

Grado en Ingeniería de Tecnologías de la
Telecomunicación
2016 - 2017

Trabajo Fin de Grado

“Análisis e integración de dos soluciones horizontales para el Internet de las Cosas: oneM2M y OpenHAB”

Carlos de la Herrán Martín

Daniel Díaz Sánchez

Madrid, ... 2017



Esta obra se encuentra sujeta a la licencia Creative Commons **Reconocimiento – No Comercial – Sin Obra Derivada**

Agradecimientos

Abstract

Resumen

Índice

Índice

Índice de figuras

Índice de figuras

Índice de tablas

índice de tablas

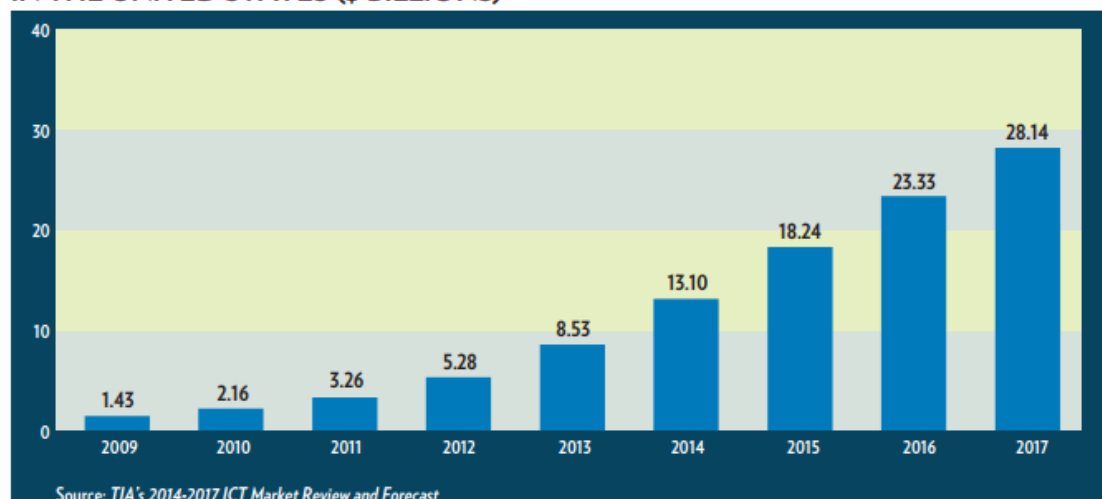
1. Introducción

1.1. Motivación del proyecto

En los últimos años, los avances tecnológicos y en telecomunicaciones nos hace predecir que una nueva revolución está a punto de aparecer en nuestras vidas: el Internet de las Cosas (**IoT – Internet of Things**). Este término expresa el hecho de que cualquier objeto de nuestra vida cotidiana, como una nevera, un coche, la calefacción de nuestra casa o incluso nuestras prendas de ropa sean dotados de cierta inteligencia (mediante sistemas embebidos) y estén conectados a Internet.

El rápido avance del Internet de las Cosas, y su gran popularidad, se debe a varios factores: la gran penetración actual de conexiones de banda ancha, el gran aumento de velocidad de las redes móviles y el aumento de la capacidad de computación de los microprocesadores, lo que permite desarrollar y fabricar dispositivos más pequeños y más baratos. Esto ha hecho posible la instalación de todo tipo de sensores en los dispositivos y las máquinas, cada vez más precisos y pequeños. Esto, unido a su conexión a Internet, nos lleva a un escenario en el que los consumidores y las empresas podrán obtener y analizar cantidades enormes de datos en tiempo real, permitiendo hacer decisiones para maximizar su eficiencia en tiempo y coste.

**MACHINE-TO-MACHINE SERVICES SPENDING
IN THE UNITED STATES (\$ BILLIONS)**



Además de los grandes cambios y transformaciones que puede suponer este nuevo paradigma en nuestra sociedad y en nuestra vida diaria, el **potencial económico** del Internet de las Cosas es enorme. En el año 2012 se estimó que un total de unos 8.700 millones de “cosas” estaban conectadas a Internet a lo largo del mundo, y las proyecciones estiman que este número crezca hasta los 50.000 millones en el año 2020, generando unos beneficios globales de 8.9 billones de dólares en el

proceso. Esto afectará principalmente al mercado de las tecnologías de la información y comunicación (TICs), sus fabricantes, proveedores y vendedores, pero también tendrá efectos secundarios positivos en todos los ámbitos de la economía mundial.

El último gran cambio social en nuestra especie se produjo durante los siglos XVIII y XIX, durante la **Revolución Industrial**, que produjo un punto de inflexión en todos y cada uno de los aspectos de la vida humana. Al sustituir la mano de obra humana y animal por maquinaria gracias a la introducción de nuevos avances técnicos, nuestra sociedad cambió para siempre. Cambiaron nuestros hábitos, nuestras costumbres; y mejoraron nuestra salud y nuestra calidad de vida. Como consecuencia, la población y la economía creció masivamente. Los avances tecnológicos del siglo XX, la llamada “Era Digital” pueden llegar a culminar con el Internet de las Cosas en el mayor cambio que se ha producido en nuestra sociedad desde entonces.

En este nuevo mundo de “cosas” conectadas, la mayoría de comunicaciones serán Máquina a Máquina (**M2M – Machine to Machine**), y habrá un continuo intercambio de información entre sensores, dispositivos y redes. Este término, M2M, es uno de los temas centrales de este trabajo y, aunque no se refieren a lo mismo, muchas veces es utilizado en distintos ámbitos como un sinónimo del IoT, ya que las fronteras entre ambos términos son difusas. Podemos definir la comunicación Máquina a Máquina como aquella comunicación en la que las “Máquinas” (o “cosas”) usan los recursos que les proporciona la red para comunicarse de forma automática entre ellas, con el propósito de su control o monitorización. Por lo tanto, la comunicación Máquina a Máquina es lo que provee al Internet de las Cosas de la capacidad de conexión que permite desarrollar su potencial. También se podría decir que ambos términos (IoT y M2M) se refieren al mismo concepto. La única diferencia residiría en que IoT estaría enfocado desde una perspectiva de un servicio en un mercado global, mientras que M2M se enfoca desde un punto de vista técnico de telecomunicaciones.

Las posibilidades del Internet de las cosas son enormes, y abarcan muchos y distintos campos de nuestra economía. Podemos hacer una división entre los diferentes campos en los que se están haciendo esfuerzos por implantar este nuevo paradigma, incluyendo varios ejemplos de implementación en cada uno de ellos:

- **Seguridad ciudadana:** sistemas de vigilancia, controles de acceso físico y sistemas de control medioambiental.
- **Redes eléctricas inteligentes:** producción y consumo de electricidad, gas y agua. Control de la producción según la demanda. También conocido con el término anglosajón “Smart Grids”.
- **Automoción:** gestión de flotas, seguridad al volante, navegación avanzada, información sobre el tráfico, pago automático de peajes y monitorización de la telemetría de los vehículos a distancia.

- **Pagos:** puntos de venta automatizados, cajeros y máquinas expendedoras conectadas.
- **Sanidad:** monitorización de las constantes vitales, ayuda para las personas mayores o minusválidos, diagnóstico remoto y acceso por web a profesionales médicos.
- **Control y mantenimiento remoto:** automatización industrial, sensores, iluminación, autodiagnóstico de fallos.
- **Dispositivos de consumo:** teléfonos móviles, tecnología vestible (relojes, pulseras o prendas inteligentes), ebooks. También incluimos aquí la domótica (automatización del hogar): automatización e interconexión de luces, sistemas de climatización, sensores de humos, etc.

Aunque el potencial del Internet de las Cosas en todos los ámbitos descritos anteriormente es muy grande, la mayoría de los primeros desarrollos e intereses en este entorno han tenido un enfoque vertical, es decir, con respecto a un mercado específico. Se ha llegado a desarrollar soluciones importantes y útiles de este modo, pero uno de los factores cruciales del IoT es la intercomunicación y la interoperatividad entre sistemas de diferentes mercados o industrias, y esto es imposible de conseguir con un enfoque vertical y con tecnologías “cerradas”. Para la correcta evolución del IoT necesitaremos alejarnos de este acercamiento vertical y de tecnologías cerradas para ir hacia un mundo de sistemas abiertos, basados en APIs y protocolos estandarizados en todos los niveles o capas.

Por lo tanto, para evitar un escenario adverso, es necesario desarrollar un entorno común con un enfoque horizontal, que pueda englobar todos los posibles casos de uso del Internet de las Cosas, que permita la intercomunicación entre todos los sistemas, seguido de aplicaciones específicas verticales cuando sea necesario para la resolución de problemas específicos. Alcanzar una uniformidad regulatoria y unos estándares comunes entre industrias que poco o muy poco tienen en común será sin duda muy complejo, y a veces imposible, pero maximizar los elementos que estas industrias tienen en común y en los que pueden converger será lo que libere el verdadero potencial de innovación del Internet de las Cosas.

El IoT continua confirmando su importante posición en el contexto de las TIC y del desarrollo de la sociedad. Sin embargo, es un campo todavía en plena maduración, particularmente debida a varios factores que limitan su plena explotación. Mientras que sus fundamentos básicos ya han sido elaborados y han conseguido una cierta madurez, todavía es necesario un gran esfuerzo para que el Internet de las Cosas pueda desarrollar todo su potencial. Estos esfuerzos resultarán en una mejor explotación del Internet de las cosas mediante una mayor interactividad entre todos los diferentes sectores de la industria, un mayor conocimiento del mundo que nos rodea y la utilización de un infinito espacio de resolución de problemas.

1.2. Objetivos

El principal objetivo de este trabajo es estudiar el tipo de soluciones horizontales y para el Internet de las Cosas. El primer paso será realizar un estudio de los estándares, tecnologías y consorcios que han ido apareciendo a lo largo de la última década para potenciar y regular el IoT.

Con el paso de los años, ha surgido la necesidad de un estándar horizontal que sea implementado a lo largo de todas las diferentes industrias verticales, para asegurar la facilidad de uso de la tecnología Máquina a Máquina (M2M), la interoperatividad de los datos generados por las diferentes “cosas” conectadas y un desarrollo eficiente de los sistemas M2M. Con este objetivo, en el año 2012, siete de las más importantes Organizaciones de Desarrollo de Estándares (SDOs – Standards Development Organizations) formaron la iniciativa global **oneM2M** para la estandarización de la comunicación Máquina a Máquina. Su objetivo es estandarizar una capa común de servicios M2M para cualquier tipo de caso de uso dentro del Internet de las Cosas, con el objetivo de facilitar aplicaciones M2M multi-industria, es decir, para habilitar la interconexión entre sistemas que a priori no tienen nada que ver entre si para conseguir objetivos como las “Smart Grids” o las “Smart Cities”, ciudades inteligentes. El objetivo de la iniciativa es que esta capa de servicios común sea implementada en todos los diferentes componentes de hardware y de software de las diferentes industrias para asegurar que los dispositivos M2M puedan comunicarse a gran escala.

El primer objetivo de este trabajo será estudiar a fondo el funcionamiento y la arquitectura de las diferentes especificaciones de oneM2M. Después, se diseñará una red oneM2M con diferentes dispositivos, puntos de acceso y aplicaciones para el testeo de la misma, y de los principios básicos de la plataforma horizontal, así como las facilidades que aporta y los retos que puede llegar a suponer. Se implementará esta red usando la plataforma **OM2M**. OM2M es un proyecto de la Eclipse Foundation, que consiste en una implementación, en Java, de código abierto de una capa de servicios común para la interoperatividad Máquina a Máquina basada en las especificaciones de oneM2M.

Existe otra iniciativa de código abierto y de cierta perspectiva horizontal llamada **OpenHAB**, orientada únicamente a la automatización del hogar. Esta está basada en el proyecto Eclipse Smarthome, también de la Eclipse Foundation y también implementada en Java. OpenHAB es un software para integrar diferentes sistemas y tecnologías de domótica o automatización del hogar en una única solución que permite implementar reglas de interconexión y automatización entre todos los diferentes dispositivos, así como una interfaz gráfica común para controlarlos. Otro de

los objetivos de este trabajo será estudiar esta solución, sus ventajas y su facilidad de uso, así como su instalación. Por último, se intentará integrar de alguna manera estas dos soluciones horizontales para el Internet de las Cosas, oneM2M y OpenHAB.

1.3. Contenido de la memoria

La primera parte de la memoria consiste en un análisis del “estado del arte” del Internet de las Cosas y de las comunicaciones Máquina a Máquina. Se hará una descripción y análisis de los más importantes protocolos, plataformas y alianzas que tienen el propósito de hacer avanzar el IoT.

Posteriormente, se hará una descripción y análisis más detallado de las tecnologías que se van a utilizar en este proyecto: oneM2M (y su implementación de código abierto OM2M) y OpenHAB. Se describirá su propósito, funcionamiento, arquitectura y funcionalidades. Se analizará también objetivamente los puntos fuertes y débiles de cada uno, así como las facilidades que pueden dar para desarrollar aplicaciones o plug-ins para las mismas.

Después de este análisis se expondrá el diseño de la red oneM2M a implementar, así como los pasos para hacerlo y se explicará detalladamente su funcionamiento. Se expondrá de manera que la red sea fácilmente reproducible por cualquiera que quiera probarla por él mismo. Así mismo, se procederá a la instalación de openHAB, a la descripción de su funcionamiento y al testeo de la plataforma para comprender los aspectos necesarios para desarrollar un plug-in funcional.

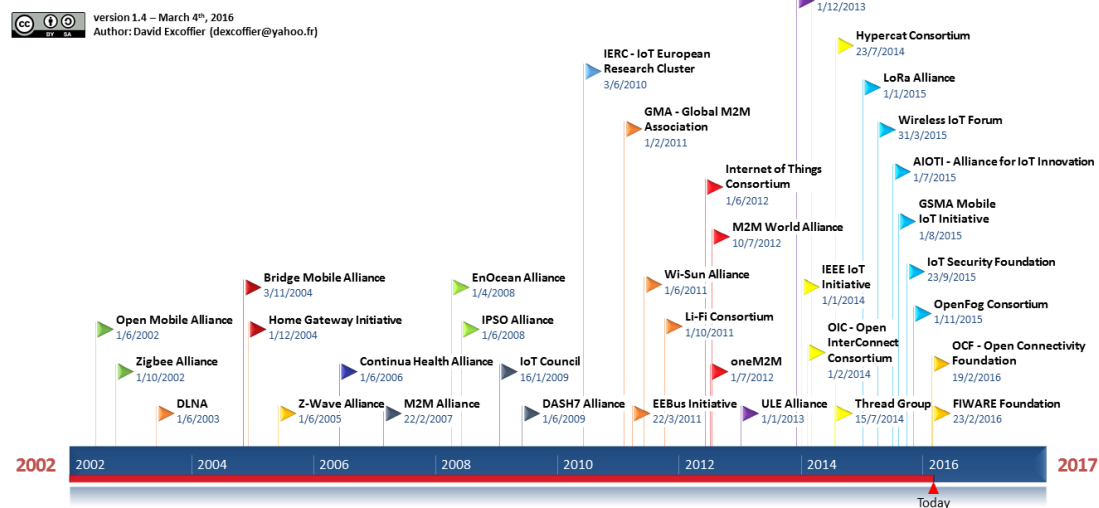
Por último, se estudiará la posibilidad de integrar el protocolo OneM2M como un “binding” de openHAB, para conseguir la interoperatividad entre ambos sistemas, de manera unidireccional o bidireccional.

2. Estado del arte

2.1. Alianzas, consorcios y grupos de trabajo

Gracias al gran avance en varios campos de la tecnología, como las comunicaciones móviles, han hecho posible que empecemos a pensar en el Internet de las Cosas como en una realidad muy cercana y prometedora, aunque todavía son necesarios grandes esfuerzos para que se convierta en realidad. A lo largo de la última década los grandes actores de varias industrias orientadas hacia el IoT; fabricantes, desarrolladores, organizaciones de estandarización, organizaciones gubernamentales, etc. se han dado cuenta de la necesidad de estándares consistentes y de la necesidad de cooperación en el desarrollo de los mismos para hacer del IoT un ecosistema realmente conectado, y permitir la interoperatividad entre diferentes sectores verticales.

Internet of Things Alliances & Consortia timeline



A continuación, se procederá a hacer una enumeración y análisis de las alianzas y consorcios de IoT más importantes en la actualidad

2.1.1. AIOTI

La Alianza para la Innovación del Internet de las Cosas (Alliance for Internet of Things Innovation) es una iniciativa iniciada en el año 2015 como resultado del desarrollo europeo y global de la tecnología y mercados del IoT. Su objetivo es crear y mantener ecosistemas de innovación dentro de la Unión Europea en el entorno global para poder afrontar los retos de estandarización, interoperatividad y legislación que supone este nuevo paradigma.

El trabajo de preparación de esta alianza se inició en el el IoT European Research Cluster (IERC), en el año 2014, y fue seguido de una reunión de alto nivel en la Comisión Europea en Bruselas en febrero de 2015. Esta reunión congregó a varios altos cargos de toda la cadena de valor del Internet de las Cosas: fabricantes de semiconductores y microelectrónica, operadores de red, teleoperadoras, proveedores de plataformas “en la nube”, proveedores de seguridad, así como varios

representantes de diferentes industrias como la energética, la automoción, iluminación, sanidad, etc. En esta reunión, se redactó la declaración “Momentum”, que define los pilares básicos de la AOITI, que son los siguientes:

- La Unión Europea tendrá el ecosistema y la industria de IoT más dinámico y ágil de todo el mundo, que transformará la vida de las personas, traerá empleo y crecimiento económico y supondrá varios desafíos sociales.
- Serán necesarias actividades de colaboración en innovación para conseguir un desarrollo satisfactorio del IoT.
- Entendiendo el potencial de las “cosas conectadas”, su inteligencia y datos que aportan, apoyamos la creación de un ecosistema para el IoT que apoye la creación de valor, escalabilidad, sostenibilidad, coexistencia y apertura.
- Los principios esenciales serán la cooperación y el compartimiento de conocimientos entre los actores existentes y nuevos dentro de la industria, y hacer posibles acuerdos flexibles con el objetivo de la convergencia, interoperatividad y estandarización.
- Serán necesarios modelos de referencia comunes y proyectos piloto de IoT a gran escala para hacer avanzar al IoT estimulando la creación, aceptación y lanzamiento de servicios, desde los puntos de vista del creador y del usuario de los mismos.

Hasta la fecha, la AIOTI se ha centrado en elaborar recomendaciones generales sobre temas como la privacidad y la seguridad. La AIOTI deberá facilitar la adopción de estas recomendaciones a sus miembros y asegurarse de ser el órgano de recomendación y consulta más importante para los mandatarios de los diferentes países en temas de IoT. En el año 2016 fue establecida como una organización sin ánimo de lucro por sus miembros fundadores.

Algunos de sus miembros son: ARTEMISIA, Arthur's Legal, ATOS, Bosch, BT, CNH Industrial, Digital Catapult, Engineering LOI, GRADIANT, Huawei, IBM, Infineon Technologies, John Deere, Nokia, Philips Lighting, Samsung, Schneider Electric, Siemens, STMicroelectronics, Telit Communications y Vodafone.

2.1.2. AllSeen Alliance

La alianza AllSeen fue fundada en el año 2013 por varias de las más importantes empresas de tecnología de consumo en el mundo, incluyendo a Qualcomm, Microsoft, Panasonic, Sony y LG. Su objetivo es permitir la interoperatividad estandarizada entre diferentes productos de diferentes fabricantes, mediante un entorno o “framework” común.

Con este objetivo, los miembros de esta alianza empezaron a desarrollar el entorno de desarrollo “AllJoyn”, un protocolo de código abierto que facilita la comunicación entre aplicaciones y/o dispositivos, para que los desarrolladores de software sean capaces de comunicarse y operar entre si sin tener que preocuparse del fabricante del dispositivo o de la capa física de transporte a usar. AllJoyn está ideado para funcionar en redes locales , y da a los dispositivos y aplicaciones la capacidad para encontrarse los unos a los otros dentro de la red y establecer comunicación. Su primera versión estable fue lanzada a finales de 2016 y da soporte a varios lenguajes de programación (C, C++, Objective-C y Java), a varias capas de transporte (Wi-Fi, Ethernet y PLC) y a múltiples plataformas (RTOS, Arduino, Linux, Android, iOS, Windows y Mac).

En octubre de 2016 la alianza AllSeen anunció su fusión con la Open Connectivity Foundation (**OIF**), bajo este mismo nombre, para acelerar el desarrollo del Internet de las Cosas. Esta organización fue creada por Intel, Broadcom y Samsung Electronics, con un objetivo muy similar al de la alianza AllSeen. También ellos han desarrollado un framework para la interoperatividad similar a AllSeen llamado IoTivity. A partir de ahora , ambos proyectos trabajarán de la mano para implementar las futuras especificaciones de la OIF en una única implementación de IoTivity que aglutine lo mejor de ambas plataformas.

2.1.3. IPSO Alliance

La alianza IPSO (Internet Protocol – Smart Object) es una organización sin ánimo de lucro fundada en el año 2008 por miembros de la industria tecnológica, de las telecomunicaciones y compañías energéticas. Como las anteriores, su objetivo es permitir que dispositivos IoT se comuniquen entre si permitiendo una interoperatividad global a través de estándares abiertos. Apoya un enfoque basado en IP (Internet Protocol) para conectar objetos inteligentes.

Su principal aportación es la definición del Smart Object Model.

-----> **Obtener más info**

Algunos de sus miembros son: ARM, BOSCH, Dust Networks, EATON, Ericsson, greenWAVE systems, intel, ST, Atmel and Sun Microsystems.

“The IPSO Alliance provides a foundation for industry growth by fostering awareness, providing education, promoting the industry, generating research, and creating a better understanding of IP and its role in the Internet of Things.”

2.1.5. IIC

El Industrial Internet Consortium es un programa global fundado en el año 2014 por AT&T, Cisco, General Electric, IBM e Intel, que promueve el crecimiento acelerado

del Industrial Internet of Things mediante la coordinación de iniciativas de ecosistemas para conectar, controlar e integrar de manera segura sistemas de la industria con las personas que los utilizan. Promueven el uso de arquitecturas comunes, interoperatividad y estándares abiertos para transformar la industria y obtener beneficios sociales en la misma y en las infraestructuras públicas.

El objetivo principal del IIC es acelerar el llamado “Internet Industrial”, apoyando la innovación mediante diferentes acciones:

- Utilizando los casos de uso del IoT existentes en la industria y creando otros nuevos que sirvan como proyectos de incubación para su posterior aplicación en el mundo real.
- Proveyendo a la industria de arquitecturas de referencia, casos de estudio y definición de estándares para facilitar el despliegue de tecnologías conectadas.
- Influenciando el proceso de desarrollo de estándares globales para Internet y sistemas industriales.
- Facilitando foros globales y abiertos en los que los actores del IoT y de la industria puedan compartir e intercambiar ideas, prácticas, lecciones y detalles.

2.1.6. oneM2M

La iniciativa global oneM2M es una asociación internacional creada con el objetivo de elaborar una serie de especificaciones para conseguir una capa de servicios M2M aplicable globalmente y que sea independiente de la o las tecnologías utilizadas, que permitan el fácil despliegue de aplicaciones M2M entre sistemas heterogéneos. Con esto pretenden facilitar aplicaciones IoT multi-industria como las “Smart Grids” o las “Smart Cities”. Esta capa de servicios M2M común está ideada para ir embebida en varios componentes de software y de hardware para asegurar que los dispositivos del Internet de las Cosas puedan comunicarse a escala global.

Esta iniciativa fue iniciada en el año 2012 por siete de las más importantes organizaciones de normalización y estandarización a nivel mundial: la “Association of Radio Industries and Businesses” (**ARIB**) y el “Telecommunication Technology Committee” (**TTC**), de Japón; la “Alliance for Telecommunications Industry Solutions” (**ATIS**) y la “Telecommunications Industry Association” (**TIA**), de Estados Unidos; la “China Communications Standards Association” (**CCSA**), de China; el “European Telecommunications Standards Institute” (**ETSI**), de Europa; y la “Telecommunications Technology Association” (**TTA**), de Corea del Sur.

El principal objetivo de esta alianza es minimizar la fragmentación de los estándares de capas de servicios M2M, muchos de ellos elaborados por las diferentes

alianzas que se han mencionado anteriormente, consolidando los estándares existentes actualmente y intentando juntar lo mejor de cada uno de ellos es un estándar global.

oneM2M planea aportar los siguientes beneficios al ecosistema M2M:

- Acelerando las economías de escala y recortando el tiempo de salida al mercado de productos M2M mediante la eliminación de la necesidad de desarrollar nuevos estándares.
- Simplificando el proceso de desarrollo de aplicaciones aportando un conjunto común de Interfaces de Programación de Aplicaciones (APIs).
- Haciendo uso de las redes globales existentes para el mayor potencial de los servicios M2M y para expandir las oportunidades de negocio basadas en estándares interoperables.
- Aportando evolución, desarrollo e interoperatividad de la capa de servicios M2M común y de sus funciones.

2.1.7. Eclipse IoT

El grupo de trabajo Eclipse IoT es una iniciativa amparada bajo la Fundación Eclipse, que apoya la colaboración entre organizaciones y desarrolladores individuales que comparten la meta de crear un entorno abierto, de software libre, para el Internet de las Cosas. Esta colaboración se centra en el desarrollo, promoción y adopción de tecnología de software libre para IoT.

Bajo el amparo de esta iniciativa se han desarrollado varias soluciones de software libre para el Internet de las Cosas. A continuación enumeraremos algunos de ellos:

- **Eclipse Edge:** para asegurar la portabilidad del software, es necesario que un dispositivo IoT incluya una capa de software que permita el acceso a las funcionalidades de hardware del microprocesador del dispositivo. Eclipse Edge aporta un API de alto nivel para acceder a estos recursos de hardware (GPIO, ADC, MEMS, etc.), que puede ser conectada directamente a las capas de software de bajo nivel aportadas por el fabricante del dispositivo.
- **Eclipse Paho:** este proyecto provee a los desarrolladores de una implementación fiable de código abierto del protocolo MQTT.
- **Eclipse Leshan:** consiste en una implementación en Java de código abierto del protocolo OMA Lightweight M2M (LWM2M).
- **Eclipse Smarthome:** aporta una plataforma para gateways IoT especialmente enfocadas en la automatización del hogar. Consiste en un entorno que permite

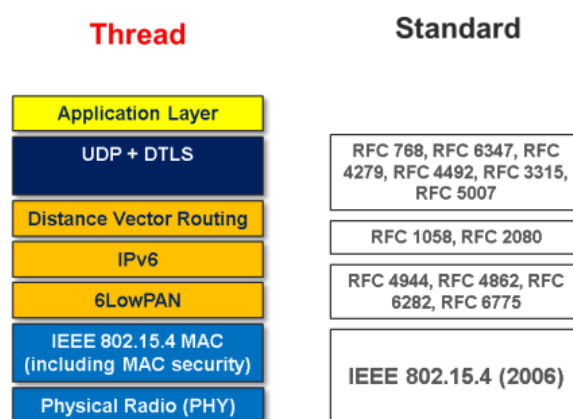
diseñar soluciones de domótica en entornos heterogéneos. Es decir, soluciones que tienen que enfrentarse a la integración de diferentes protocolos y estándares para poder controlar todos los dispositivos desde un punto central. Consiste en un sistema operativo, un contenedor de aplicaciones OSGi y aporta los servicios de conectividad entre dispositivos, almacenamiento y mensajería de los datos obtenidos de los mismos y su manejo remoto. OpenHAB, una de las tecnologías centrales que usaremos posteriormente en este proyecto, está basado en Eclipse Smarthome.

- **Eclipse OM2M:** consiste en una implementación en Java y en código abierto de las especificaciones de oneM2M. Aporta una Entidad de Servicios Común (CSE – Common Service Entity) que puede ser desplegada en gateways, servidores y dispositivos, y que aporta varios servicios comunes.

Algunos de los miembros de este grupo de trabajo son: Bosch Software Innovations, Eurotech, Red Hat Inc., Canonical, Deutsche Telekom, Siemens y Huawei, entre otros muchos.

2.1.8. Thread group

El grupo Thread fue creado en el año 2014, con la intención de elaborar un protocolo de comunicación (thread) que permita conectar y controlar todos los dispositivos del hogar de la manera más sencilla posible. El grupo fue establecido por varias empresas y organizaciones que no podían encontrar un protocolo de red inalámbrico que pudiese satisfacer sus necesidades en materia de automatización del hogar. Los siete miembros fundadores de esta iniciativa son ARM, Bigass Fans, freescale semiconductor, nest, Samsung, Silicon Labs y la Universidad de Yale.



Este grupo a desarrollado un protocolo de comunicación inalámbrica peer-to-peer usando estándares de red ya existentes, como podemos ver en la figura. Todas las capas menos la de aplicación se basan en estos estándares abiertos, siendo esta última propietaria y cerrada. El protocolo está diseñado para el hogar, con el objetivo de hacer converger diferentes proyectos de automatización del mismo. Está diseñado para soportar varios tipos de productos para el hogar como termostatos, electrodomésticos, iluminación, sistemas de entretenimiento, seguridad, etc.

2.1.9. Zigbee Alliance

La alianza Zigbee es una organización abierta y sin ánimo de lucro que fue establecida en el año 2002, con el objetivo de desarrollar los estándares necesarios para la comunicación inalámbrica de bajo consumo, para aplicaciones de control, monitorización o sensores implementados en dispositivos de pequeño tamaño que puedan funcionar con baterías durante meses e incluso años. Esta alianza es la responsable del protocolo de comunicación Zigbee, del que se darán más detalles en el siguiente apartado.

2.1.10 Z-Wave Alliance

“Established in 2005, the Z-Wave Alliance is comprised of industry leaders throughout the globe that are dedicated to the development and extension of Z-Wave as the key enabling technology for 'smart' home and business applications.”

→ [buscar más info](#)

2.1.11 LoRa Alliance

La alianza LoRa es una alianza abierta, sin ánimo de lucro, iniciada por varios actores de la industria con la misión de estandarizar las redes de área amplia de bajo consumo (LPWAN - Low Power Wide Area Networks) y su despliegue a lo largo del mundo para fortalecer el IoT, M2M, las “Smart Cities” y las aplicaciones industriales de este tipo de redes. Algunos de sus miembros son Orange, Swisscom, FastNet, proximus, SK telecom y Gemtek.

Ha creado la especificación de red de área amplia LoRaWAN, para su uso en redes de IoT con dispositivos alimentados por batería en un ámbito regional, nacional o global. Permite una comunicación bidireccional y servicios de movilidad y localización de los dispositivos, y promete una interoperatividad continua entre las “cosas inteligentes” sin la necesidad de complejas instalaciones locales y devolver la libertad a los usuarios, desarrolladores y fabricantes para acelerar la implementación del Internet de las cosas.

Handbook: Internet of Things Alliances and Consortia



Creo que aquí debería incluir una conclusión sobre el gran número de alianzas que hay ahora mismo en el IoT, que los objetivos de las mismas son parecidos y que muchas veces se está haciendo el mismo trabajo, que quizás debería haber unir sus fuerzas en una iniciativa como oneM2M.

2.2. Protocolos de IoT y M2M

Hacer una introducción a esta parte, de como los protocolos de capa física y de aplicación permiten la comunicación entre dispositivos, que cada uno tiene sus características determinadas y por lo tanto, ventajas e inconvenientes para cada caso de uso del M2M.

2.2.1. Protocolos de capa física

2.2.1.1. Wi-Fi

Falta

2.2.1.2 Bluetooth / Bluetooth Low Energy

Falta

2.2.1.3 RFID

RFID, o Identificación por Radio Frecuencia (Radio Frequency Identification) es una tecnología que se vale de los campos electromagnéticos para identificar y localizar objetos. Para ello, se usan etiquetas o “tags”, que se adhieren a los objetos y que contienen información almacenada de forma electrónica. Estas etiquetas suelen ser pasivas, no están alimentadas, y cuando un lector activo emite ondas, estas interactúan con la etiqueta receptora obteniendo así la información almacenada en ella.

El concepto de RFID conglomerar a muchas tecnologías diferentes con el mismo propósito, habiendo unos 140 estándares ISO con una amplio rango de diferentes aplicaciones de esta tecnología. La distancia de detención varía en rango de unos pocos centímetros a varios metros, siendo estas últimas las que utilizan etiquetas activas para poder abarcar una mayor distancia. Así mismo, las diferentes tecnologías de RFID pueden operar en un rango de frecuencias muy amplio.

Banda de frecuencias	Descripción	Rango
125 kHz	LF (Baja Frecuencia)	Hasta 50 cm.
13,56 MHz	HF (Alta Frecuencia)	De 8 cm.
400 MHz – 1.000 MHz	UHF (Ultra Alta Frecuencia)	De 3 a 10 m.
2,45 GHz – 5,4 GHz	Microondas	Más de 10 m.

Sus usos más comunes son el control y seguimiento de objetos en varias situaciones, como en almacenes, procesos de envío o sistemas de identificación y anti-robo en comercios y grandes superficies.

2.2.1.4. NFC

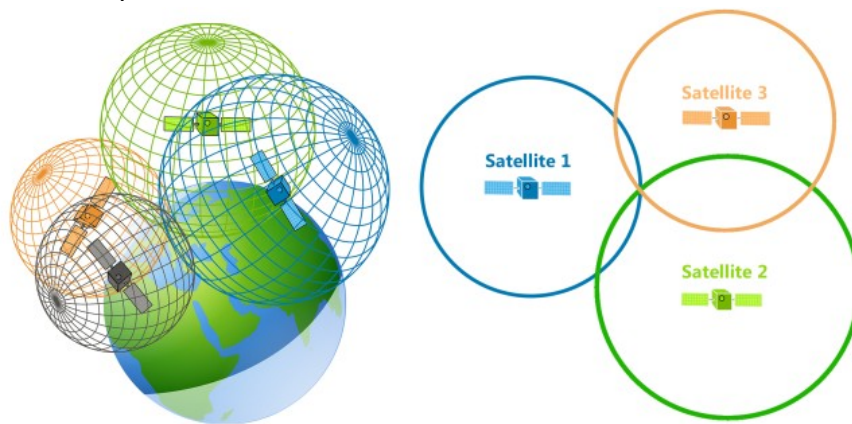
NFC (Near Field Communication) es una de las múltiples tecnologías de RFID. Su especificación engloba un conjunto de estándares desarrollados y promovidos por el NFC Forum, compuesto por empresas del sector de la electrónica de consumo y de la banca, que incluyen a MasterCard International, Microsoft, Motorola, NEC, Nokia, Panasonic, Philips, Renesas, Samsung Electronics, Sony, Texas Instruments y a Visa.

Permite la comunicación entre dos dispositivos que se encuentren muy cercanos el uno al otro, normalmente a unos 4-6 cm de máxima distancia entre ellos. Esta distancia es así de corta por motivos de privacidad y seguridad. El dispositivo necesita incluir un módulo NFC para poder realizar la comunicación. Normalmente estos módulos son activos, lo que permite una comunicación bidireccional, pero en algunos casos el módulo del receptor puede ser pasivo y alimentarse del campo electromagnético generado por el emisor en la comunicación. Opera en la banda de 13.56 Mhz y alcanza velocidades de transmisión desde 106 kbit/s hasta 424 kbit/s.

Su uso ha ganado gran popularidad el los llamados “smartphones”. Hoy en día casi todos los teléfonos móviles lanzados al mercado incluye un módulo NFC. Por eso, las aplicaciones de está tecnología más populares actualmente son el pago por móvil, juegos e identificación. También es popular su uso como forma de establecimiento de otro tipo de conexiones, como Wi-Fi y Bluetooth, sustituyendo a otros métodos como claves, ya que aporta seguridad al tener que estar ambos objetos muy cercanos entre sí para establecer la conexión.

2.2.1.5. GPS

GPS (Global Positional System) es un sistema de posicionamiento global creado por Estados Unidos inicialmente para aplicaciones militares que comenzó su despliegue en el año 1978. A partir de 1983 se abrió su uso para aplicaciones civiles, aunque con una menor precisión.

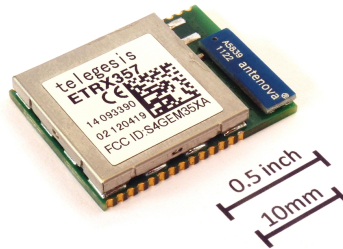


Consiste en una constelación de unos treinta satélites situados en seis planos orbitales diferentes de la órbita media terrestre (MEO), a unos 20.180 km de altitud con respecto a la Tierra. Permite la localización de cualquier dispositivo que disponga de un receptor GPS mediante la comunicación únicamente en sentido descendente entre los satélites que forman la constelación y el receptor. Para que el receptor pueda determinar su posición, necesita recibir la señal de al menos cuatro satélites diferentes. Al saber la posición exacta de estos cuatro satélites sobre la tierra y su distancia a los mismos, puede resolver la ecuación que determina sus coordenadas en la superficie terrestre, con un margen de error de 5 m para aplicaciones civiles. Este proceso se conoce como triangulación. El sistema utiliza varias bandas de alta frecuencia, como la de 1575.42 Mhz, la de 1227.60 Mhz o la de 1176.45, para cada uno de los tipos diferentes de satélites de los que está compuesta la constelación.

Su implantación en numerosos dispositivos de electrónica de consumo les permite saber en todo momento su localización exacta, por lo que se usa para seguimiento de personas o de flotas, navegación por carretera, aérea o marítima, etc. Existen en el mundo sistemas de posicionamiento mediante constelaciones de satélites similares, desarrollados por otras grandes potencias mundiales como GLONASS de Rusia, BeiDou de China o Galileo de la Unión Europea.

2.2.1.6 Zigbee

Zigbee es la especificación de un conjunto de protocolos, desarrollada por la Zigbee Alliance, para la comunicación inalámbrica de bajo consumo, para pequeños dispositivos que requieren una tasa baja de envío de datos y la mayor duración de sus baterías posibles. La especificación se aprobó en el año 2004 y se basa en el estándar IEEE 802.15.4 de redes inalámbricas de área personal.



Se ideó con el objetivo de llenar el vacío dejado por otros tipos de redes de área personal, tales como el WiFi o el Bluetooth, ya que está diseñado para cumplir funciones que no permiten estos dos últimos. En la siguiente tabla se comparan algunas de sus diferencias:

Estándar	Tasa de transmisión	Consumo de potencia
Wi-Fi	2 Mbps – 1.3 Gbps	400 mA transmitiendo y 20 mA en reposo (aprox.)
Bluetooth	0.7 Mbps – 3 Mbps	40 mA transmitiendo y 0.2 mA en reposo (aprox.)
Zigbee	250 kbps	30 mA transmitiendo y 3 mA en reposo (aprox.)

Como podemos ver en la tabla, la principal ventaja de Zigbee es su menor consumo frente a los otros dos, sobre todo en reposo, lo que permite a los dispositivos que utilicen este protocolo de transporte un consumo medio bastante menor. Otra ventaja es que permite varias topologías de red diferentes, como la topología en forma de estrella con un nodo central por el que pasan todas las comunicaciones, topología en árbol o la topología en malla, que es la más utilizada. Esta permite una comunicación peer-to-peer entre los diferentes nodos y una comunicación multisalto, por lo que un nodo de la red puede mandar información a cualquiera de los otros nodos, sea cual sea la distancia entre ellos, a través de la retransmisión de mensajes. Este tipo de redes pueden estar formadas por hasta unos 1000 nodos diferentes.

2.2.1. Redes Móviles

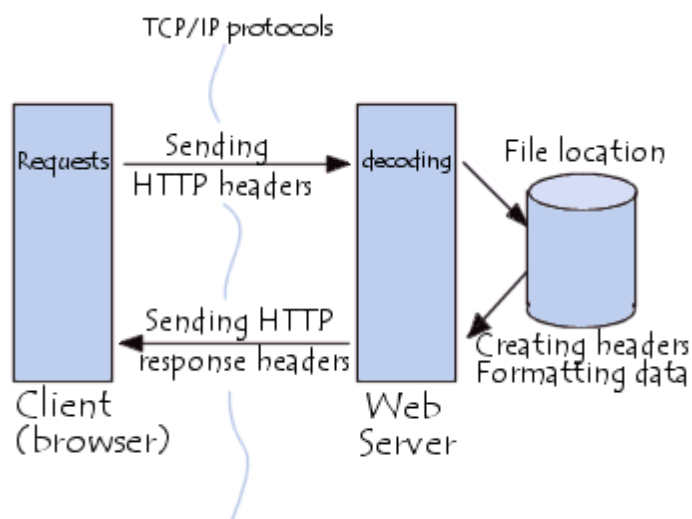
Las redes móviles agrupan los diferentes generaciones estándares de transmisión (2G, 3G y 4G) que utilizan el aire como medio de transmisión y la infraestructura de redes celulares ya existentes para la telefonía móvil. En un principio ideada sólo para esta última, para su uso en teléfonos móviles, en la actualidad se utilizan para dotar de conexión a Internet a varios tipos de dispositivos, como ordenadores o pequeños componentes del IoT como sensores o gateways. Es de gran utilidad sobre todo en zonas rurales y alejadas de los grandes núcleos urbanos en los que no existe una infraestructura de internet fija que cumpla sus necesidades, como las conexiones de fibra óptica.

2.2.2. Protocolos de capa de aplicación

2.2.2.1. HTTP

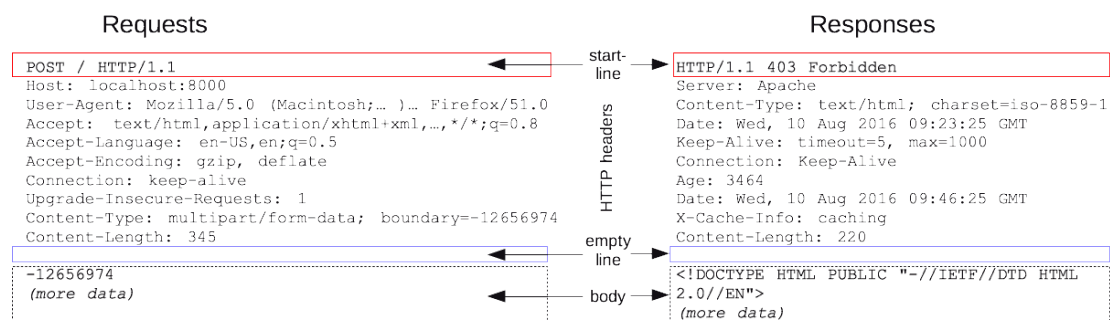
HTTP (Hypertext Transfer Protocol) es un protocolo de red a nivel de aplicación que fue el origen de lo que hoy conocemos como “la Web”. Su desarrollo empezó en el año 1989 y fue guiado por el Internet Engineering Task Force (IETF) y el World Wide Web Consortium (W3C). La versión más utilizada de este estándar hoy en día es la HTTP/1.1, que fue definida en el año 1997 en la RFC-2068 y recibió varias correcciones en los años 1999 y 2014 en las RFC-2616 y RFC-7290. Una versión posterior, la HTTP/2, fue estandarizada en el año 2015, y empieza a ser soportada por la mayoría de servidores Web.

HTTP permite la obtención de recursos o archivos, a través de la red, con el principal objetivo de conseguir los archivos necesarios para componer una página web, aunque también se utiliza para componer otros tipos de recursos. Los clientes se comunican con uno o varios servidores HTTP mediante el intercambio de mensajes individuales. Los mensajes que van en el sentido cliente-servidor se llaman **peticiones** y los que van en el sentido servidor-cliente se llaman **respuestas**. Entre una petición y una respuesta puede haber varias entidades por las que pase el mensaje, llamadas proxies, que pueden tener varias funciones como la re-dirección de los mensajes o su cacheo.



Los mensajes que intercambian cliente y servidor están compuestos por información codificada en ASCII y pueden ocupar varias líneas, siempre separadas por una retorno de carro y un salto de línea. Las peticiones y las respuestas comparten una estructura similar, y están compuestas por:

- **La línea inicial:** la primera línea del mensaje. En el caso de las **peticiones**, contiene el método de petición, seguido del identificador (URI) del recurso y de la versión de HTTP que soporta el emisor del mensaje. Los métodos de petición son verbos que definen las diferentes acciones que el cliente puede requerir a un servidor. Los más importantes son HEAD, GET, POST, PUT y DELETE. En el caso de las **respuestas**, la primera línea incluye la versión de HTTP que utiliza y el código de respuesta asociado a la petición. Los códigos de respuesta son muy variados y se agrupan en cinco categorías. Por ejemplo, las respuestas con formato 2xx corresponden a respuestas correctas, mientras que los que tienen formato 4xx corresponden a respuestas con errores causados por el servidor.
- **Cabeceras:** sirven para hacer más específica la petición o para describir mejor el cuerpo del mensaje. Se pueden agregar una o varias cabeceras a un mensaje, cada una formada por un valor y su nombre, con el formato Nombre: Valor, y cada una contenida en una nueva línea.
- **Cuerpo del mensaje:** como su propio nombre indica, esta parte está formada por el contenido del mensaje. El cuerpo del mensaje es opcional, algunos tipos de mensajes pueden llevar esta parte vacía ya que no les es necesaria.



HTTP se apoya sobre los protocolos TCP/IP, ya sea con o sin TLS (Transport Layer Security). Hay una versión segura del protocolo, llamada HTTPS, que utiliza un

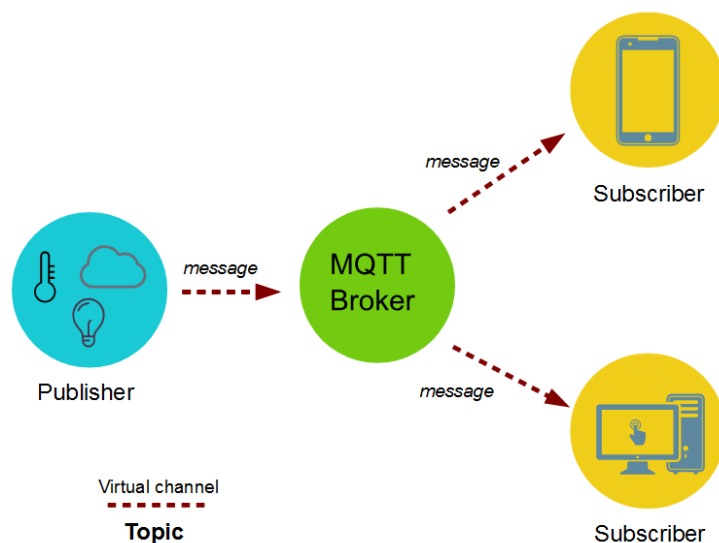
cifrado basado en SSL/TLS para crear un canal cifrado entre el servidor y cliente, en el que el contenido de los mensajes es cifrado. Por lo tanto, su uso está pensado para el intercambio de información sensible o secreta.

→ hablar sobre REST

2.2.2.2. MQTT

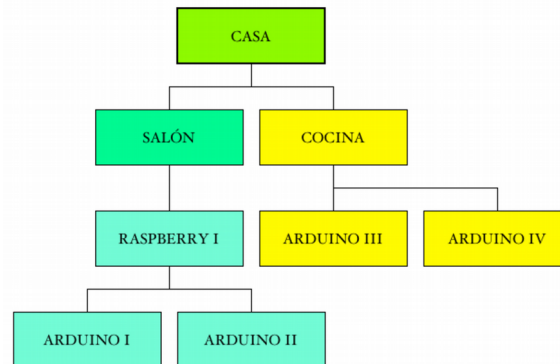
MQTT (Message Queue Telemetry Transport) es un protocolo de capa de aplicación ideado por IBM en el año 1999, diseñado para la transmisión de datos en redes de bajo ancho de banda y alta latencia. Su uso está orientado hacia sensores y dispositivos de pequeño tamaño, ya que consume pocos recursos, de procesamiento y de memoria. Está estandarizado en la norma ISO/IEC PRF 20922.

Este protocolo se basa en la idea de la publicación de mensajes en la red y de la subscripción a los diferentes “topics” o temas. Varios clientes se conectan a un servidor central, llamado “broker”, y se suscriben a los “topics” en los que están interesados. Los clientes pueden enviar mensajes al “broker” y publicar mensajes en un determinado “topic”, que el broker reenviará a todos los clientes que se hayan suscrito al mismo. Este método de comunicación permite que la comunicación sea de los tipos uno a uno o uno, o uno a muchos, sin que el cliente tenga que preocuparse por este hecho, ya que es el servidor el que se encarga de la retransmisión de los mensajes. Por lo tanto, el “broker”, junto con el propio protocolo MQTT, actúan como interfaz común que permite a los clientes enviar y recibir mensajes de forma sencilla.



Los “topics” o temas están estructurados en forma de árbol, lo que permite establecer una jerarquía entre ellos. Por lo tanto, permite establecer relaciones padre-hijo, lo que hace posible que al suscribirnos a un tema recibiremos los mensajes asociados a ese tema junto con los mensajes asociados a sus temas hijo. A la hora de representar la jerarquía del árbol de temas, se utiliza el carácter “/” para separar los

temas padre-hijo. Por ejemplo, en la siguiente figura, un cliente que quiera suscribirse al tema “Salón” debería hacerlo a CASA/SALÓN, y emezaría a recibir todos los mensajes publicados en dicho tema, así como los de sus hijos, CASA/SALÓN/RASPBERRY I, CASA/SALÓN/ARDUINO I y CASA/SALÓN/ARDUINO II. También tiene la posibilidad de suscribirse directamente al tema CASA para recibir los mensajes publicados a lo largo de toda la jeraquía de temas, ya que CASA es el tema raíz en el árbol jerárquico.



MQTT tiene funcionalidades de Calidad de Servicio (Quality of Service – QoS). El protocolo define tres niveles de QoS que representan el esfuerzo que deberán hacer el servidor y el cliente para asegurarse de que los mensajes son recibidos correctamente por los suscriptores. Esta diferenciación de QoS se puede hacer a nivel de mensajes individuales o a nivel de suscripción. Un nivel más alto de QoS significa un mayor número de retransmisiones del mensaje. A más alto nivel de QoS la comunicación es más fiable, pero conlleva una latencia mayor y un mayor uso del ancho de banda.

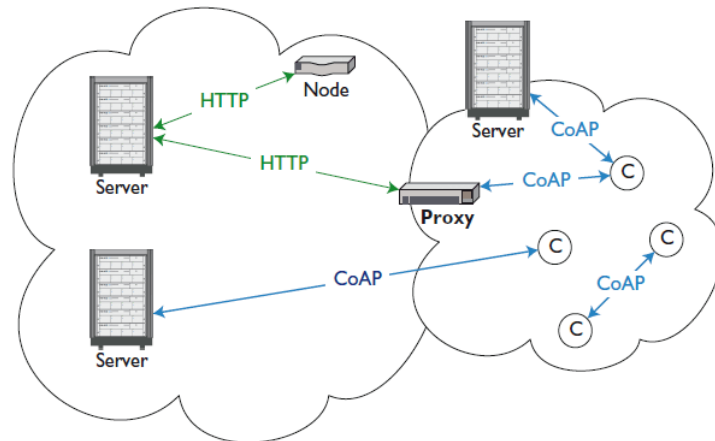
La capa de red sobre la que funciona el protocolo MQTT debe aportar una comunicación ordenada, bidireccional y sin pérdidas, pero no está definida específicamente por el estándar. En la práctica, se han desarrollado varias implementaciones basadas principalmente en uno o varios de los protocolos siguientes:

- TCP/IP.
- TCP/IP con Seguridad a Nivel de Transporte (Transport Level Security – TLS).
- WebSocket.

2.2.2.3. CoAP

CoAP es un protocolo de comunicación web especializado para su uso en dispositivos con recursos limitados, como sensores, interruptores y similares, con el objetivo de permitir la comunicación entre ellos a través de las redes estándares de Internet. Está diseñado para aplicaciones M2M tales como la automatización del hogar. Fue definido en el año 2014 en la RFC 5741.

Este protocolo es bastante similar en arquitectura y funcionamiento a HTTP, ya que está basado en este protocolo. La comunicación sigue el modelo petición-respuesta de HTTP entre servidor y cliente, además de soportar el modelo REST, identificar los recursos mediante URLs y soportar los principales tipos de petición de HTTP como GET, PUT, POST y DELETE. CoAP, por lo tanto, está diseñado para que pueda haber una correspondencia directa entre ambos protocolos, lo que permite por ejemplo acceder a recursos CoAP mediante peticiones HTTP de manera uniforme, mediante una interfaz “traductora” entre ambas redes.

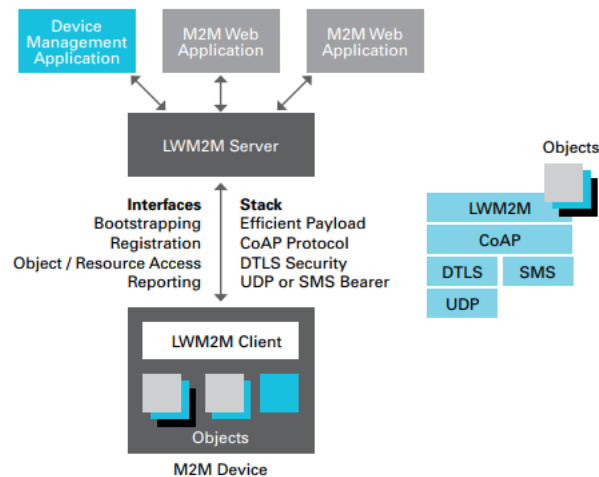


Sin embargo, CoAP presenta varias diferencias importantes con respecto a HTTP con el principal objetivo de consumir menos recursos de red y energéticos que este, para su uso en dispositivos limitados. Una de sus principales diferencias son que CoAP funciona sobre UDP (User Datagram Protocol), que es mucho más ligero en cuanto a recursos que TCP, pero no proporciona control de errores, por lo que algunos paquetes de información son perdidos. Para mitigar esto, CoAP proporciona un mecanismo de ACK (Acknowledgment), una confirmación de la correcta recepción de cada mensaje a nivel de aplicación. Otra gran diferencia es que además del modelo de comunicación basado en petición-respuesta entre cliente y servidor, CoAP también soporta un modelo de suscripción a temas y publicación de mensajes en estos grupos, parecido al de MQTT.

Payload	Payload
HTTP	CoAP
TCP	UDP
IP	IP
Ethernet link	Constrained link

2.2.2.4. OMA LWM2M

LWM2M (Lightweight Machine to Machine) es un protocolo diseñado por la Open Mobile Alliance. Es un protocolo de administración y control de dispositivos M2M diseñado para redes de sensores y de dispositivos sencillos de bajos recursos. Se basa en el principio REST y corre por encima de una capa de aplicación CoAP. Hay disponibles varias implementaciones de este protocolo para múltiples plataformas y dispositivos, como Eclipse Leshan de la Eclipse Foundation.



Su propósito es aportar una capa de servicios horizontal que permita la interoperatividad mediante su despliegue entre múltiples plataformas de sectores verticales, de manera similar a oneM2M. Pero, a diferencia de este último, LWM2M se centra solo en ofrecer servicios comunes para la administración de los dispositivos, incluyendo funciones como:

- Seguridad entre los servidores de administración (servidores LWM2M) y los dispositivos o clientes.
- Control de acceso que permite definir diferentes grupos y privilegios para la administración de los dispositivos.
- Obtención de toda la información asociada a cada dispositivo específico.
- Actualización del firmware de los dispositivos.
- Monitorización del estado de la conexión de los dispositivos y obtención de datos estadísticos de la misma.

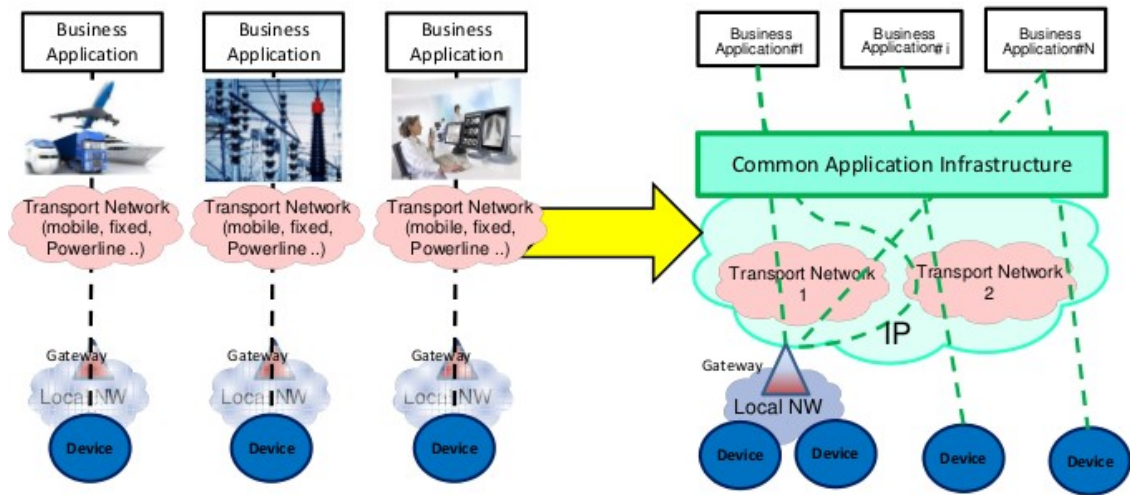
3. Arquitectura

A continuación se hará una descripción de la arquitectura y del funcionamiento de las dos plataformas a utilizar en este proyecto, oneM2M y OpenHAB.

3.1. oneM2M.

3.1.1. Introducción

oneM2M es una iniciativa global que tiene el objetivo de estandarizar una serie de protocolos que permitan la interoperatividad entre los diferentes sectores verticales de la industria. Hasta el momento, han aparecido numerosas soluciones para la comunicación M2M, pero se han centrado en su aplicación en sectores verticales, es decir, en soluciones integrales para cada diferente aplicación de este tipo de comunicación, que dan una solución a un problema determinado de extremo a extremo, pero que son incompatibles entre si.



oneM2M pretende estandarizar una plataforma de capa de servicios M2M común (Common Service Layer) que sirva como punto de inicio y común para servicios Máquina a Máquina que sea globalmente aplicable, es decir, en cualquier tipo de industria, negocio o aplicación. Como meta final, la Iniciativa Global oneM2M pretende por lo tanto minimizar la fragmentación actual existente entre los estándares de comunicación M2M existentes, consolidándolos y perfeccionándolos, y entre todos los actores de cada uno de estos estándares elaborar una serie de especificaciones conjuntas, oneM2M, que permita la interoperatividad entre todos ellos.

El primer paso de esta iniciativa fue elaborar una serie de documentos centrados en el estudio del mercado y de las soluciones existentes hasta ese momento para tener una referencia inicial para el desarrollo de las nuevas especificaciones. Primero se hizo un estudio de todos los posibles casos de uso de la comunicación Máquina a Máquina, recogido en el documento **(TR-0001 Uses Cases Collection)**, que cubre un amplio rango de diferentes aplicaciones y campos como el energético, empresarial, sanitario, de servicios públicos, de automatización del hogar, transporte y otros muchos. El objetivo de este documento es estudiar a conciencia cada uno de estos campos para tener en cuenta las necesidades específicas y peculiaridades de cada una de estas aplicaciones, pero siempre intentando identificar también las características que tienen en común, es decir, las funciones comunes que debería implementar una especificación de interoperatividad horizontal como oneM2m.

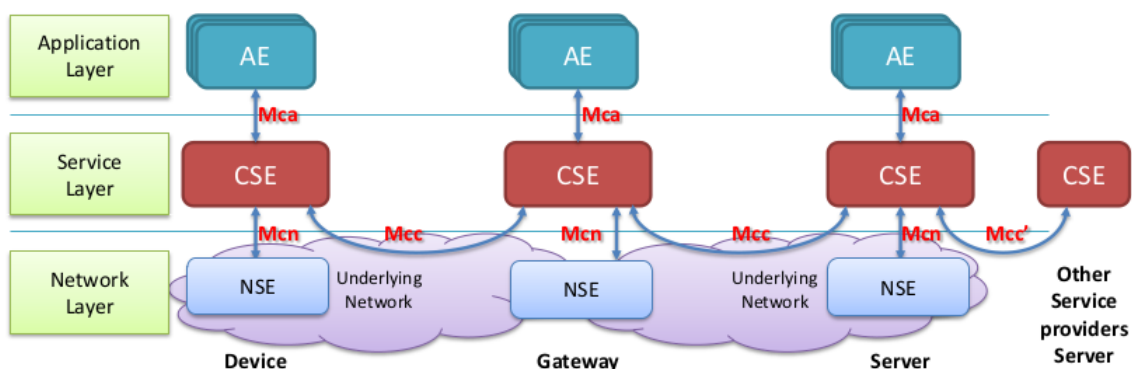
El siguiente paso, antes de empezar a definir las especificaciones y protocolos de oneM2M, fue formular los requerimientos exactos que la capa común de servicios de oneM2M debería cumplir para poder satisfacer las necesidades comunes de todos y cada uno de los casos de uso anteriormente estudiados y para poder permitir la total interoperatividad entre ellos. Estas especificaciones fueron recogidas en el documento **(TS-0002 Requirements)** muchas han sido ya cumplidas en las primeras dos versiones de oneM2M, pero otras están todavía en desarrollo o por implementar. Estos requerimientos fueron categorizados en siete grupos funcionales: globales del sistema, administración, operación, seguridad, monetización, comunicación y procesado de

datos. Además de estos requerimientos, el documento también recoge una serie de recomendaciones globales. Este documento también identifica los diferentes roles dentro de cualquier sector M2M, para poder satisfacer y equilibrar las necesidades de cada uno de ellos:

- El **usuario** (individual o empresarial): utiliza una solución M2M extremo a extremo para realizar una actividad determinada. Parte de esta solución debe incluir una serie de servicios comunes estén en conformidad con las especificaciones de oneM2M.
- La **aplicación M2M**: la parte de la solución que contiene todos los aspectos específicos de cada aplicación o industria vertical. Solución extremo a extremo de un problema, pero la interoperatividad de estas aplicaciones es inexistente sin una serie de servicios comunes entre las diferentes aplicaciones.
- Los **servicios comunes M2M**: una serie de servicios o funciones que permiten que las diferentes aplicaciones M2M verticales interactúen entre si. El objetivo de oneM2M es ofrecer este tipo de servicios.
- La **red subyacente**: los diferentes tipos de redes, locales y de área global, que permiten la conectividad y el tránsito de información entre las diferentes entidades M2M.

3.1.2. Arquitectura funcional

Para permitir aplicaciones o servicios M2M de extremo a extremo, oneM2M identifica tres capas diferenciadas dentro de su arquitectura: la capa de aplicación, la capa de servicios comunes y la capa de servicios de red. En la **figura X** podemos identificar las diferentes entidades de cada una de las capas y los puntos de unión entre las mismas:



Podemos identificar tres tipos de entidades:

- **Application Entity (AE):** la entidad de aplicación es la entidad que implementa la lógica de un servicio M2M determinado. Una misma entidad de aplicación puede ser instanciada en varios nodos M2M diferentes y cada nodo M2M puede contener una o varias Aes. Por ejemplo, una AE puede consistir en una aplicación que encienda bombillas, un sensor que aporte datos de la calidad del aire del entorno o una aplicación de smartphone que permita controlar diferentes dispositivos.
- **Common Services Entity (CSE):** la entidad de servicios comunes consiste en una particularización de un conjunto de funciones de servicio comunes que es ofrecidos a los otros dos tipos de entidades para la interoperatividad entre sistemas.
- **Network Services Entity (NSE):** la entidad de servicios de red proporciona los servicios de la red subyacente a las diferentes CSEs, para permitir el intercambio de información entre las mismas. OneM2M no define una implementación determinada de esta capa ni de los servicios que debe proporcionar, si no que permite su operatividad en diferentes tipos de redes.

Entre estas entidades funcionales encontramos varios tipos de puntos de referencia, consistentes en una o varias interfaces, que permiten la comunicación entre las mismas:

- **Mca:** punto de referencia por el que pasa la información entre una AE y una CSE. La comunicación entre estos dos tipos de entidades permite que una AE utilice los servicios comunes que ofrece una CSE y que esta CSE pueda comunicarse con la aplicación para ofrecer sus funcionalidades a otras entidades que pertenezcan a la misma capa de servicios.
- **Mcc:** punto de referencia por el que pasa la comunicación entre dos CSEs distintas, que permiten a una utilizar los servicios suministrados por la otra.
- **Mcn:** punto de referencia por el que pasa la comunicación entre una CSE y una NSE, que permite a la CSE utilizar los servicios proporcionados por la NSE, como la comunicación y el transporte de información.
- **Mcc':** cumple la misma función que el punto de referencia Mcc, pero una CSE pertenecientes a diferentes dominios M2M. Por lo tanto, permite que una CSE de un proveedor de servicios M2M determinado utilice las funciones ofrecidas por una CSE de otro proveedor.

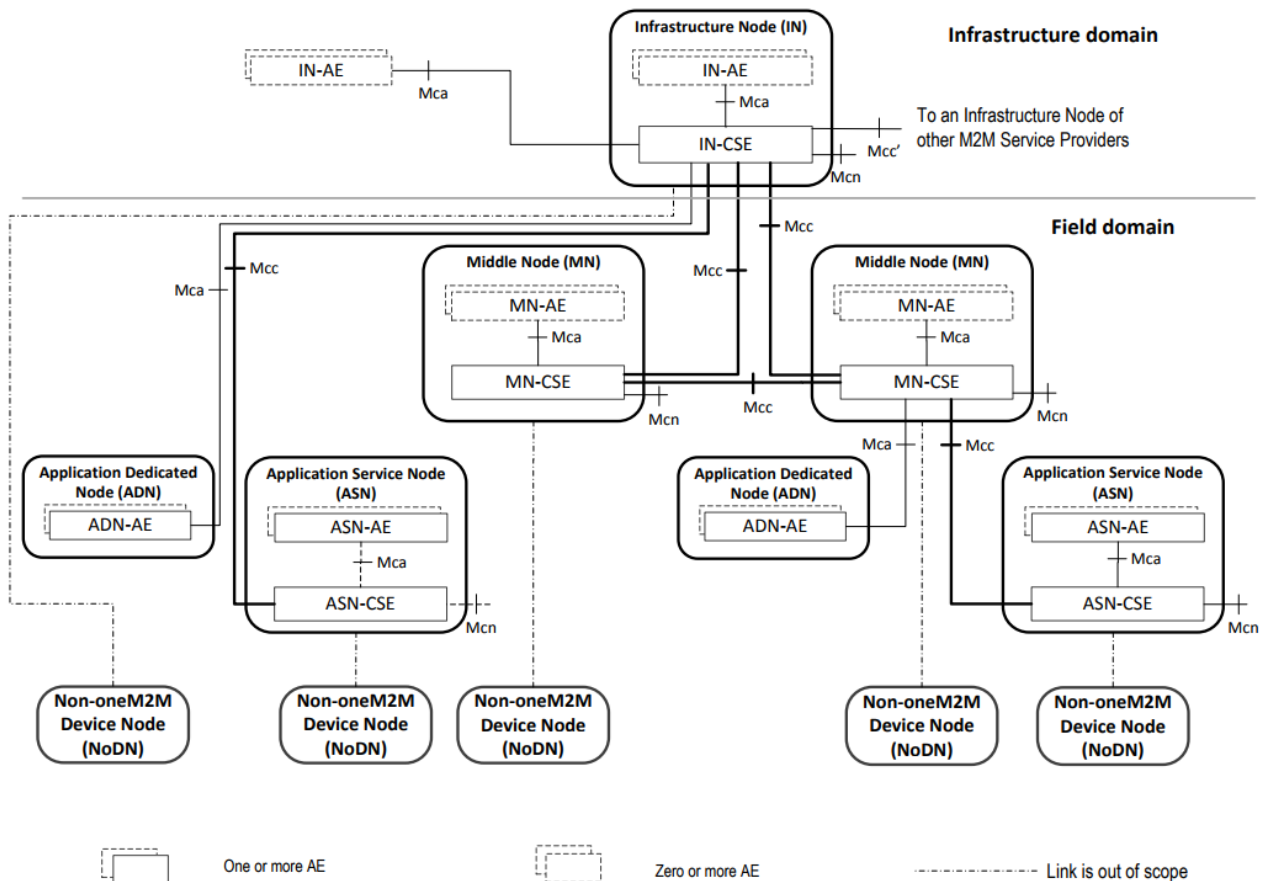
3.1.3. Tipos de nodos y posibles configuraciones

Los nodos son entidades lógicas que son identificables individualmente en un sistema oneM2M, que no tienen por qué equivaler a sistemas físicos. Podemos dividirlos en dos grupos diferenciados: los que contienen al menos una CSE y ninguna,

una o varias Aes, llamados “CSE-Capable Nodes”, y los que no contienen ninguna CSE y contienen cero, una o varias Aes, llamados “Non-CSE-Capable Nodes”.

La arquitectura de oneM2M permite los siguientes tipos de nodos:

- **Application Service Node (ASN):** nodo que contiene una CSE y al menos una AE. Puede implementarse en un dispositivo M2M sin excesivas restricciones de consumo o recursos. Un sistema oneM2M puede incluir cero, uno o varios ASNs.
- **Application Dedicated Node (ADN):** nodo que contiene al menos una AE y que no contiene ninguna CSE. Debe conectarse a otro nodo para acceder a los servicios comunes mediante su CSE. Puede implementarse en un dispositivo de recursos restringidos, ya que no necesita implementar la lógica de la CSE. Un sistema oneM2M puede incluir cero, uno o varios ADNs.
- **Middle Node (MN):** nodo que contiene una CSE y cero o más AEs. Las diferencias con un ASN es que no necesita contener una AE y que a un MN se pueden conectar otros nodos ADNs o ASNs. Se podría implementar en un punto de acceso M2M. Un sistema oneM2M puede incluir cero, uno o varios MNs.

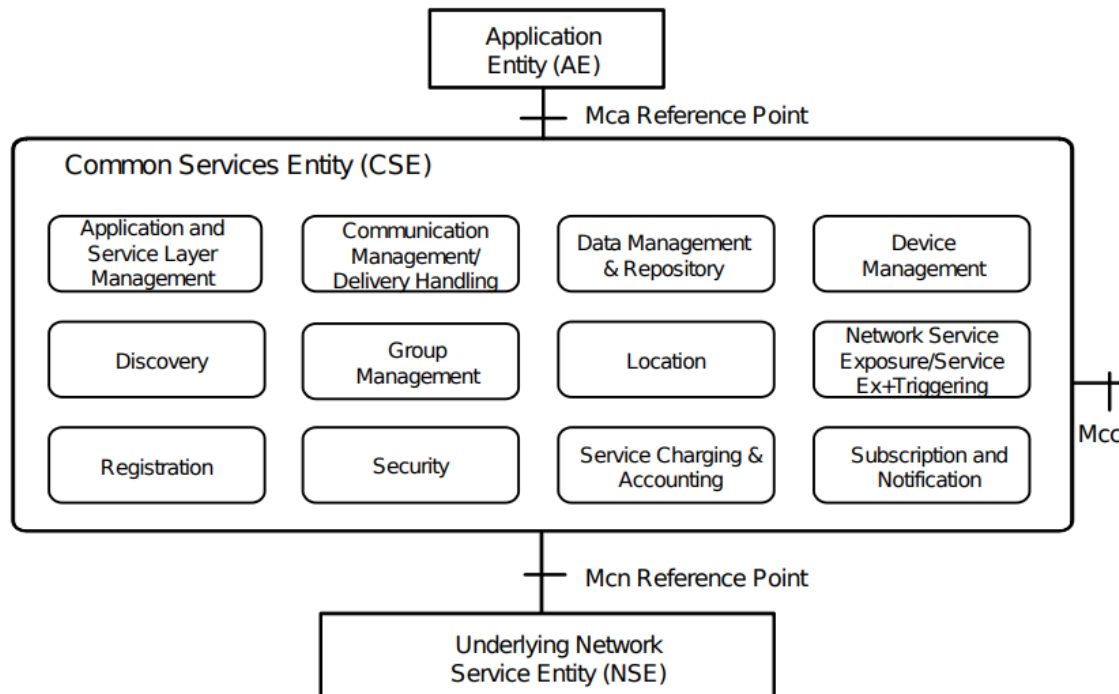


- **Infrastructure Node (IN):** nodo que contiene una CSE y cero, una o varias AEs. La CSE de un IN puede contener funciones especiales que no pueden ser utilizadas en las CSEs de otros nodos. Es necesario que en un sistema oneM2M haya un IN, y solamente uno, por cada proveedor de servicios oneM2M. Un IN se podría implementar en la infraestructura en la nube de un proveedor de servicios determinado.
- **No-oneM2M Node (NoDN):** nodo que no contiene ninguna entidad funcional de oneM2M, ni AEs ni CSEs, que representan dispositivos que no implementan las especificaciones de oneM2M, pero que pueden ser conectados al sistema para permitir su interoperatividad, mediante otro tipo de entidad llamada "Interworking Proxy".

Por último, podemos identificar dos dominios diferentes en la configuración de un sistema oneM2M: el dominio de infraestructura (Infrastructure Domain) y el dominio de campo (Field Domain). El dominio de infraestructura de un proveedor de servicios oneM2M determinado incluye un único IN. El dominio de campo es el que incluye el resto de tipos de nodos.

3.1.4. Funciones de Servicio Común

La capa de servicios comunes (Common Service Layer) es el principal servicio que aporta oneM2M para permitir la interoperatividad entre los diferentes sistemas verticales de la industria. Comprende un conjunto de funciones, llamadas Funciones de Servicios Comunes (Common Services Functions - **CSFs**) básicas y avanzadas que proveen de servicios M2M comunes a cualquier tipo de nodo o dispositivo oneM2M, sin importar la red subyacente que los conecta. Estas funciones residen en cada instancia de la capa de servicios comunes, es decir, en las CSEs incluidas en los diferentes nodos. Las CSFs proveen de los diferentes servicios a las AEs a través del punto de referencia Mca y a otras CSEs mediante el punto de referencia Mcc. Además, las CSFs también pueden interactuar con la red subyacente a través del punto de referencia Mcn, e incluso estas funciones pueden interactuar entre ellas dentro de una misma CSE.



Estas funciones se dividen en los siguientes grupos:

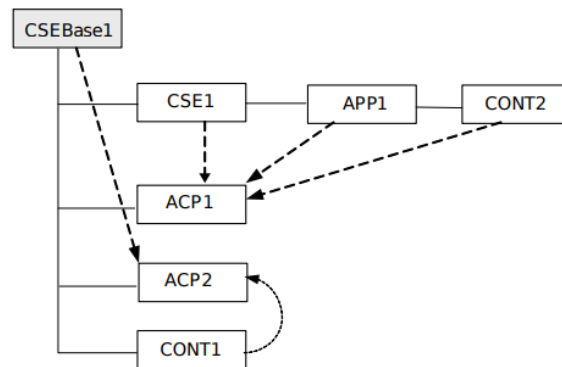
- **Application and Service Layer Management** (Administración de la Capas de Servicio y Aplicación): posibilita la administración de las CSEs y las AEs que residen en los diferentes tipos de nodo, incluyendo su configuración, actualización y la resolución de problemas.
- **Communication Management / Delivery Handling** (Administración de la Comunicación y Manejo de Envío): permiten la comunicación entre las diferentes entidades de oneM2M, es decir, entre las CSEs, AEs y NSEs. Estas funciones deciden en qué momento exacto y que ruta utilizar para llegar a cada uno de los destinatarios de los mensajes enviadas a través de ellas, además de almacenar dicha información en un buffer si es necesario su posterior uso o retransmisión.
- **Data Management and Repository** (Administración y Almacenamiento de los Datos): son las responsables de proporcionar almacenamiento y administración de los datos generados por las entidades o los dispositivos dentro de un sistema oneM2M. Incluye la capacidad de agregación de datos de diferentes fuentes para un uso determinado, así como la transformación de estos datos a un formato especificado.
- **Device Management** (Administración de Dispositivos): proporcionan la capacidad de administración de los dispositivos físicos en los que residen los diferentes nodos de un sistema oneM2M y de otros tipos de dispositivos que estén conectados a este sistema. Incluye capacidades como la actualización del firmware de dispositivos o la gestión de su batería.

- **Discovery** (Descubrimiento): sirven para que una entidad del sistema oneM2M pueda buscar información sobre aplicaciones y servicios contenidos en otras entidades o nodos del mismo sistema. El resultado de la búsqueda dependerá de un filtro determinado por el emisor y de las políticas de control de acceso que existan de ese emisor sobre los recursos a buscar.
- **Group Management** (Administración de Grupos): proporciona la capacidad de la creación, eliminación y edición de grupos de recursos, además de la posibilidad de realizar operaciones sobre la totalidad de recursos pertenecientes a un mismo grupo.
- **Location** (Localización): permite a las AEs obtener información sobre la ubicación geográfica de los nodos oneM2M. Esta información puede ser sobre el propio nodo en el que reside la AE solicitante de la información o sobre un nodo remoto.
- **Network Service Exposure, Service Execution and Triggering**: encargadas de controlar la comunicación entre las entidades de oneM2M y la red subyacente, a través del punto de referencia Mcn.
- **Registration** (Registro): el registro es el proceso por el cual una AE o una CSE envía su información a otra CSE para poder acceder a los recursos ofrecidos por esta última. Por lo tanto, estas funciones procesan las peticiones de registro originadas en una AE o CSE con el objetivo de permitir a las entidades registradas a acceder a una serie de recursos determinados ofrecidos por la CSE que aloja dichos recursos.
- **Security** (Seguridad): funciones que permiten establecer seguridad y privacidad dentro de una red oneM2M. Incluyen el manejo de datos sensibles, la administración de la seguridad, manejo de identidades y el control de acceso a los recursos del sistema por medio de la identificación, autenticación y autorización de las diferentes entidades y recursos.
- **Service Charging and Accounting** (Cobro por Servicio y Contabilidad): funciones que permiten la explotación económica de un sistema oneM2M. Estas funciones permiten la configuración de las políticas de cobro por el uso de la capa de servicios comunes y el control y registro de los cargos aplicados.
- **Subscription and Notification** (Suscripción y Notificación): permiten a las entidades de oneM2M suscribirse a la información generada por recursos o servicios del sistema, y se aseguran de que dicha información llegue correctamente a las entidades suscriptoras.

3.1.4. Gestión de recursos y flujo de datos

El modelo que sigue oneM2M para el permitir el acceso a los servicios y el manejo de información está basado en el concepto de recurso. Todos los servicios ofrecidos por un sistema oneM2M y toda la información almacenada en él están representados en forma de recurso. Un recurso es una estructura de datos que puede

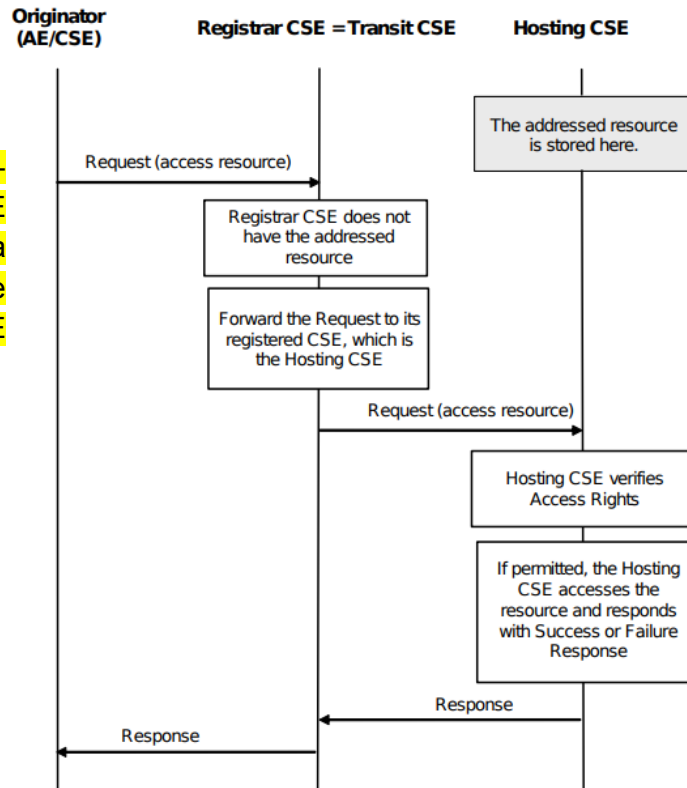
ser identificada inequívocamente con una Identificación de Recurso Uniforme (Uniform Resource Identifier – URI), que consiste en un conjunto de caracteres usados para identificar dicho recurso.



Cada recurso es identificado por su URI única y debe pertenecer a un tipo determinado de recurso de entre los que son definidos por las especificaciones de oneM2M. Esta clasificación es la que determina el significado de la información que contiene cada recurso. Un gran número de tipos de recursos han sido definidos, como los propios tipos de nodo oneM2M (node, CSEBase, AE, etc) y varios tipos definidos por cada grupo de funciones de servicio común (CSFs), como por ejemplo el tipo accessControlPolicy para las funciones de seguridad, los tipos group y members para las funciones de agrupación, el tipo locationPolicy para las funciones de localización... Existe una tabla que detalla todos los tipos de recursos que pueden formar parte de un sistema oneM2M en el documento (TS-0001, tabla 9.6.1.1-1). Los recursos están organizados dentro del sistema en una estructura con forma de árbol, con enlaces entre los nodos padre e hijos y otras posibles asociaciones que se pueden establecer, como se muestra en la figura X.

Los recursos son manejados por las diferentes entidades en base a cinco operaciones: creación, lectura, actualización, eliminación y notificación, operaciones que se pueden ejecutar sobre cualquier tipo de recursos siempre que el originador de la operación tenga los permisos necesarios para hacerlo. Este conjunto de operaciones se conoce por las siglas, en inglés, de los cuatro verbos que las forman, es decir, CRUD (Create, Retrieve, Update and Delete). Todas estas operaciones se realizan mediante intercambio de mensajes a través del sistema a través de los puntos de referencia explicados anteriormente, basado en el modelo de petición-respuesta.

Figure 8.2.1.0-2: AE/CSE accesses a resource at the Hosting CSE (One Hop)



Las peticiones indican una de las cuatro operaciones posibles (CRUD) y el identificador único del recurso sobre el que se desea realizar la operación (URI), y opcionalmente información adicional en el cuerpo de la misma que sea necesaria para realizar la operación. Los mensajes pueden ser originados por una AE o por una CSE, y el receptor siempre será otra CSE. La CSE receptora comprobará si el recurso sobre el que se hace la petición reside en ella misma. Si es así y si las políticas de acceso al recurso lo permiten, realizará la operación sobre el recurso y devolverá el resultado al originador de la petición. Si el recurso en cambio reside en otra CSE, la receptora de la petición se encargará de redirigir la petición a la CSE que contiene el recurso, actuando de mediadora, recibiendo la respuesta de la CSE del recurso y redirigiendo las misma al emisor de la petición. Las peticiones se pueden realizar tanto de forma síncrona, es decir, esperando la respuesta antes de hacer cualquier otra operación, o de forma asíncrona, pudiendo realizar varias operaciones en cadena sin tener que esperar a la respuesta de la anterior para realizar la siguiente.

3.1.5. Protocolos de oneM2M

Los protocolos que componen las especificaciones de oneM2M son muchos y podemos hacer una clara división entre cuatro tipos de ellos:

Protocolos de Capa de Servicio: son los que definen la capa de servicios comunes del un sistema M2M. Definen los tipos de recursos, la identificación de los mismos, las funciones comunes, el flujo de datos, etc. Es decir, sobre todo lo que se ha explicado hasta esta momento en el trabajo sobre las especificaciones de oneM2M.

Encapsulado de protocolos existentes: una de las metas de oneM2M es impulsar el uso y el desarrollo de protocolos existentes que puedan cumplir los requerimientos y las especificaciones de oneM2M. Por lo tanto, protocolos existentes y muy probados de seguridad y administración de dispositivos, están siendo encapsulados, es decir, se traducen los recursos de estos protocolos a recursos equivalentes a la especificación de recursos de oneM2M para poder utilizarlos en el sistema.

Enlace con protocolos subyacentes: siendo un protocolo horizontal, oneM2M debe confiar en varios protocolos subyacentes para el envío y transporte de sus mensajes, especificando un mapeo entre los mensajes de oneM2M y sus equivalentes en estos protocolos. En este momento oneM2M ha publicado especificaciones para realizar el intercambio de mensajes mediante los protocolos HTTP, MQTT, CoAP y WebSocket.

Interoperatividad con sistemas existentes: oneM2M, siendo una plataforma horizontal, necesita poder comunicarse con varios sistemas de la industria ya existentes y extendidos que no sean compatibles con él. Para ello, se formularán especificaciones que definen entidades llamadas “Interworking Proxies”, que traduzcan los mensajes y los recursos entre un sistema oneM2M y otro sistema propietario.

3.1.4. OM2M

OM2M es una implementación de algunas de las especificaciones publicadas de oneM2M en Java, de código abierto. El proyecto forma parte de la Eclipse Foundation. Proporciona un API REST para crear, administrar y acceder a los recursos de un sistema oneM2M, independiente de la red subyacente. Tiene una arquitectura modular que corre en una plataforma OSGi, por lo que es fácilmente extensible mediante plug-ins. Además, proporciona una interfaz web desde la que se puede visualizar todo el árbol de recursos del sistema oneM2M.

Los tipos de recursos de oneM2M soportados por la versión actual de OM2M son los siguientes:

- **CseBase:** representa la CSE “anfitriona”, es decir, la CSE que reside en el propio nodo. Es la raíz por la cual se accede a todo el resto de recursos alojados en la misma CSE.
- **remoteCse:** contiene la información relacionada con otras CSEs residentes en el resto de nodos de un sistema oneM2M. Para que exista este recurso ha de haber antes autenticación mutua entre los diferentes nodos. Permite realizar operaciones sobre recursos residentes en nodos externos.
- **AE:** contiene la información de una entidad de aplicación (AE) una vez se haya registrado de manera satisfactoria en la CSE anfitriona.
- **Container:** actúa como mediador para permitir el intercambio de información entre las AEs y las CSEs almacenando dicha información.

- **AccessControlPolicies:** políticas de control de acceso. Administran los permisos y las identidades registradas para limitar y proteger el acceso a la estructura del árbol de recursos.
- **Group:** permite crear y administrar grupos de recursos para poder realizar operaciones conjuntas sobre todos ellos.
- **Subscription:** almacena la información relacionada con las suscripciones a los datos de recursos determinados y permite a los suscriptores recibir actualizaciones de dicha información.

Soporta varios protocolos subyacentes para la transmisión de los mensajes, ya que tiene enlaces con los protocolos HTTP, MQTT y CoAP. En esta trabajo va a usarse OM2M sobre HTTP para el despliegue del sistema oneM2M a analizar.

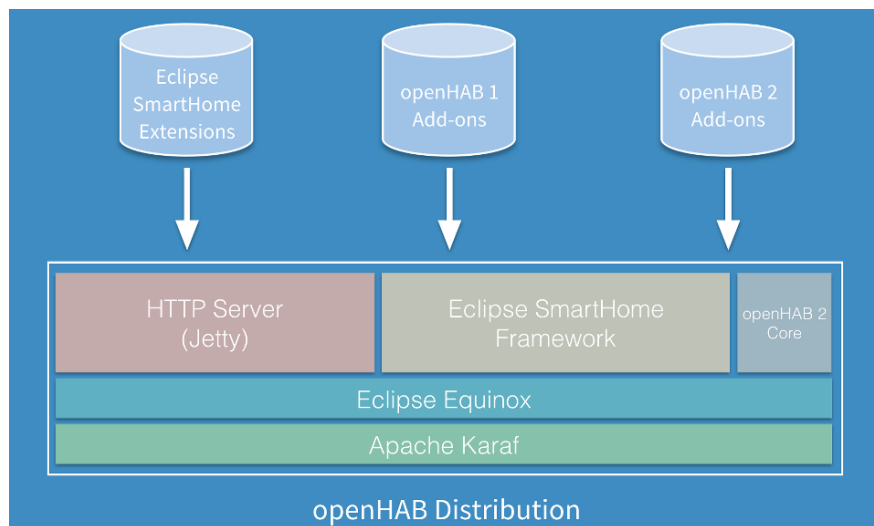
3.2. OpenHAB / Eclipse Smarthome

3.2.1. Introducción

El ecosistema de la automatización del hogar está altamente fragmentado. Existen múltiples soluciones diferentes enfocadas en los mismos problemas, y muchas veces cada fabricante utiliza sus propios sistemas, protocolos y estándares para el control de sus dispositivos. En los últimos años se ha producido un gran crecimiento del mercado de los dispositivos móviles, smartphones y tablets, y se ha abierto la posibilidad a que los usuarios de estos dispositivos los utilicen para controlar los “dispositivos inteligentes” que forman parte de su hogar mediante aplicaciones o interfaces web de una manera sencilla. El problema es que no existía ninguna solución única para poder controlarlos todos desde un mismo punto. Para solucionar este problema apareció openHAB en el año 2010.

OpenHAB es un plataforma de software libre que sirve para integrar diferentes soluciones y dispositivos de automatización del hogar. Actúa como sistema central de control del llamado “Intranet de las Cosas”, concepto que agrupa a todos los tipos de dispositivos conectados que forman parte de un hogar y que, en principio, no están pensados para ser controlados desde el exterior, ya que su ámbito es el de las redes locales. Como punto central, openHAB tiene la capacidad de conectarse a una gran variedad de este tipo de dispositivos de muchos fabricantes diferentes para poder controlarlos todos al mismo tiempo desde un mismo sistema, pudiendo también establecer reglas de automatización para que interactúen entre ellos.

En el año 2013 su creador, Kai Kreuzer, preocupado por los aspectos legales a los que se tenían que enfrentar muchos proyectos de software libre, vio la necesidad de poner el código base de openHAB a cargo de una asociación de software libre seria, bajo licencias que pudiesen mantener la salud y desarrollo del proyecto. Con esa premisa y bajo el amparo de la Eclipse Foundation, se creó el proyecto Eclipse Smarthome. Este proyecto toma los conceptos y la base sobre la que está construida la primera versión de openHAB, mejorando y evolucionando algunas de ellas, para proporcionar a los desarrolladores del ámbito del Internet de las Cosas y de la automatización del hogar un entorno de desarrollo o framework que permita construir fácilmente sobre él soluciones para el usuario final. Sobre este framework se han construido ya varias soluciones diferentes, y una de ellas es la versión 2.0 de OpenHAB. La arquitectura y los conceptos de ambos son muy similares, la mayoría iguales, por lo que a partir de ahora en este trabajo hablaremos indistintamente sobre el uno y el otro. En la parte de desarrollo del trabajo se utilizará Eclipse Smarthome como entorno de desarrollo del “binding” del sistema oneM2M y posteriormente la solución final se probará en un dispositivo real sobre openHAB.

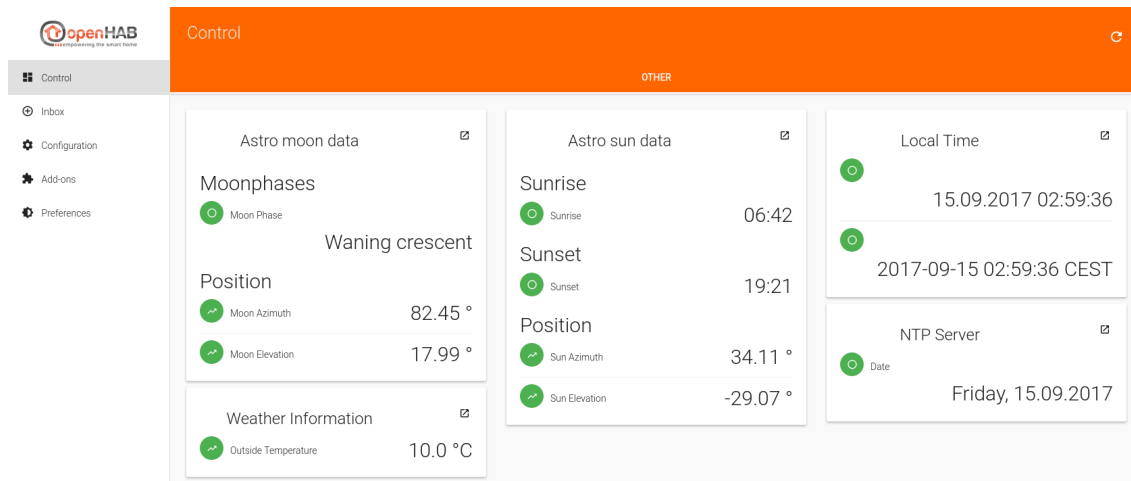


La arquitectura de openHAB está basada en un conjunto de módulos o plug-ins escritos en Java, que corren sobre un tiempo de ejecución OSGi, utilizando para ello dos soluciones de software libre en conjunto: Apache Karaf y Eclipse Equinox. Sobre ellas corren un servidor HTTP (Jetty), el conjunto de pulg-ins que componen el framework de Eclipse Smarthome y los módulos específicos pertenecientes a openHAB. Al sistema que forman los módulos nombrados anteriormente se le pueden añadir una serie de módulos llamados “add-ons” o “bindings” que proporcionan la capacidad de conexión entre openHAB y los dispositivos del hogar. Estos pueden formar parte del proyecto Eclipse SmartHome o de openHAB. Esta diferenciación se debe a ciertos aspectos legales de los módulos de openHAB que no son compatibles con las licencias de software de la Eclipse Foundation.

3.2.2. Metas y servicios proporcionados

Según la Eclipse Foundation, las metas del proyecto Eclipse Smarthome son las siguientes:

- Proporcionar un entorno o framework flexible para la automatización del hogar, que se centre en los casos de uso de este ámbito, por ejemplo, la automatización simple y aspectos de visualización.
- Especificar una serie de puntos de extensión para posibilitar su integración y su uso por servicios de alto nivel. Extender y customizar la solución proporcionada por Eclipse Smarthome debe ser lo más sencillo posible.
- Proporcionar extensiones que implementen los sistemas, protocolos y estándares más importantes, ya que pueden ser muy útiles para varias soluciones de automatización del hogar.
- Proporcionar un entorno de desarrollo y las herramientas necesarias para desarrollar implementaciones de extensiones para el framework. Las herramientas de desarrollo adecuadas pueden ser el apoyo necesario para que haya más contribuciones de código al proyecto y para que se desarrollen nuevas extensiones.
- Proporcionar, junto al entorno de desarrollo, instalaciones de demostración con dispositivos simulados. Aunque lo importante debe ser el framework, también es necesario enseñar al desarrollador el aspecto de una solución final construida con Eclipse Smarthome, para que pueda comprender sus posibilidades.
- El proyecto Eclipse Smarthome está enfocado en permitir la construcción de soluciones para el usuario final en entornos de automatización del hogar muy heterogéneos, es decir, que tienen que enfrentarse a la dificultad de integrarse con diferentes protocolos y estándares. Por lo tanto, este proyecto deberá resolver este problema proporcionando un acceso uniforme y homogéneo a distintos tipos de dispositivos y fuentes de información y facilitar la interacción con ellos y entre ellos.



Eclipse Smarthome y openHAB han sido diseñados para correr en cualquier dispositivo que sea capaz de jorrar la máquina virtual de Java y un entorno de ejecución OSGi, y está especialmente optimizado para correr en dispositivos de recursos limitados como una Raspberry Pi. En el caso de openHab, por ejemplo, se proporciona incluso una distribución de Linux, llamada openHABian, pensada para ser instalada en este tipo de dispositivos, que facilita el uso de openHAB e instala y configura todos los módulos necesarios. Los servicios principales en los que se centran en proveer al usuario son los siguientes: El **manejo de datos** generados por los distintos dispositivos, proveyendo de un modelo homogéneo de la información originada en sistemas heterogéneos. Un **motor de reglas** que permite establecer reglas y rutinas determinadas para poder automatizar acciones sobre varios dispositivos distintos al mismo tiempo, estableciendo condicionales entre los datos de los mismos, como por ejemplo, encender las luces de una habitación cuando un sensor de presencia detecte que alguien ha entrado en ella. **Interfaces de usuario** que permitan el manejo de los dispositivos, la lectura de los datos generados por ellos y la configuración de los mismos, y que se puedan modificar su aspecto y su disposición de una manera declarativa, sin tener que modificar el código fuente de las mismas. Y por último, la **persistencia de los datos**, mediante diferentes extensiones que permiten almacenar todos los datos generados por las fuentes de información y dispositivos conectados en diferentes tipos de bases de datos para su posterior uso o análisis.

3.2.3. Conceptos

Eclipse Smarthome y openHAB se basan en una serie de conceptos para manejar los dispositivos y las fuentes de información que se reúnen en los mismos. Estos conceptos, al ser comunes para todo lo “conectable” a openHAB/ESH, son los que permiten que toda la información se maneje de forma homogénea una vez estén configurados y conectados. Estos conceptos están basados en la estricta diferenciación que realizan estas dos plataformas sobre el mundo físico y el mundo

funcional dentro de su sistema. El mundo físico se refiere a la instalación, configuración y solución de problemas con los dispositivos o las fuentes de información. Por otro lado, el mundo funcional se refiere a los datos generados por los mismos, así como la lógica necesaria para su automatización. Es necesario explicar estos conceptos ya que serán importantes para la parte del desarrollo del binding oneM2M de este trabajo.

El concepto de cosa (**Thing**) se refiere a entidades que pueden ser añadidas físicamente a un sistema y que pueden proporcionar una o varias funciones distintas. Un dispositivo determinado cumple este concepto, pero al igual una cosa puede ser un servicio web o cualquier tipo de fuente de información y funcionalidad. Las cosas tienen parámetros de configuración que, sin ser necesarios para la operación de las mismas, son importantes para la instalación y conexión de las mismas con openHAB. Para el usuario final, las cosas solamente son importantes en esta configuración, pasando a un segundo plano una vez estén funcionando correctamente. Para agregar cosas al sistema, openHAB utiliza una serie de extensiones o add-ons, también llamados bindings, que implementan la conexión de un dispositivo, servicio web o tecnología determinada con el sistema.

Itemname	Description	Command Types
Color	Color information (RGB)	OnOff, IncreaseDecrease, Percent, HSB
Contact	Item storing status of e.g. door/window contacts	OpenClose
DateTime	Stores date and time	
Dimmer	Item carrying a percentage value for dimmers	OnOff, IncreaseDecrease, Percent
Group	Item to nest other items / collect them in groups	-
Number	Stores values in number format	Decimal
Player	Allows to control players (e.g. audio players)	PlayPause, NextPrevious, RewindFastforward
Rollershutter	Typically used for blinds	UpDown, StopMove, Percent
String	Stores texts	String
Switch	Typically used for lights (on/off)	OnOff

El concepto de cosa se refiere al mundo físico. Por otro lado, el mundo funcional, openHAB define el concepto de **Item**. Un item o artículo representa una funcionalidad determinada que es usada por las interfaces de usuario de openHAB o por su motor de reglas. Cada item tiene un estado determinado en cada momento y son usados o accionados mediante eventos. Mientras que las cosas pueden pertenecer a una cantidad ilimitada de tipos, definidos cada uno por los diferentes bindings utilizados para conectarlas, los tipos de items están restringidos a los definidos por el sistema, que son los que podemos ver en la siguiente tabla:

Otro concepto importante es el de canal (**Channel**). Los canales son proporcionados por las cosas, y representan las funciones y los datos que la cosa es capaz de ofrecer al sistema. Por ejemplo, una cosa que represente una bombilla multicolor ofrecería dos canales distintos, uno para encenderla o apagarla y otro para

cambiar el color de la misma. Otro ejemplo sería una cosa que represente la información sobre el tiempo, podría ofrecer varios canales que representen la temperatura, la humedad relativa, la velocidad del viento, etc. Los canales están enlazados a ítems determinados dentro del sistema, por lo tanto, son la entidad que conecta el mundo físico con el mundo funcional. Una vez estos enlaces son realizados, una cosa enviará eventos con la información que genera a un ítem determinado a través de un canal, pudiendo manejar esa información en el sistema. De igual manera, la producción de eventos sobre un ítem, ya sea en las interfaces gráficas o mediante el motor de reglas, provocará una reacción determinada en la cosa a través de un canal.

Por último, también existe el concepto de puente (**Bridge**), que es un tipo especial de cosa. No proveen ninguna funcionalidad directa al sistema, es decir, no tienen ningún canal que exponer, pero actúan como puerta de enlace a otro tipo de cosas. Por lo tanto será necesario añadirlos al sistema para tener acceso a las cosas que están conectadas al mismo.

3.3. OSGi?

4. Diseño

El primer paso para lograr la integración entre un sistema oneM2M y openHAB será el diseño y despliegue de una red o sistema, utilizando OM2M, permita analizar los puntos clave y los procedimientos que necesitaremos para poder controlar dispositivos o aplicaciones oneM2M desde openHAB. Los dispositivos o cosas que

deberá descubrir y controlar openHAB serán dos bombillas simuladas por software en un ordenador.

A continuación se presentarán las dos arquitecturas propuestas y que han sido analizadas y puestas a prueba. La principal diferencia entre ellas es que en la primera el nodo IN-CSE, es decir, el de infraestructura o principal, reside en una máquina situada en la misma res local que el resto de entidades del sistema, mientras que en la segunda este nodo reside en una máquina en la “nube”. El segundo caso sería más realista y más fiel a la arquitectura de oneM2M, pero al mismo tiempo es más complicado para usarlo como entorno de desarrollo o análisis, ya que requiere de un mayor tiempo de configuración y puesta en marcha. Por lo tanto, el análisis se ha realizado básicamente sobre la primera arquitectura, para finalmente realizar las mismas pruebas en ambas para comprobar la validez de la segunda.

4.1. Arquitectura

En ambos casos se utilizará una arquitectura que nos permita el manejo de dos bombillas simuladas por software. Estas estarán formadas por los siguientes nodos y entidades de oneM2M:

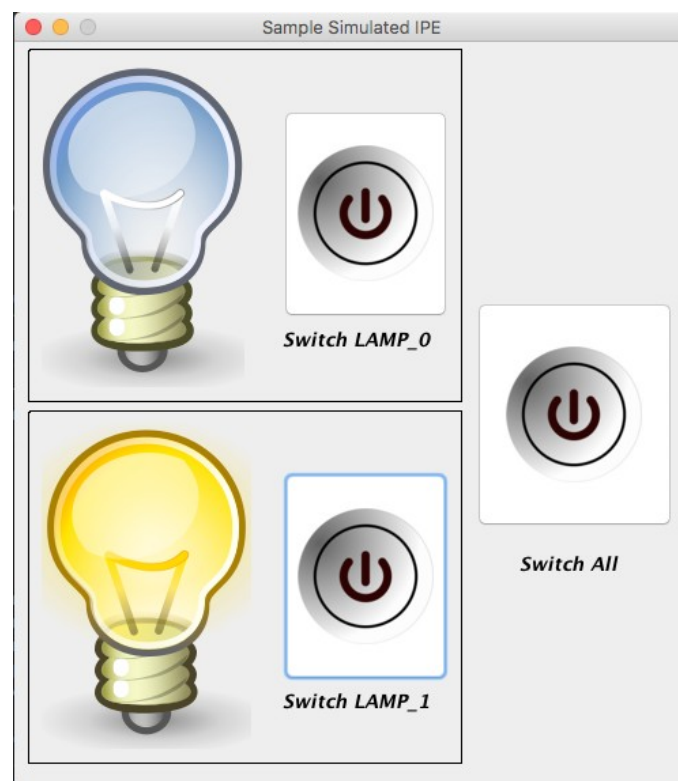
[ESQUEMA]

IN-CSE: se trata del llamado nodo de infraestructura. Únicamente puede existir uno de este tipo en un sistema oneM2M por cada proveedor de servicios. Actuará como puerta de enlace al sistema para todas las aplicaciones externas, ya que la entidad de servicios comunes (CSE) que reside en ella será el nodo padre o raíz de todo el árbol de recursos de los que dispondrá el sistema. Por lo tanto, recibirá cualquier operación sobre entidades del sistema, en este caso mediante HTTP, y será el encargado de redirigirla al nodo que contenga el recurso al que va dirigido la operación, y después se encargará de recibir la respuesta de dicho nodo y de enviársela al emisor de la operación.

MN-CSE: es el nodo intermedio (Middle Node), sobre el que reside una CSE. En este caso se tratará de la puerta de enlace al sistema oneM2M de un hogar, al que se conectarán todas las aplicaciones y dispositivos oneM2M que residen en el mismo para poder tener acceso al exterior del sistema, y que permitirá que aplicaciones o dispositivos que se encuentren fuera de la red local del hogar puedan acceder a ellos a través del nodo de infraestructura. Dentro de este nodo, en la arquitectura diseñada para este trabajo, residirán las diferentes entidades de aplicación que nos permitan el manejo de las dos bombillas situadas en el hogar.

AEs: son las entidades de aplicación del sistema. Estas entidades pertenecen a la capa de aplicación de oneM2M y son las que implementan la lógica de un servicio

M2M determinado que se ofrece al sistema. En nuestra arquitectura, tendremos tres entidades de aplicación que serán registradas en el dominio del nodo MN-CSE, es decir, dentro del hogar. Estas nos permitirán controlar las bombillas. Las dos primeras nos servirán para obtener el estado y controlar cada una de las bombillas por separado, mientras que la tercera nos permitirá realizar las operaciones de encendido y apagado sobre las dos bombillas al mismo tiempo. Por sencillez y ante la falta de otros ejemplos funcionales a utilizar en este proyecto, se ha optado por usar un módulo OSGi proporcionado como ejemplo por el equipo de desarrollo de OM2M consistente en un ejemplo "Interworking Proxy", figura de oneM2M que permite la interoperatividad entre sistemas propietarios o no compatibles con un sistema oneM2M. Su función es Actuar como intermediario entre ambos, traduciendo las operaciones que le llegan a través del sistema a mensajes comandos que el sistema de las dos bombillas puedan entender. El módulo se arrancará desde la consola OSGi del nodo MN-CSE, y este se encargará de simular las dos bombillas mediante software y de registrar las AEs necesarias para su control y monitorizado en el nodo intermedio que reside en la puerta de enlace al hogar.



Al iniciar este módulo, aparecerá en el ordenador en el que reside la MN-CSE una ventana, que podemos observar en la figura X, cuya interfaz gráfica consiste en la representación de dos bombillas con la capacidad de encenderse y apagarse gráficamente, además de tres botones. Dos de estos botones son para controlar cada bombilla por separado, y el tercero cumple la función de encender o apagar las dos al mismo tiempo. Estos botones forman parte del sistema no-oneM2M que contiene las bombillas, es decir, dichas instrucciones u operaciones no pasan en ningún caso por el sistema oneM2M.

4.1.1. Con nodo de infraestructura (IN-CSE) en red local

[ESQUEMA]

Los equipos utilizados en este caso están dentro de la misma red local. Para el nodo IN-CSE se ha utilizado una Raspberry Pi Model 3, conectada directamente al router de la red local mediante Ethernet. Para el nodo MN-CSE se ha utilizado un ordenador portátil, un Macbook Pro, ya que era necesaria una pantalla y una interfaz gráfica para poder utilizar el módulo que registra y simula las bombillas. Uno de los beneficios o funcionalidades de OM2M es precisamente este, que al estar implementado en módulos que corren sobre un tiempo de ejecución OSGi, el cual corre sobre la máquina virtual de Java (Java Virtual Machine – JVM) se puede desplegar de la misma manera en una gran variedad de dispositivos, de diferentes plataformas y con diferentes características. Los únicos requisitos de OM2M son que git esté instalado en el equipo para poder clonar el proyecto desde el repositorio de Eclipse, que Maven 3 o superior esté instalado en el equipo para poder compilar los módulos del proyecto desde el código fuente previamente clonado y que el equipo tenga una versión de la máquina virtual de Java igual o superior a la 1.7.

[TABLA CARACTERÍSTICAS RASPBERRY]

[TABLA CARACTERÍSTICAS MACBOOK]

Para el correcto funcionamiento del sistema oneM2M, necesitamos que estas dos máquinas tengan una dirección IP fija dentro de la red local. Si esta no fuese fija, deberían cambiarse los archivos de configuración de cada uno de los nodos para que puedan conectarse entre si. Por eso, se ha configurado dos direcciones fijas para estas dos máquinas en el router del hogar mediante reserva DHCP. DHCP (Dynamic Host Configuration Protocol) es un protocolo que utilizan los routers para asignar una dirección IP determinada a las entidades conectadas a su red local, de un rango de direcciones dinámicas predeterminado. Cuando un nuevo cliente se conecta a la red solicita una dirección IP para él mismo, haciendo una petición DHCP al router, y el router le asigna una dirección determinada. El problema, en el caso de nuestra arquitectura, es que cada vez que se conectan los nodos a la red el router les asigna una dirección IP que puede ser distinta a la anterior. Por lo tanto, se ha configurado una reserva DHCP en el router para las dos máquinas a tratar, que consiste en la asociación de la dirección MAC de la interfaz de red por la que se conectan a la red local del router con una dirección IP fija determinada. A partir de este momento cada vez que los nodos se conecten a la red local el router les asignará la misma dirección IP.

MAC Address	IP Address	Remove
b8:27:eb:da:13:6c	192.168.1.42	<input type="checkbox"/>
28:cf:da:eb:05:18	192.168.1.40	<input type="checkbox"/>

El primer paso para desplegar la red será descargar la última versión de OM2M en cada uno de los dispositivos que formarán parte de la red. Para esto, es necesario clonar el proyecto, situado en un servidor git de control de versiones de la Eclipse Foundation. Para evitar fallos o errores inesperados, se ha utilizado la última versión estable, y se ha evitado utilizar versiones de las ramas de desarrollo del proyecto.

Al clonar el repositorio, nos encontramos con la siguiente estructura de directorios:

org.eclipse.om2m

- ├─ org.eclipse.om2m.binding.coap
- ├─ org.eclipse.om2m.binding.http
- ├─ org.eclipse.om2m.binding.mqtt
- ├─ org.eclipse.om2m.binding.service
- ├─ org.eclipse.om2m.commons
- ├─ org.eclipse.om2m.commons.logging
- ├─ org.eclipse.om2m.core
- ├─ org.eclipse.om2m.core.service
- ├─ org.eclipse.om2m.datamapping.jaxb
- ├─ org.eclipse.om2m.datamapping.service
- ├─ org.eclipse.om2m.interworking.service
- ├─ org.eclipse.om2m.ipe.sample
- ├─ org.eclipse.om2m.persistence.eclipselink
- ├─ org.eclipse.om2m.persistence.service
- ├─ org.eclipse.om2m.site.asn-cse
- ├─ org.eclipse.om2m.site.in-cse
- ├─ org.eclipse.om2m.site.mn-cse
- ├─ org.eclipse.om2m.webapp.resourcesbrowser.json
- ├─ org.eclipse.om2m.webapp.resourcesbrowser.xml

Cada uno de estos directorios corresponde a uno de los diferentes módulos OSGi que componen OM2M. Algunos de ellos pertenecen al núcleo de la implementación de oneM2M, como `org.eclipse.om2m.core` o `org.eclipse.om2m.commons`, y son necesarios en cualquier aplicación OM2M, mientras que otros son opcionales. Cabe destacar los módulos `org.eclipse.om2m.binding.coap`, `org.eclipse.om2m.binding.http` y

org.eclipse.om2m.binding.mqtt, que hacen de enlace con los protocolos subyacentes necesario para el envío y la recepción de las operaciones sobre los recursos del sistema por parte de los diferentes nodos. Hacen un mapeo de estas a los tipos de mensajes de estos tres protocolos de acuerdo con las especificaciones de oneM2M a este respecto. Además, serán importantes para nuestro sistema los módulos org.eclipse.om2m.site.in-cse, org.eclipse.om2m.site.mn-cse, que son las implementaciones de los dos tipos de nodos oneM2M, IN-CSE y MN-CSE, que formarán parte del sistema. Por último, también cabe destacar el módulo org.eclipse.om2m.ipe.sample, que es el módulo mencionado anteriormente que proporciona la implementación de las dos bombillas simuladas por software.

El siguiente paso para poder arrancar la CSE en cada uno de los nodos será compilar los módulos, ya que el proyecto que ha sido clonado en ellos solo contiene el código fuente de los mismos, y necesitamos construir los módulos OSGi en archivos con extensión .jar. El proyecto OM2M utiliza Maven para esta tarea. Maven es una herramienta utilizada principalmente en proyectos escritos en Java, que permite la gestión y la automatización de la compilación de proyectos y la gestión de dependencias (librerías, módulos, etc.) de los mismos. La información de los pasos a seguir para compilar un proyecto, como la versión de Java a utilizar, y las dependencias del mismo son definidas en el archivo “pom.xml”, situado en el directorio raíz del proyecto, que se denomina POM (Project Object Model). En el caso de OM2M, existe un POM global, situado en la carpeta raíz, que describe el proyecto y los diferentes módulos o bundles OSGi que lo componen. Después, cada módulo posee su propio POM con las particularidades de dependencias y compilación de cada uno. Para compilar todos los módulos del proyecto es necesario abrir una terminal en la carpeta raíz del proyecto OM2M previamente clonado e introducir el comando “mvn clean install”. La opción clean indica que se elimine cualquier archivo generado por una compilación anterior, mientras que la opción install indica que se descarguen las dependencias necesarias y que se compilen los módulos del proyecto.

```
[INFO] Executed tasks
[INFO] -----
[INFO] Reactor Summary:
[INFO]
[INFO] org.eclipse.om2m :: parent ..... SUCCESS [ 0.368 s]
[INFO] org.eclipse.om2m :: commons ..... SUCCESS [ 1.343 s]
[INFO] org.eclipse.om2m :: logging fragment ..... SUCCESS [ 0.035 s]
[INFO] org.eclipse.om2m :: binding service ..... SUCCESS [ 0.045 s]
[INFO] org.eclipse.om2m :: core service ..... SUCCESS [ 0.048 s]
[INFO] org.eclipse.om2m :: binding http ..... SUCCESS [ 0.205 s]
[INFO] org.eclipse.om2m :: binding coap ..... SUCCESS [ 0.087 s]
[INFO] org.eclipse.om2m :: data mapping service ..... SUCCESS [ 0.037 s]
[INFO] org.eclipse.om2m.binding.mqtt ..... SUCCESS [ 0.204 s]
[INFO] org.eclipse.om2m :: persistence service ..... SUCCESS [ 0.053 s]
[INFO] org.eclipse.om2m :: persistence eclipselink ..... SUCCESS [ 0.460 s]
[INFO] org.eclipse.om2m :: datamapper jaxb ..... SUCCESS [ 0.155 s]
[INFO] org.eclipse.om2m :: webapp resourcesbrowser xml .... SUCCESS [ 0.068 s]
[INFO] org.eclipse.om2m :: webapp resourcesbrowser json ... SUCCESS [ 0.063 s]
[INFO] org.eclipse.om2m :: interworking service ..... SUCCESS [ 0.133 s]
[INFO] org.eclipse.om2m :: core ..... SUCCESS [ 0.212 s]
[INFO] org.eclipse.om2m :: ipe sample ..... SUCCESS [ 0.238 s]
[INFO] org.eclipse.om2m :: asn product ..... SUCCESS [ 2.274 s]
[INFO] org.eclipse.om2m :: in product ..... SUCCESS [ 2.072 s]
[INFO] org.eclipse.om2m :: mn product ..... SUCCESS [ 1.944 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
```

Una vez clonado el proyecto y compilados todos los módulos en ambas máquinas, es necesario proceder a la configuración de ambas.

4.1.1.1. Configuración y arranque del nodo IN-CSE

El primer paso para desplegar el sistema consiste en el arranque del nodo de infraestructura. Antes de proceder al arranque, es necesario editar los parámetros de configuración del mismo para adecuarlos a nuestras necesidades. Estos parámetros son varios, algunos han tenido que cambiarse para adecuar el nodo a la arquitectura propuesta y otros se han dejado como estaban por defecto. A continuación se nombrarán y explicarán los parámetros más importantes de entre estos para el despliegue. Estos parámetros se editan en un fichero de configuración llamado “config.ini”, que se encuentra dentro del directorio del módulo de implementación del nodo IN-CSE.

- **org.eclipse.om2m.cseType=IN-CSE**

Este parámetro indica el tipo de CSE que se va a desplegar en el nodo. En este caso, como va a consistir en un nodo de infraestructura, se ha fijado un valor igual a “IN-CSE” para este campo.

- **org.eclipse.om2m.cseBaseId=in-cse**

Este parámetro indica el identificador único de la CSE desplegada en el nodo. Se corresponde con el identificador determinado en las especificaciones de oneM2M llamado CSE-ID. Este identificador tiene que ser globalmente único dentro de un sistema oneM2M, dentro de los dominios de un proveedor de servicios M2M., ya que sirve para formar las URIs (Uniform Resource Identifier) que identifican a los diferentes recursos que forman parte de la CSE y para poder realizar operaciones sobre los mismos.

- **org.eclipse.om2m.cseBaseName=in-local**

El parámetro cseBaseName representa el nombre de la CSE que reside en el nodo. Es otra forma de identificarla, de una manera más textual, y no tienen por qué ser únicos dentro del sistema oneM2M. En este caso se ha elegido el nombre “in-local”, ya que el nodo de infraestructura reside en la red local de un hogar.

- **org.eclipse.om2m.cseBaseAddress=192.168.1.42**

Este parámetro indica la dirección IP de la máquina en la que va a correr el nodo. En este caso se ha introducido la dirección que había sido previamente fijada para este nodo mediante una reserva DHCP.

- **org.eclipse.om2m.adminRequestingEntity=admin\;cherran**

El parámetro de configuración `adminRequestingEntity` representa la combinación de usuario y contraseña de administración del nodo. Estas credenciales son los que se deben usar para acceder a la interfaz web de nodo proporcionada por OM2M, y habrá que introducirlos en las cabeceras HTTP de las operaciones de administración que se realicen sobre el nodo. En este caso se ha configurado “admin” como nombre del usuario administrador y “cherran” como su contraseña.

- **org.eclipse.om2m.cseBaseProtocol.default=http**

Este parámetro indica al nodo OM2M el protocolo por defecto que debe utilizar para el envío y la recepción de mensajes y operaciones de oneM2M. OM2M proporciona módulos para el uso de HTTP, CoAP y MQTT. En la arquitectura propuesta se va a utilizar HTTP como protocolo de intercambio de mensajes.

- **org.eclipse.equinox.http.jetty.http.port=8080**

Este parámetro de configuración indica el puerto por el que el servidor web del nodo OM2M tiene que escuchar la recepción de peticiones HTTP del exterior, así como enviar las que procedan del mismo nodo. Este parámetro es importante ya que se puede utilizar para correr múltiples nodos OM2M en la misma máquina, haciendo que estos escuchen puertos diferentes. Para el nodo de infraestructura se ha elegido el puerto 8080.

- **org.eclipse.om2m.webInterfaceContext=/webpage**

Por último, el parámetro `webInterfaceContext` indica la ruta dentro del servidor web por la que se accede a la interfaz web de administración del nodo de infraestructura. En este caso el parámetro está configurado como “/webpage”, por lo tanto para acceder a esta interfaz web se deberá introducir la dirección “http://192.168.1.42:8080/webpage” en un navegador web. El resto de peticiones HTTP, es decir, las diferentes operaciones de oneM2M se deberán hacer a través del punto de acceso “http://192.168.1.42:8080/~/[recurso_solicitado]”.

Una vez configurado el nodo de infraestructura correctamente, se ha de arrancar el nodo mediante la ejecución de un script llamado “start.sh” que se incluye dentro del directorio del nodo. Al ejecutarlo se inicia una consola OSGi que va arrancando cada uno de los módulos que son necesarios en el nodo. Una vez estén todos arrancados, es posible acceder a la interfaz web insertando en nombre de usuario y contraseña de administración configurados. Dentro de la interfaz web podemos ver el árbol de recursos y obtener la información de cada uno de ellos.



username:

password

Login

4.1.1.2. Configuración y arranque del nodo MN-CSE

El siguiente paso para la puesta en marcha de la arquitectura propuesta es la configuración y arranque de la CSE del nodo intermedio (MN-CSE). Para ello, como en el caso anterior, se debe primero determinar los parámetros de configuración necesarios en el archivo "config.ini". Los parámetros explicados en el apartado anterior también son importantes en este caso, y a continuación se muestran con sus correspondientes valores:

- **org.eclipse.om2m.cseType=MN-CSE**
- **org.eclipse.om2m.cseBaseId=mn-cse1**
- **org.eclipse.om2m.cseBaseName=mn-home**
- **org.eclipse.om2m.cseBaseAddress=192.168.1.40**
- **org.eclipse.om2m.adminRequestingEntity=admin\cherran**
- **org.eclipse.om2m.cseBaseProtocol.default=http**
- **org.eclipse.equinox.http.jetty.http.port=8383**
- **org.eclipse.om2m.webInterfaceContext=/webpage**

Además de estos parámetros de configuración, que también estaban presentes en el nodo de infraestructura, la MN-CSE necesita de otros adicionales que le indican la información que necesita para registrarse en el nodo de infraestructura. Estos parámetros indican el identificador, el nombre, la dirección IP y el puerto de HTTP del nodo de infraestructura, y son los siguientes:

- **org.eclipse.om2m.remoteCseId=in-cse**
- **org.eclipse.om2m.remoteCseName=in-local**
- **org.eclipse.om2m.remoteCseAddress=192.168.1.42**
- **org.eclipse.om2m.remoteCsePort=8080**

Con todos los parámetros de configuración correctamente anunciados, se inicia el nodo de la misma manera que el nodo de infraestructura. Una vez iniciado, el nodo se registra en el nodo de infraestructura, y así empieza a formar parte de su árbol de recursos. Todos los recursos del nodo intermedio serán accesibles desde ese momento mediante peticiones HTTP al servidor web del nodo de infraestructura, por el puerto 8080, y este se encargará de redirigirlas al nodo intermedio. De igual manera, los recursos del nodo intermedio son también accesibles mediante su propio servidor web en el puerto 8383, lo que permitiría a otras entidades inferiores registrarse en él. A partir de este momento también podremos acceder a la interfaz web del nodo intermedio desde la interfaz del nodo de infraestructura.

Una vez registrado el nodo intermedio en el nodo de infraestructura, para completar la arquitectura propuesta, se ha de arrancar en la consola OSGi del nodo intermedio el módulo que implementa las dos bombillas simuladas por software. Para ello, se inserta en la consola el comando “ss”, que devuelve un listado con todos los módulos OSGi instalados, indicando su nombre, estado y identificador. Para arrancar el módulo deseado, se ejecuta el comando start seguido del identificador del módulo a arrancar. En el momento que se ejecuta aparece en pantalla la interfaz gráfica con las dos bombillas y el módulo registra las tres entidades de aplicación (AEs) mencionadas anteriormente. Desde este momento será posible acceder a la interfaz web del nodo intermedio a través del nodo de infraestructura para ver el árbol de recursos con las tres AEs registradas y todos los datos de las mismas.

[Logout](#)

OM2M CSE Resource Tree

<http://192.168.1.42:8080/~mn-cse1/CAE755426155>



```

- mn-home
  - acp_admin
  - LAMP_0
    - DATA
    - DESCRIPTOR
  - LAMP_1
    - DESCRIPTOR
      - cin_33615742
    - DATA
      - cin_665725397
  - LAMP_ALL
    - DESCRIPTOR
      - cin_270461326
  - in-local
  
```

Attribute	Value
ty	2
ri	/mn-cse1/CAE755426155
pi	/mn-cse1
ct	20170916T190408
lt	20170916T190408
acpi	<div>AccessControlPolicyIDs</div> <div>/mn-cse1/acp-673726391</div>
et	20180916T190408
api	LAMP_0
aei	CAE755426155
poa	<div>Point Of Access</div> <div>sample</div>
rr	true

4.1.2. Con nodo de infraestructura (IN-CSE) en la nube

[ESQUEMA]

Esta arquitectura es conceptualmente igual que la anterior, con la única diferencia de la ubicación del nodo de infraestructura. A diferencia de la anterior, que lo corría una Raspberry Pi en una red local, en este caso se corre en una máquina virtual en la nube de Amazon. El servicio Amazon Elastic Compute Cloud (Amazon EC2) es un servicio que ofrece la empresa norteamericana que permite a sus usuarios tener acceso a máquinas virtuales para poder ejecutar sus aplicaciones o alojar sus páginas web. Este sistema ofrece muchos beneficios frente al paradigma anterior de la gestión de sistemas, en los que se utilizaban máquinas físicas dedicadas. Con este servicio es posible crear instancias de máquinas virtuales en apenas segundos, con una capacidad de red, computación o almacenamiento casi infinita. Además se complementa con otros servicios que Amazon ofrece en la nube como bases de datos o servicios para Big Data. Estos servicios se pagan por uso, dependiendo del tiempo del tiempo de funcionamiento de las máquinas virtuales, la cantidad de datos generados, transmitidos o almacenados y otros factores. Para este trabajo, se ha utilizado una máquina virtual de la capa gratuita de Amazon, que nos proporciona una única instancia t2.micro. Este tipo de instancia es la más simple y con menos recursos, pero será suficiente para desplegar en ella el nodo de infraestructura IN-CSE.



[CARACTERÍSTICAS EC2 T2.MICRO]

Antes de poder desplegar el sistema, serán necesarios ciertos ajustes en la configuración de red de la máquina t2.micro y del router del hogar al que están conectado, en red local, el nodo intermedio. Las máquinas virtuales de Amazon EC2 poseen una dirección IP pública por la que se puede acceder a ellas a través de internet. Esta IP puede ser fija o dinámica. Para desplegar el sistema lo mejor sería que esta fuese fija, para no tener que cambiar los archivos de configuración de los nodos cada vez que esta cambiase. El problema es que este servicio es de pago, así que se optará para esta arquitectura por una IP dinámica, que forzosamente cambiará cada vez que la máquina virtual sea parada o pase un cierto tiempo. Durante las pruebas que se han realizado en este trabajo ha sido necesario ajustar la configuración del nodo intermedio varias veces para que pudiese encontrar al nodo de infraestructura y registrarse el él. Al hacerse esta conexión a través de la red pública, por Internet, lo mismo sucede con la IP pública que tiene el router del hogar, en el caso probado conectado a la red de Movistar en Madrid. También existe la posibilidad de solicitar a Movistar una IP pública fija para este caso, pero es un servicio de pago.

En el caso de la máquina virtual t2.micro, es necesario modificar el grupo de seguridad asociado a esta máquina. Un grupo de seguridad consiste en un conjunto de reglas de acceso a red que limitan el acceso a los puertos de la interfaz de red pública de la instancia, por motivos de seguridad. Por defecto, el grupo de seguridad está configurado para que permita todo el tráfico saliente, pero para el tráfico entrante solamente permite el tráfico a través del puerto 22 (TCP), para controlar remotamente la instancia mediante SSH, que se emite desde la IP pública desde la que se creó la instancia. Para poder tener acceso permanente a la máquina ha sido necesario editar esta regla para permitir el tráfico entrante por el puerto 22 desde cualquier IP pública, ya que la IP desde el ordenador por el cual se ha accedido para realizar las pruebas de este trabajo es dinámica. Para mayor seguridad, la única manera de autenticarse al conectarse por SSH a una máquina de Amazon es mediante un certificado digital único que se genera al crear una nueva instancia, y al que solamente tiene acceso el creador de la misma. Además de modificar esta norma, se ha añadido otra para permitir el tráfico entrante por el puerto 8080 desde cualquier dirección de IP pública, para que el nodo intermedio tenga acceso para registrarse en el nodo de infraestructura y para poder acceder a los recursos de este desde cualquier parte del mundo.

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ
Custom TCP Rule	TCP	8080	0.0.0.0/0
Custom TCP Rule	TCP	8080	::/0
SSH	TCP	22	83.60.133.149/32

En el router del hogar también se ha tenido que modificar la configuración para que el nodo intermedio sea accesible desde el exterior de la red local del hogar. Para ello, se ha tenido que redirigir el puerto externo 8383 del router a la dirección IP privada que en la arquitectura anterior hemos fijado, en su puerto 8383. Además, para controlar al nodo desde el exterior para su administración, se ha cambiado el puerto del servicio SSH de la misma al puerto 19127 para intentar evitar posibles ataques y se ha redirigido este mismo puerto externo en el router al nodo intermedio. Estas redirecciones se han realizado modificando la configuración NAT (Network Address Translation) del router.

Server Name	External Port Start	External Port End	Protocol	Internal Port Start	Internal Port End	Server IP Address	WAN Interface	Remove
OM2M MN-CSE macbook	8383	8383	TCP	8383	8383	192.168.1.40	ppp0.1	
SSH macbook	19127	19127	TCP	19127	19127	192.168.1.42	ppp0.1	

Una vez realizados estos cambios ya es posible desplegar el sistema oneM2M propuesto por esta arquitectura. Para ello, se clonará el proyecto OM2M desde el repositorio git de Eclipse y se compilarán los módulos del mismo de la misma manera que en el apartado anterior. También se deberán modificar los archivos “config.ini” de ambos nodos como en la anterior arquitectura, pero teniendo en cuenta la nueva topología de red.

Los parámetros de configuración más relevantes del nodo de infraestructura que hemos desplegado en la máquina virtual t2.micro son los siguientes:

- **org.eclipse.om2m.cseType=IN-CSE**
- **org.eclipse.om2m.cseBaseId=in-cse**
- **org.eclipse.om2m.cseBaseName=in-cloud**
- **org.eclipse.om2m.cseBaseAddress=52.210.48.121** (Dirección IP pública de la máquina virtual, que al ser dinámica puede cambiar a lo largo del tiempo)
- **org.eclipse.om2m.adminRequestingEntity=admin\;cherran**
- **org.eclipse.om2m.cseBaseProtocol.default=http**
- **org.eclipse.equinox.http.jetty.http.port=8080**
- **org.eclipse.om2m.webInterfaceContext=/webpage**

En el caso del nodo intermedio desplegado en en MacbookPro conectado a la red del hogar, los parámetro de configuración serán los siguientes:

- **org.eclipse.om2m.cseType=MN-CSE**
- **org.eclipse.om2m.cseBaseId=mn-cse1**
- **org.eclipse.om2m.cseBaseName=mn-home**
- **org.eclipse.om2m.cseBaseAddress=83.60.133.14** (Dirección IP pública del router, que al ser dinámica puede cambiar a lo largo del tiempo)
- **org.eclipse.om2m.adminRequestingEntity=admin\;cherran**
- **org.eclipse.om2m.cseBaseProtocol.default=http**
- **org.eclipse.equinox.http.jetty.http.port=8383**
- **org.eclipse.om2m.webInterfaceContext=/webpage**
- **org.eclipse.om2m.remoteCseId=in-cse**
- **org.eclipse.om2m.remoteCseName=in-local**
- **org.eclipse.om2m.remoteCseAddress=52.210.48.121** (Dirección IP pública de la máquina virtual, que al ser dinámica puede cambiar a lo largo del tiempo)
- **org.eclipse.om2m.remoteCsePort=8080**

Una vez configurados correctamente ambos nodos, se prueba que se inicien completamente y que podemos acceder a los recursos que residen en la CSE del

nodo intermedio desde la interfaz del nodo de infraestructura para asegurar que el sistema oneM2M propuesto por esta segunda arquitectura ha sido desplegado correctamente.

4.2 Análisis

Para poder implementar el control de las entidades de aplicación oneM2M que incluyen las dos arquitecturas propuestas desde un complemento desarrollado con tal motivo para openHAB, previamente será necesario el estudio de las operaciones concretas a realizar a través de peticiones HTTP al nodo de infraestructura de la red por parte del complemento. En este apartado del trabajo, se realizará un análisis de dichas peticiones y el motivo de las mismas.

El complemento de openHAB a desarrollar deberá seguir varios pasos para poder controlar las bombillas, todos ellos mediante el uso de las funciones de servicio común que proporciona la capa de servicios comunes (CSE) desplegada en el sistema oneM2M que se ha definido e implementado en los apartados anteriores. El primer paso necesario será el registro del propio complemento como una entidad de aplicación en el sistema oneM2M, para que pueda ser reconocida por el resto de entidades e identificada por ellas al recibir operaciones sobre sus recursos que sean originadas en el complemento. De esta manera las entidades pueden controlar el tipo de operaciones que puede realizar el complemento y sobre qué recursos determinados puede solicitarlos. Por lo tanto, después de que el nodo de infraestructura IN-CSE anuncie el registro en él del complemento de openHAB, el nodo intermedio deberá cambiar la política de acceso asociada a las entidades de aplicación que controlan las bombillas residentes en su árbol de recursos, para permitir ciertas operaciones sobre ellas a la entidad de aplicación formada por el complemento. Una vez el complemento de openHAB tenga garantizado el acceso a estos recursos deberá ser capaz, mediante peticiones realizadas al nodo de infraestructura, de descubrir la existencia de la puerta de acceso al hogar (el nodo intermedio), averiguar los recursos que esta ofrece y obtener el control de sus entidades de aplicación.

En los siguientes apartados se realizará una descripción y análisis de estos pasos a dar por el complemento a desarrollar. Para ello se ha utilizado la aplicación POSTMAN, una herramienta gratuita que consiste en un cliente gráfico para realizar peticiones HTTP a cualquier API REST, facilitando la creación, gestión y guardado de las peticiones y permitiendo el análisis detallado de las respuestas obtenidas.

4.2.1. Aspectos comunes de las peticiones HTTP a realizar

Como ya se ha comentado anteriormente en este trabajo, las operaciones sobre los recursos de un sistema oneM2M siguen el modelo de petición-respuesta, consistiendo en diferentes acciones que se pueden realizar, como obtener, actualizar o eliminar, sobre todos los recursos que componen el sistema. La especificación de estas operaciones ha sido recogida por los integrantes del grupo de estandarización oneM2M en el documento (TS-0004 – Service Layer Core Protocol), que define el formato y el flujo de cualquier operación permitida por la Capa de Servicios Comunes (CSE). Esta especificación es independiente del protocolo subyacente que se utilice para el intercambio de las mismas. Se puede utilizar cualquier protocolo de intercambio de mensajes siempre que cumpla los requerimientos definidos por este documento, y que se haya definido por parte de oneM2M una equivalencia o mapeo entre sus operaciones y dicho protocolo. Hasta el momento de realización de este trabajo, oneM2M ha publicado cuatro especificaciones con este fin, para poder realizar operaciones en un sistema oneM2M mediante HTTP, CoAP, MQTT y Websocket. El proyecto Eclipse OM2M utilizado en este trabajo para desplegar el sistema ha implementado tres de estas especificaciones hasta el momento, quedando pendiente la del protocolo Websocket. Para este trabajo se ha utilizado HTTP para el intercambio de mensajes, que está definido en el documento (TS-009 – HTTP Protocol Binding).

Al utilizar HTTP, primero deberemos tener en cuenta los roles de cada una de las entidades oneM2M desplegadas en las arquitecturas propuestas en una comunicación de este tipo. Las entidades de aplicación, en este caso el complemento de openHAB, cumplen la función del cliente en una comunicación HTTP. Realizará peticiones equivalentes a operaciones oneM2M sobre las entidades de aplicación registradas en la capa de servicios común del nodo intermedio, por lo que será equivalente a la figura de cliente en HTTP. El rol del servidor, por lo tanto, será cumplido por el nodo intermedio, ya que responderá al cliente con los resultados de las operaciones solicitadas sobre sus recursos. Por último, el nodo intermedio se encargará de recibir en primera instancia las peticiones del cliente y redirigirlas al nodo intermedio donde residen los recursos, actuando por lo tanto como un proxy HTTP.

El primer punto importante en el uso de HTTP será la equivalencia con los tipos de operaciones que oneM2M permite realizar sobre sus recursos con el tipo de operaciones que permite realizar el protocolo HTTP. Las operaciones posibles en oneM2M sobre los recursos son la creación, lectura, actualización, eliminación y notificación. Se especifica, por lo tanto, una equivalencia entre estos tipos de operación y los métodos HTTP necesarios para realizar estas peticiones, que podemos ver en la **tabla X**. El método POST de HTTP se utiliza para realizar dos de estas operaciones, por lo que será necesario que el receptor de la petición diferencie entre ambos al recibirla. La diferencia entre estas dos operaciones consiste en la inclusión del tipo de contenido oneM2M (Resource Type -ty) en la cabecera “Content-Type” de la petición HTTP en el caso de la creación de recursos. Más adelante se explicará este término en detalle.

Operación oneM2M	Método HTTP
Create	POST
Retrieve	GET
Update	PUT
Delete	DELETE
Notify	POST

El recurso al que va dirigido la operación se determina como el recurso que se pide en la petición HTTP, es decir, la URL que sigue al método de petición en la primera línea de cualquier petición HTTP. Los recursos se pueden identificar de forma absoluta en todo el dominio de un proveedor de servicios M2M, es decir, incluyendo en la URI del recurso todos los nodos padre que tiene en dicho dominio, o de forma relativa al los recursos del nodo al que se realiza la petición, en el que se excluyen de la URI las entidades que son así mismo nodos padre del nodo receptor. Para la primera se utiliza después de la dirección del servidor web receptor la cadena de caracteres “/_” seguido de identificador de las entidades y el recurso de manera jerárquica. En segundo caso, el que se va a utilizar en este trabajo para realizar todas las peticiones, se utiliza la cadena de caracteres “/~”. En el caso particular de alguna petición de una función común que lo requiera, como es el caso del descubrimiento, se pueden añadir cadenas de consulta o *query strings* al final de la URI que representa al recurso. Estos consisten en pares de nombre de un parámetro y su valor, separadas por el carácter “&”, y determinan parámetros que pueden determinar de menor manera la operación. Por ejemplo, en el caso de la función de descubrimiento se utilizan para filtrar según estos parámetros el resultado deseado de una búsqueda de recursos en el sistema. Por último, como en cualquier petición HTTP, la primera línea de la petición debe terminar con la versión de HTTP que el cliente desea utilizar. La especificación de oneM2M para el mapeo de HTTP con sus operaciones solamente es compatible con su versión 1.1, por lo que todas las peticiones que realizaremos al nodo de infraestructura deberán fijar esta versión. En los siguientes ejemplos no se incluirá este campo, ya que siempre será el mismo.

En las respuestas a las operaciones oneM2M, se incluye un código de respuesta (Response Status Code – RSC) determinado por el nodo receptor de la petición, en forma de código numérico, que indica el resultado de la operación. HTTP tiene un sistema de códigos de respuesta similar, por lo que se ha establecido una equivalencia entre estos y los RSCs definidos por oneM2M. Los códigos de oneM2M son más específicos que los de HTTP, y por lo tanto el mismo código de respuesta HTTP se utiliza en el caso de varios RSCs de oneM2M diferentes. Por este motivo, la especificación de oneM2M define la cabecera de HTTP “X-M2M-RSC” que debe ir

incluida en todas las respuestas a las operaciones determinando el código de respuesta de oneM2M. Esta equivalencia queda definida en la **tabla X**.

Además de la cabecera de HTTP para indicar el RSC en las respuestas, varias cabeceras se pueden o deben incluir para el correcto funcionamiento de las operaciones. A continuación se citan las más importantes:

Accept: el emisor de la operación puede incluir esta cabecera en la petición para determinar el formato de datos que espera recibir en la respuesta de la misma. Los posibles valores para este campo son “application/xml” y “application/json”, los dos tipos de estructuras de datos que están determinados en las especificaciones de oneM2M. En el caso de OM2M, las entidades implementadas por este aceptan cualquiera de los dos tipos de datos, pero el cuerpo de sus respuestas siempre se rellena en formato XML, por lo que en el caso de este trabajo las peticiones siempre llevarán esta cabecera con el valor “application/xml”

Content-Type: cualquier petición o respuesta de oneM2M debe de ir acompañada de esta cabecera, que indica el formato de los datos incluidos en el cuerpo de la petición, siempre que lleve algún tipo de dato en su cuerpo. Sus valores también pueden ser “application/xml o application/json”. Como se ha mencionado antes, en el caso de la operación de creación de recursos, este campo también debe incluir el tipo de recurso oneM2M que desea crear. Por ejemplo, una cabecera Content-Type con un valor igual a “application/xml;ty=2” indicaría que se desea crear un recurso de tipo dos (entidad de aplicación) con los datos que incluye la petición en su cuerpo en formato XML.

oneM2M Response Status Codes	HTTP Status Codes
1000 (ACCEPTED)	202 (Accepted)
2000 (OK)	200 (OK)
2001 (CREATED)	201 (Created)
4000 (BAD_REQUEST)	400 (Bad Request)
4004 (NOT_FOUND)	404 (Not Found)
4005 (OPERATION_NOT_ALLOWED)	405 (Method Not Allowed)
4008 (REQUEST_TIMEOUT)	408 (Request Timeout)
4101 (SUBSCRIPTION_CREATOR_HAS_NO_PRIVILEGE)	403 (Forbidden)
4102 (CONTENTS_UNACCEPTABLE)	400 (Bad Request)
4103 (ACCESS_DENIED)	403 (Forbidden)
4104 (GROUP_REQUEST_IDENTIFIER_EXISTS)	409 (Conflict)
4105 (CONFLICT)	409 (Conflict)
5000 (INTERNAL_SERVER_ERROR)	500 (Internal Server Error)
5001 (NOT_IMPLEMENTED)	501 (Not Implemented)
5103 (TARGET_NOT_REACHABLE)	404 (Not Found)
5105 (NO_PRIVILEGE)	403 (Forbidden)
5106 (ALREADY_EXISTS)	403 (Forbidden)
5203 (TARGET_NOT_SUBSCRIBABLE)	403 (Forbidden)
5204 (SUBSCRIPTION_VERIFICATION_INITIATION_FAILED)	500 (Internal Server Error)
5205 (SUBSCRIPTION_HOST_HAS_NO_PRIVILEGE)	403 (Forbidden)
5206 (NON_BLOCKING_REQUEST_NOT_SUPPORTED)	501 (Not Implemented)
5207 (NOT_ACCEPTABLE)	406 (Not Acceptable)
6003 (EXTERNAL_OBJECT_NOT_REACHABLE)	404 (Not Found)
6005 (EXTERNAL_OBJECT_NOT_FOUND)	404 (Not Found)
6010 (MAX_NUMBER_OF_MEMBER_EXCEEDED)	400 (Bad Request)
6011 (MEMBER_TYPE_INCONSISTENT)	400 (Bad Request)
6020 (MANAGEMENT_SESSION_CANNOT_BE_ESTABLISHED)	500 (Internal Server Error)
6021 (MANAGEMENT_SESSION_ESTABLISHMENT_TIMEOUT)	500 (Internal Server Error)
6022 (INVALID_CMDTYPE)	400 (Bad Request)
6023 (INVALID_ARGUMENTS)	400 (Bad Request)
6024 (INSUFFICIENT_ARGUMENT)	400 (Bad Request)
6025 (MGMT_CONVERSION_ERROR)	500 (Internal Server Error)
6026 (CANCELLATION_FAILED)	500 (Internal Server Error)
6028 (ALREADY_COMPLETE)	400 (Bad Request)
6029 (COMMAND_NOT_CANCELLABLE)	400 (Bad Request)

X-M2M-Origin: esta cabecera está presente en las peticiones y en las respuestas y sirve para identificar a la entidad oneM2M en la que se origina el mensaje. Su propósito principal es que el receptor de una petición sobre un recurso identifique de qué entidad le llega la petición y si esta tiene los permisos suficientes para poder realizar la operación deseada sobre el recurso determinado.

X-M2M-RI: consiste en el identificador único de una petición (Request Identifier). Su propósito es el de que las entidades oneM2M puedan identificar la petición inicial a la que se deben una o varias respuestas. Como el intercambio de información en oneM2M puede producirse de manera asíncrona, utilizando esta cabecera un emisor de operaciones puede identificar respuestas de peticiones que ha enviado hace cierto tiempo o entre las que ha realizado más operaciones diferentes. Consiste en una cadena de caracteres alfanuméricos determinado por el emisor que se reproduce en el resto de mensajes (retransmisiones y respuestas) asociados a dicha petición.

Content-Location: esta cabecera se incluye únicamente en la respuesta a la operación de un recurso en una CSE determinada. Su valor especifica la ruta en el

árbol de recursos, o identificador único (URI) mediante el cual se podrá acceder a partir de ese momento al recurso creado.

Además de estas cabeceras, la especificación de oneM2M define otras varias que pueden dotar de un significado más determinado a las peticiones y respuestas, pero su uso no ha sido necesario para la realización de este trabajo. A continuación se procederá al análisis de las peticiones y respuestas HTTP a realizar por el complemento de openHAB en desarrollo, analizando los puntos más relevantes de cada una de ellas.

4.2.2. Registro de la entidad de aplicación openHAB_AE en el nodo de infraestructura

Al desplegar el sistema oneM2M de las dos arquitecturas propuestas, varios procesos de registro se llevan a cabo. El primer registro que se produce es el de la capa de servicios comunes del nodo intermedio (MN-CSE) en la capa de servicios comunes del nodo de infraestructura. A partir de este momento se puede acceder a los recursos del nodo intermedio desde el nodo de infraestructura, ya que este queda registrado en su propio árbol de recursos. Posteriormente, al iniciar el complemento del nodo intermedio que se encarga de simular por software las dos bombillas incluidas en la arquitectura, el propio módulo registra en la MN-CSE las tres entidades de aplicación que representan las operaciones permitidas sobre las mismas, previamente mencionadas.

Tras el despliegue del sistema, el complemento desarrollado para openHAB deberá registrarse a si mismo en el nodo de infraestructura como una entidad de aplicación, para que la capa CSE que reside en él notifique al resto de nodos el registro del mismo, y que el nodo intermedio ajuste las políticas de acceso asociadas a las AEs de las bombillas para permitir su acceso por parte del complemento. Para realizar este registro, se realiza la petición HTTP descrita a continuación.

- **Petición:**

POST http://192.168.1.42:8080/~in-cse HTTP/1.1

Content-Type: application/xml; ty=2

X-M2M-Origin: C_OPENHAB_AE

X-M2M-RN: 0001

```
<m2m:ae xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="OPENHAB_AE" >
```

```
  <api>org.eclipse.smarthome.binding.onem2m</api>
```

```
<rr>false</rr>
```


</m2m:ae>

- **Respuesta:**

201 CREATED

Content-Length: 407

Content-Location: /in-cse/C_OPENHAB_AE

Content-Type: application/xml; charset=ISO-8859-1

Server: Jetty(8.1.16.v20140903)

X-M2M-Origin: /in-cse

X-M2M-RI: 0001

X-M2M-RSC:2001

<?xml version="1.0" encoding="UTF-8"?>

<m2m:ae xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="OPENHAB_AE">

<ty>2</ty>

<ri>/in-cse/C_OPENHAB_AE</ri>

<pi>/in-cse</pi>

<ct>20170918T210056</ct>

<lt>20170918T210056</lt>

<acpi>/in-cse/acp-249998520</acpi>

<et>20180918T210056</et>

<api>org.eclipse.smarthome.binding.onem2m</api>

<aei>C_OPENHAB_AE</aei>

<rr>>false</rr>

</m2m:ae>

La petición HTTP para realizar el registro de la entidad de aplicación del complemento de openHAB consiste en una petición de método POST, cuyo cuerpo incluye la información del recurso a registrar en formato XML. Además, según lo comentado anteriormente, se debe incluir dentro de la cabecera "Content-Type" el tipo de contenido M2M que se pretende crear, al ser una petición POST usada para crear un recurso. Podemos ver en la petición realizada que se incluye el parámetro ty=2. Este parámetro consiste en el tipo de recurso oneM2M (resourceType) se pretende crear, de entre todos los definidos por sus especificaciones. En este caso, el tipo de recurso 2 equivale al tipo AE (entidad de aplicación). La equivalencia entre cada tipo de recurso y el código a utilizar se puede encontrar en la tabla 6.3.4.2.1-1 del documento **TS-0004**. Además de esta cabecera, se ha incluido también la cabecera "X-M2M-Origin", que debe estar presente en cualquier petición o respuesta de oneM2M y que indica el origen del mensaje. Como la entidad originadora de la petición es la que se pretende registrar en el sistema, todavía no posee una identidad dentro de él. Para registrarse en el nodo de infraestructura deberá incluir en esta cabecera el identificador de entidad de aplicación (AE-ID) con el que quiere registrarse. La letra "C"

inicial no es casual, ya que el primer carácter del AE-ID está reservado según las especificaciones de oneM2M. Los identificadores que empiecen por esta letra serán únicos dentro de la CSE en la que se registra, por lo que en nodos externos podrán registrarse aplicaciones con el mismo identificador. La otra letra que se puede utilizar como inicio del AE-ID es la letra “S”, que indica que este identificador debe ser único en todo el sistema oneM2M. EL encargado de comprobar la disponibilidad del identificador en ambos casos será la entidad en la que se registra la AE.

En el cuerpo de la petición se han incluido los datos o parámetros de la aplicación a crear. Todos los campos de la respuestas están dentro de la etiqueta “m2m:ae”, ya que corresponden a una entidad de aplicación. El campo “rn” consiste en el nombre del recurso (resourceName), que estará formado por una cadena de caracteres. En este caso, se ha elegido el nombre “OPENHAB_AE”. EL campo “api” corresponde al identificador de la aplicación (App-ID), que en esta petición se ha rellenado como “org.eclipse.smarthome.binding.onem2m”, ya que será el nombre que se usará posteriormente para el desarrollo del complemento de openHAB. Por último, el parámetro “rr” representa el alcance de las peticiones (requestReachability) realizadas sobre el recurso creado. El valor “true” en este campo indica que las operaciones sobre este recurso pueden hacerse sobre cualquier entidad oneM2M que lo tenga a su alcance, aunque se deban hacer retransmisiones hacia otros nodos, mientras que el valor false indica que las peticiones sobre este recurso unicamente se pueden hacer directamente desde el emisor al nodo intermedio del sistema desplegado en este trabajo, que es en el que está registrado el recurso.

La respuesta incluye varias cabeceras importantes. La cabecera “X-M2M-Origin” incluye en este caso la entidad que origina la respuesta, es decir, /in-cse, que es el identificador único del nodo de infraestructura en el sistema. También incluye la cabecera “Content-Location”, que incluye el identificador único en el sistema de la entidad de aplicación que se ha creado. También se incluye la cabecera “X-M2M-RI” con el mismo valor que el que se ha definido en la petición, como se ha explicado anteriormente en este trabajo, para poder realizar un seguimiento de cada petición. Además de estas cabeceras, se incluyen otras comunes en HTTP, “Content-Length”, “Content-Type” y “Server”, que indican el tamaño de los datos incluidos en el cuerpo de la respuesta, su formato y el servidor web que ha elaborado la respuesta. Estas cabeceras se obviarán en las siguientes respuestas descritas ya que no son importantes para el estudio de oneM2M que se está realizando.

4.2.3. Acceso a los recursos registrados en el nodo intermedio

Una vez la nueva AE esté registrada en el nodo de infraestructura, podremos realizar peticiones originadas en esta sobre los miembros del árbol de recursos registrados en su capa de servicios comunes. Para ello, deberemos realizar una petición de tipo lectura (Retrieve) sobre el nodo de infraestructura, utilizando para ello una petición de método GET, seguida de la URL del nodo de infraestructura, que está compuesta por la dirección IP y el puerto del servidor web del nodo seguida de los

caracteres “/~”, como se ha especificado en el apartado 4.2.1 de este trabajo, y de el identificador único de su CSE, /in-cse. En la cabecera “X-M2M-Origin” se introduce el identificador único del emisor, que ha sido proporcionado por el nodo de infraestructura en la respuesta a la petición de registro mediante la cabecera “Content-Location”.

- **Petición:**

GET http://192.168.1.42:8080/~in-cse?rcn=5 HTTP/1.1

X-M2M-Origin: /in-cse/C_OPENHAB_AE

X-M2M-RN: 0002

- **Respuesta:**

200 OK

X-M2M-Origin: /in-cse

X-M2M-RI: 0002

X-M2M-RSC: 2000

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<m2m:cb xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="in-local">
```

```
  <ty>5</ty>
```

```
  <ri>/in-cse</ri>
```

```
  <ct>20170918T205736</ct>
```

```
  <lt>20170918T205736</lt>
```

```
  <acpi>/in-cse/acp-249998520</acpi>
```

```
  <cst>1</cst>
```

```
  <csi>in-cse</csi>
```

```
  <srt>1 2 3 4 5 9 14 15 16 17 23</srt>
```

```
  <poa>http://192.168.1.42:8080/</poa>
```

```
  <ch rn="acp_admin" ty="1">/in-cse/acp-249998520</ch>
```

```
  <ch rn="OPENHAB_AE" ty="2">/in-cse/C_OPENHAB_AE</ch>
```

```
  <ch rn="mn-cse1" ty="16">/in-cse/csr-48399777</ch>
```

```
</m2m:cb>
```

El cuerpo de la respuesta incluye la información sobre la entidad solicitada, /in-cse. En el la dirección del recurso sobre el que se hace la petición GET se ha incluido una cadena de consulta (rcn=5) que indica el contenido del resultado (resultContent) de la petición que se espera. Para las peticiones en este trabajo siempre se utilizará su valor 5, que indica al receptor que se desea que la respuesta incluya los atributos del recurso solicitado y las referencias a sus nodos hijos, es decir, los recursos que están registrados en él. En los parámetros del recurso solicitado podemos encontrar el tipo

de recurso, 5 en este caso, que corresponde al tipo CSEBase, que representa a la entidad de servicios comunes que reside en el nodo receptor. Los parámetros “ct” y “lt” indican el tiempo de creación y el de última modificación del recurso. El parámetro “cst” indica el tipo de CSE que es, en este caso el uno equivale al tipo CSEBase, y el parámetro “csi” corresponde al identificador de la CSE. El parámetro “srt” indica los tipos de recursos hijos que se pueden sobre este recurso (supportedResourceType). El parámetro “poa” indica la dirección y el puerto por la que la CSE puede recibir los mensajes (pointOfAccess). Por último, los recursos hijo del recurso solicitado aparecen rodeados por la etiqueta “ch”, seguidos de su identificador único en el dominio del nodo de infraestructura. En este caso, se observan tres recursos registrados, la propia entidad de aplicación OPENHAB_AE registrada anteriormente, el nodo mn-cse1 de tipo 16, correspondiente a una CSE que no reside en ese nodo (remoteCSE) y otro recurso de tipo 1 (accessControlPolicy), que define las políticas de acceso a los recursos registrados en el nodo de infraestructura, de las cuales se hablará más adelante en este trabajo.

Para acceder a las entidades de aplicación que están registradas en el nodo intermedio, será necesario primero obtener la información de la mn-cse1 que aparece como nodo hijo de la CSE del nodo de infraestructura en la anterior respuesta. Para ello, primero deberemos realizar una petición GET sobre este recurso con el identificador que nos ha proporcionado el nodo de infraestructura en la respuesta anterior (/in-cse/csr-48399777). La respuesta recibida tendrá un formato similar a la anterior, pero con los parámetros pertenecientes a mn-cse1. De estos parámetros, obtendremos el que determina su identificador de CSE único en el sistema, “csi”, que nos servirá para realizar la siguiente petición y obtener los recursos registrados en el nodo intermedio. Esta petición se detalla a continuación.

- **Petición:**

```
GET http://192.168.1.42:8080/~in-cse?rcn=5 HTTP/1.1
X-M2M-Origin: /in-cse/C_OPENHAB_AE
X-M2M-RI: 0003
```

- **Respuesta:**

```
200 OK
X-M2M-Origin: /in-cse
X-M2M-RI: 0003
X-M2M-RSC: 2000
```

```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:cb xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="mn-home">
  <ty>5</ty>
  <ri>/mn-cse1</ri>
  <ct>20170918T210011</ct>
```

```

<lt>20170918T210011</lt>
<acpi>/mn-cse1/acp-729403706</acpi>
<cst>1</cst>
<csi>mn-cse1</csi>
<srt>1 2 3 4 5 9 14 15 16 17 23</srt>
<poa>http://192.168.1.40:8383/</poa>
<ch rn="acp_admin" ty="1">/mn-cse1/acp-729403706</ch>
<ch rn="LAMP_0" ty="2">/mn-cse1/CAE152960455</ch>
<ch rn="LAMP_1" ty="2">/mn-cse1/CAE690784542</ch>
<ch rn="LAMP_ALL" ty="2">/mn-cse1/CAE47952112</ch>
<ch rn="in-local" ty="16">/mn-cse1/csr-912717516</ch>
</m2m:cb>

```

4.2.4. Ajuste de la política de acceso de las AEs que controlan las bombillas

- **Petición:**

```

GET http://192.168.1.42:8080/~mn-cse1/acp-729403706?rcn=5 HTTP/1.1
X-M2M-Origin: admin:cherran
X-M2M-RI: 0004

```

- **Respuesta:**

```

200 OK
X-M2M-Origin: /in-cse
X-M2M-RI: 0004
X-M2M-RSC: 2000

```

```

<?xml version="1.0" encoding="UTF-8"?>
<m2m:acp xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="acp_admin">
  <ty>1</ty>
  <ri>/mn-cse1/acp-729403706</ri>
  <pi>/mn-cse1</pi>
  <ct>20170918T210011</ct>
  <lt>20170918T210011</lt>
  <pv>

```

```
<acr>
  <acor>/mn-cse1 admin:cherran</acor>
  <acop>63</acop>
</acr>
<acr>
  <acor>* guest:guest</acor>
  <acop>34</acop>
</acr>
</pv>
<pvs>
  <acr>
    <acor>admin:cherran</acor>
    <acop>63</acop>
  </acr>
</pvs>
</m2m:acp>
```

- **Petición:**

PUT http://192.168.1.42:8080/~mn-cse1/acp-729403706?rcn=5 HTTP/1.1
X-M2M-Origin: admin:cherran
X-M2M-RI: 0005

- **Respuesta:**

200 OK
X-M2M-Origin: /in-cse
X-M2M-RI: 0005
X-M2M-RSC: 2004

```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:acp xmlns:m2m="http://www.onem2m.org/xml/protocols">
  <lt>20170918T212837</lt>
  <pv>
    <acr>
      <acor>/mn-cse1 admin:cherran /in-cse/C_OPENHAB_AE</acor>
      <acop>63</acop>
    </acr>
  </pv>
  <acr>
```

```

        <acor>* guest:guest</acor>
        <acop>34</acop>
    </acr>
</pv>
<pvs>
    <acr>
        <acor>admin:cherran</acor>
        <acop>63</acop>
    </acr>
</pvs>
</m2m:acp>

```

4.2.5. Descubrimiento de recursos

- **Petición:**

```

GET http://192.168.1.42:8080/~mn-cse1/CAE152960455?rcn=5 HTTP/1.1
X-M2M-Origin: /in-cse/C_OPENHAB_AE
X-M2M-RI: 0006

```

- **Respuesta:**

```

200 OK
X-M2M-Origin: /in-cse
X-M2M-RI: 0006
X-M2M-RSC: 2000

```

```

<?xml version="1.0" encoding="UTF-8"?>
<m2m:ae xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="LAMP_0">
    <ty>2</ty>
    <ri>/mn-cse1/CAE152960455</ri>
    <pi>/mn-cse1</pi>
    <ct>20170918T210028</ct>
    <lt>20170918T210028</lt>
    <acpi>/mn-cse1/acp-729403706</acpi>
    <et>20180918T210028</et>
    <api>LAMP_0</api>
    <aei>CAE152960455</aei>

```

```
<poa>sample</poa>
<ch m="DATA" ty="3">/mn-cse1/cnt-638532714</ch>
<ch m="DESCRIPTOR" ty="3">/mn-cse1/cnt-911603507</ch>
<rr>true</rr>
</m2m:ae>
```

- **Petición:**

GET http://192.168.1.42:8080/~mn-cse1/mn-home/LAMP_0/DATA/la?rcn=5 HTTP/1.1
X-M2M-Origin: /in-cse/C_OPENHAB_AE
X-M2M-RI: 0007

- **Respuesta:**

200 OK
X-M2M-Origin: /in-cse
X-M2M-RI: 0007
X-M2M-RSC: 2000

```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:cin xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="cin_555908894">
  <ty>4</ty>
  <ri>/mn-cse1/cin-555908894</ri>
  <pi>/mn-cse1/cnt-638532714</pi>
  <ct>20170918T210028</ct>
  <lt>20170918T210028</lt>
  <st>0</st>
  <cnf>application/obix</cnf>
  <cs>216</cs>
  <con>
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <obj>
      <str val="LAMP" name="type"/>
      <str val="Home" name="location"/>
      <str val="LAMP_0" name="lampId"/>
      <bool val="false" name="state"/>
    </obj>
  </con>
</m2m:cin>
```


- **Petición:**

GET http://192.168.1.42:8080/~mn-cse1/mn-home/LAMP_0/DATA/la?rcn=5 HTTP/1.1
X-M2M-Origin: /in-cse/C_OPENHAB_AE
X-M2M-RI: 0008

- **Respuesta:**

200 OK
X-M2M-Origin: /in-cse
X-M2M-RI: 0008
X-M2M-RSC: 2000

```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:uril xmlns:m2m="http://www.onem2m.org/xml/protocols">
    /mn-cse1/mn-home/LAMP_0/DATA/cin_555908894
    /mn-cse1/mn-home/LAMP_0/DESCRIPTOR/cin_270357510
    /mn-cse1/mn-home/LAMP_1/DATA/cin_1901140
    /mn-cse1/mn-home/LAMP_1/DATA/cin_586427163
    /mn-cse1/mn-home/LAMP_1/DATA/cin_587806251
    /mn-cse1/mn-home/LAMP_1/DESCRIPTOR/cin_311088019
    /mn-cse1/mn-home/LAMP_ALL/DESCRIPTOR/cin_635499187
</m2m:uril>
```

4.2.6. Control de las bombillas

- **Petición:**

GET http://192.168.1.42:8080/~mn-cse1/mn-home/LAMP_0/DESCRIPTOR/la?rcn=5 HTTP/1.1
X-M2M-Origin: /in-cse/C_OPENHAB_AE
X-M2M-RI: 0009

- **Respuesta:**

200 OK

X-M2M-Origin: /in-cse

X-M2M-RI: 0009

X-M2M-RSC: 2000

```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:cin xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="cin_270357510">
  <ty>4</ty>
  <ri>/mn-cse1/cin-270357510</ri>
  <pi>/mn-cse1/cnt-911603507</pi>
  <ct>20170918T210028</ct>
  <lt>20170918T210028</lt>
  <st>0</st>
  <cnf>application/obix</cnf>
  <cs>663</cs>
  <con>
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <obj>
      <str val="LAMP" name="type"/>
      <str val="Home" name="location"/>
      <str val="LAMP_0" name="appld"/>
      <op href="/mn-cse1/mn-home/LAMP_0/DATA/la" name="getState"
        is="retrieve"/>
      <op href="/mn-cse1/mn-home/LAMP_0?op=getStateDirect&lampid=LAMP_0"
        name="getState(Direct)" is="execute"/>
      <op href="/mn-cse1/mn-home/LAMP_0?op=setOn&lampid=LAMP_0"
        name="switchON" is="execute"/>
      <op href="/mn-cse1/mn-home/LAMP_0?op=setOff&lampid=LAMP_0"
        name="switchOFF" is="execute"/>
      <op href="/mn-cse1/mn-home/LAMP_0?op=toggle&lampid=LAMP_0"
        name="toggle" is="execute"/>
    </obj>
  </con>
</m2m:cin>
```

- **Petición:**

POST http://192.168.1.42:8080/~mn-cse1/mn-home/LAMP_0?op=setOn&lampid=LAMP_0
HTTP/1.1

X-M2M-Origin: /in-cse/C_OPENHAB_AE

X-M2M-RI: 0009

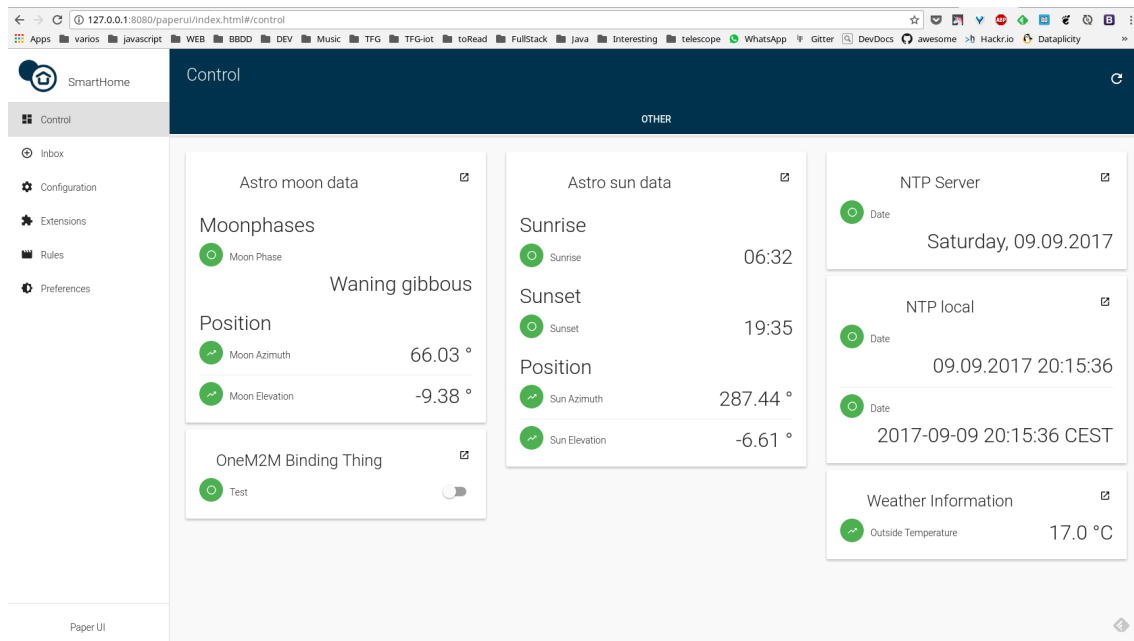
- ```
200 OK
X-M2M-Origin: /in-cse
X-M2M-RI: 0010
X-M2M-RSC: 2000
```

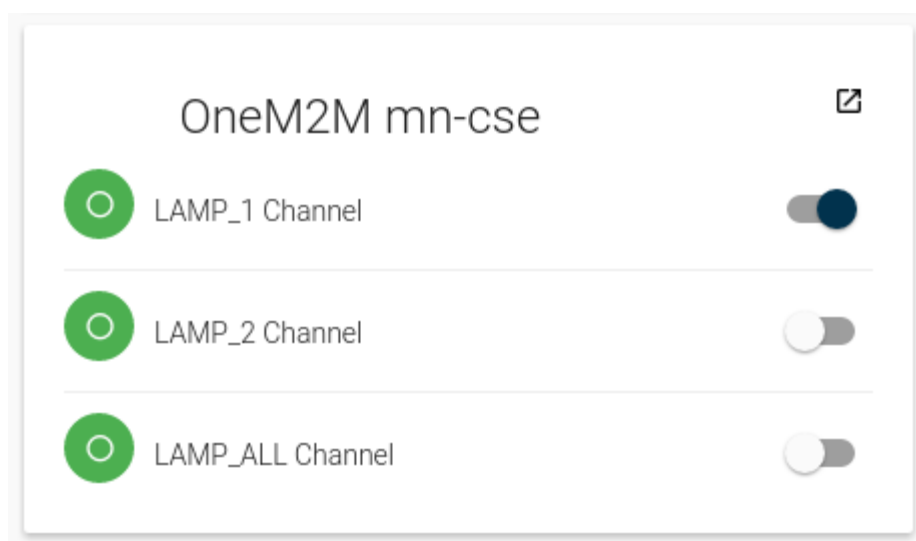
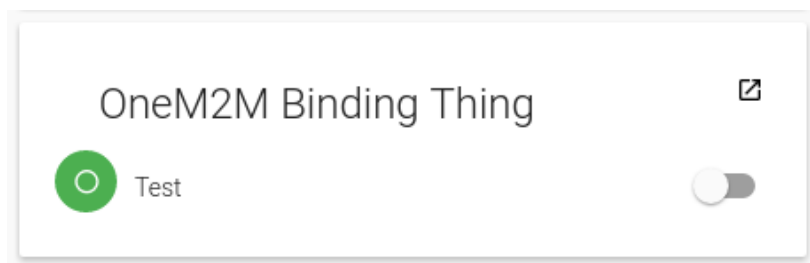
X-M2M-RSC: 2000

[illegible]

# 5. Desarrollo

## 5.1. Primera aproximación





### 5.1.1. Pruebas

| Prueba | Resultado esperado | Resultado obtenido |
|--------|--------------------|--------------------|
|        |                    |                    |
|        |                    |                    |
|        |                    |                    |

## 5.2. Segunda aproximación

### 5.2.1. Pruebas

[illegible]

## **6. Conclusiones y Recomendaciones**

*Harán referencia a los resultados de la investigación planteada en relación a los objetivos. Es conveniente que aparezcan numeradas.*

## **Bibliografía**