometer. *IEEE Sensors Journal*, 13, 1849–1856. doi:10.1109/JSEN.2013.2245231

TRUSTe (2015) TRUSTe Internet of Things Privacy Index 2015. Retrieved November 5, 2018, from https://www.trustarc.com/

Tsvetkov, T. (2011) RPL: IPv6 Routing Protocol for Low Power and Lossy Networks. *Sensor Nodes–Operation, Network and Application (SN)*, 59, 2.

Valenciaport (2015) *Dynamic real-time lighting system for port terminals*. Retrieved from http://www.fundacion.valenciaport.com/Schedule-news/News/El-proyecto-SEA-TERMINALS-disena-un-sistema-de-ilu.aspx?lang=en-US

van der Veer, H., & Wiles, A. (2008) Achieving technical interoperability. *European Telecommunications Standards Institute*.

van Heijst, G., Schreiber, A. T., & Wielinga, B. J. (1997) Using explicit ontologies in KBS development. *International Journal of Human-Computer Studies*, 46, 183–292. doi:https://doi.org/10.1006/ijhc.1996.0090

Vermesan, O., Friess, P., Guillemin, P., *et al.* (2009) *Internet of Things Strategic Research Roadmap*. Retrieved from http://www.internet-of-things-research.eu/pdf/IoT_Cluster_Strategic_Research_Agenda_2011.pdf

Vongsingthong, S., & Smanchat, S. (2014) Internet of things: a review of applications and technologies.

W3C (2014) Efficient XML Interchange (EXI) Format 1.0 (Second Edition). Retrieved November 5, 2018, from https://www.w3.org/TR/exi/

W3C Information and Knowledge domain (2016) Extensible Markup Language (XML). Retrieved August 12, 2018, from https://www.w3.org/XML/

W3C Semantic Sensor Network Incubator Group (2011) Semantic Sensor Network Ontology. Retrieved September 18, 2018, from https://www.w3.org/2005/Incubator/ssn/ssnx/ssn

Wang, K., Delbaere, K., Brodie, M. A. D., *et al.* (2017) Differences Between Gait on Stairs and Flat Surfaces in Relation to Fall Risk and Future Falls. *IEEE Journal of Biomedical and Health Informatics*, 21, 1479–1486. doi:10.1109/JBHI.2017.2677901

Wang, Y., Wu, K., & Ni, L. M. (2017) WiFall: Device-Free Fall Detection by Wireless Networks. *IEEE Transactions on Mobile Computing*, 16, 581–594. doi:10.1109/TMC.2016.2557792

WSO2 (2015) WSO2 Architecture for Agility. Retrieved from https://wso2.com/

Wu, C. W., Lin, F. J., Wang, C. H., *et al.* (2017) OneM2M-based IoT protocol integration. In: *2017 IEEE Conference on Standards for Communications and Networking, CSCN 2017*, pp. 252–257. doi:10.1109/CSCN.2017.8088630

Wu, M., Lu, T., Ling, F.-Y., *et al.* (2010) Research on the architecture of Internet of things. *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, 5, V5--484.

Yacchirema, D., BelsaPellicer, A., Palau, C., *et al.* (2018) OneM2M Based-Interworking Architecture for Heterogeneous Devices Interoperability in IoT. In: *2018 IEEE Conference on Standards for Communications and Networking (CSCN)*, pp. 1–6. doi:10.1109/CSCN.2018.8581740

Yacchirema, D. C., & Palau, C. E. (2016) Smart IoT Gateway For Heterogeneous Devices Interoperability. *IEEE Latin America Transactions*, 14, 3900–3906. doi:10.1109/TLA.2016.7786378

Yacchirema, D. C., Sarabia-Jácome, D., Palau, C. E., *et al.* (2018) A Smart System for sleep monitoring by integrating IoT with big data analytics. *IEEE Access*, 1. doi:10.1109/ACCESS.2018.2849822

Yacchirema, D., de Puga, J. S., Palau, C., *et al.* (2019) Fall detection system for elderly people using IoT and ensemble machine learning algorithm. *Personal and Ubiquitous Computing*. doi:10.1007/s00779-018-01196-8

Yacchirema, D., Gonzalez-Usach, R., Palau, C., *et al.* (2018) Interoperability of IoT Platforms applied to the transport and logistics domain. In: . Zenodo. doi:10.5281/zenodo.1451428

Yacchirema, D., Palau, C., & Esteve, M. (2016) Design and Implementation of a Gateway for Pervasive Smart Environments. In: *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 4454–4459. Budapest, Hungary.

Yacchirema, D., Palau, C., & Esteve, M. (2017) Enable IoT Interoperability in Ambient Assisted Living: Active and Healthy Aging Scenarios. In: *1st Edition of Globe-IoT 2017: Towards Global Interoperability among IoT Systems*, p. 6. Las Vegas.

Yacchirema, D., Palau, C., & Esteve, M. (2019) Edge-of-Things Computing-Based Smart Healthcare System. In: *Handbook of Research on the IoT, Cloud Computing, and Wireless Network Optimization*, pp. 1–22. IGI Global.

Yacchirema, D., Sarabia-Jácome, D., Palau, C. E., *et al.* (2018) System for monitoring and supporting the treatment of sleep apnea using IoT and big data. *Pervasive and Mobile Computing*, 50, 25–40. doi:https://doi.org/10.1016/j.pmcj.2018.07.007

Yacchirema, D., Suárez de Puga, J., Palau, C., *et al.* (2018) Fall detection system for elderly people using IoT and Big Data. In: *Procedia Computer Science*, Vol. 130, pp. 603–610. doi:https://doi.org/10.1016/j.procs.2018.04.110

Yang, L., Ren, Y., & Zhang, W. (2016) 3D depth image analysis for indoor fall detection of elderly people. *Digital Communications and Networks*, 2, 24–34. doi:https://doi.org/10.1016/j.dcan.2015.12.001

Yoshida, T., Mizuno, F., Hayasaka, T., *et al.* (2005) A wearable computer system for a detection and prevention of elderly users from falling. In: *Proceedings of the 12th International Conference on Biomedical Engineering.*

Youn, J., Okuma, Y., Hwang, M., *et al.* (2017) Falling Direction can Predict the Mechanism of Recurrent Falls in Advanced Parkinson's Disease. *Scientific Reports*, 7, 3921. doi:10.1038/s41598-017-04302-7

Youngblood, G. M., Heierman, E. O., Holder, L. B., *et al.* (2005) Automation Intelligence for the Smart Environment. In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 1513–1514. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Retrieved from http://dl.acm.org/citation.cfm?id=1642293.1642535

Yun, J., Teja, R. C., Chen, N., *et al.* (2016) Interworking of oneM2M-based IoT systems and legacy systems for consumer products. In: *2016 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 423–428. doi:10.1109/ICTC.2016.7763511

Yuwono, M., Moulton, B. D., Su, S. W., *et al.* (2012) Unsupervised machine-learning method for improving the performance of ambulatory fall-detection systems. *BioMedical Engineering OnLine*, 11, 9. doi:10.1186/1475-925X-11-9

Z. Shelby, K. Hartke, C. B. (2014) The Constrained Application Protocol (CoAP) - RFC 7252. Retrieved July 15, 2016, from http://coap.technology/

Zach; Shelby, & Carsten;Bormann (2009) *6LoWPAN:The wireless embedded Internet*. Retrieved from http://keshi.ubiwna.org/2015IotComm/6LoWPAN_The Wireless Embedded Internet.pdf

Zanella, A., Bui, N., Castellani, A., *et al.* (2012) *Internet of Things for Smart Cities. Internet of Things Journal, IEEE*, Vol. 1. doi:10.1109/JIOT.2014.2306328

Zhang, C., & Ma, Y. (2012) *Ensemble machine learning: Methods and applications*. doi:10.1007/9781441993267

Zhu, Q., Wang, R., Chen, Q., *et al.* (2010) IOT Gateway: BridgingWireless Sensor Networks into Internet of Things. In: *EEE/IFIP International Conference on Embedded and Ubiquitous Computing*, pp. 347–352. doi:10.1109/EUC.2010.58

ZigBee Alliance (2018a) Zigbee IP and 920IP. Retrieved October 1, 2018, from https://www.zigbee.org/zigbee-for-developers/network-specifications/zigbeeip/

ZigBee Alliance (2018b) Low-power, low-cost, low-complexity networking for the Internet of Things. Retrieved October 8, 2018, from https://www.zigbee.org/zigbee-for-developers/network-specifications/