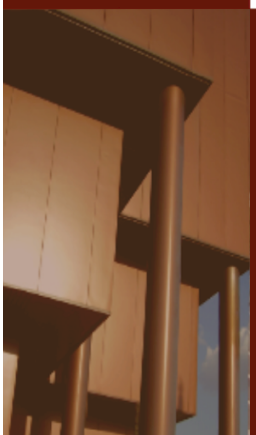
Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Sistema integrado basado en FiWare para la obtención y difusión de información de consumo y calidad de agua

Autor: Miguel Collado Quesada

Tutor: Fernando Guerrero López
e Ignacio Eguía Salinas



Dpto. **de Organización Industrial y Gestión de Empresa I**
Escuela Técnica Superior de Ingeniería

Sevilla, 2018

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Sistema integrado basado en FiWare para la obtención y difusión de información de consumo y calidad de agua

Autor:
Miguel Collado Quesada

Tutor:
Fernando Guerrero López e
Ignacio Eguía Salinas
Profesor titular

Dpto. de Organización Industrial y Gestión de Empresa I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Proyecto Fin de Carrera: Sistema integrado basado en FiWare para la obtención y difusión de información de consumo y calidad de agua

Autor: Miguel Collado Quesada

Tutor: Fernando Guerrero López e
Ignacio Eguía Salinas

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

A mi familia

A mis amigos

Agradecimientos

Debo de agradecer el hecho de haber llegado hasta aquí a mi familia. Especialmente, me gustaría agradecer a mis padres por todo haberlo hecho posible. A mi padre por imprimir en mi el tesón y el valor del esfuerzo. A mi madre por enseñarme que la perserverancia es la clave del éxito. A mis hermanos por estar en los buenos momentos y en los no tan buenos.

A mis amigos y compañeros de trabajo por ayudarme en el proyecto, aportándome ideas y mejoras que sirvieron de utilidad.

A mis tutores por todo lo aprendido gracias a hacer posible realizar este proyecto y contribuir en él.

Miguel Collado Quesada

Sevilla, 2018

Resumen

La tecnología está avanzando a pasos agigantados durante las últimas décadas. Cada vez son más diversos los protocolos y aplicaciones usadas para casos de uso similares. Es por eso, que este trabajo se centra en el uso de FiWare como plataforma del internet de las cosas que permita unificar información proveniente de dispositivos capaces de extraer información acerca del consumo y calidad del agua y su uso en aplicaciones de usuario que puedan crear valor a dichos datos

Para entender la perspectiva del internet de las cosas, se describirá el concepto del mismo, además de las tecnologías y protocolos que se vienen usando en este panorama desde su inicio hasta la actualidad, centrándonos en los más importantes. Además, se especificarán los distintos ecosistemas que pueden ser usados en la actualidad para la integración de dispositivos del internet de las cosas.

Más adelante, el trabajo se centra en la plataforma usada en el trascurso del proyecto: FiWare. Se describirán sus características, así como los componentes de la misma y estándares que se usan.

Centrándonos en el caso de uso, se expondrá el desarrollo de los diferentes sensores y dispositivos desplegados en el proyecto, así como los componentes que se integran, protocolos que se usan y estándares de comunicaciones en los que se apoyan. Se definirá la arquitectura de los dispositivos productores de información, los sensores, los cuales serán capaces de obtener informacion sobre parámetros de calidad de agua como conductividad, oxigeno disuelto, temperatura y ph, parámetros de consumo de agua y presión. Se detallará como los sensores usan los coordinadores de información como pasalera al ecosistema implantado.

Posteriormente, se detallará el despliegue de los componentes de FiWare usados para esta solución, así como su interrelación de unos a otros y las características específicas las cuales les dan utilidad en el marco del proyecto.

Por último, se describirá el uso que se hace a los datos extraídos en aplicaciones finales.

Abstract

Technology is advancing by leaps and bounds during the last decades. The protocols and applications used for similar use cases are increasingly diverse. That is why this work focuses on the use of FiWare as an internet platform for things that allows unifying information from devices capable of extracting information about the consumption and quality of water and its use in user applications that can create value to said data

To understand the perspective of the internet of things, the concept of it will be described, as well as the technologies and protocols that have been used in this panorama from its beginning to the present, focusing on the most important ones. In addition, it will specify the different ecosystems that can be used today for the integration of devices of the internet of things.

Later, the work focuses on the platform used during the project: FiWare. Their characteristics will be described as well as the components of the same and standards that are used.

Focusing on the use case, the development of the different sensors and devices deployed in the project will be exposed, as well as the components that are integrated, protocols that are used and communication standards on which they are supported. The architecture of the information-producing devices, the sensors in this case, will be defined, which will be able to obtain information on water quality parameters such as conductivity, dissolved oxygen, temperature and pH, water consumption parameters and pressure. It will be detailed how the sensors use the information coordinators as a gateway to the implanted ecosystem.

Subsequently, the deployment of the FiWare components used for this solution will be detailed, as well as their interrelation to each other and the specific characteristics which give them utility in the framework of the project.

Finally, the use made of the data extracted in final applications will be described.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 OBJETIVOS

Cuando es evidente que los objetivos no pueden ser
logrados, no ajustes los objetivos, ajusta los pasos a seguir.

- Confucio -

El principal objetivo de este Trabajo fin de Grado es el desarrollo y despliegue de una red de dispositivos productores de información y el uso final de esta información en aplicaciones que aporten valor al usuario. Todo esto se apoyará en la plataforma FiWare junto con sus componentes para conseguir una gestión que permita la flexibilidad tanto en los productores como consumidores de información, para permitir que estos puedan implementar protocolos heterogéneos y de distinta naturaleza.

En nuestro caso, se integrarán los productores de información referentes a calidad y consumo de agua, para que puedan ser usados tanto para distribuidoras de agua y clientes finales, como para organismos públicos y otras entidades.

Se estudiarán los diferentes componentes que forman la plataforma FIWARE para su aplicación en este determinado caso de uso.

Al desplegar los componentes de la plataforma en el marco de la gestión de datos de agua, se buscará una gestión más eficiente del uso y abastecimiento del agua, así como información en tiempo real.

La integración en tiempo real del abastecimiento de agua gracias a FIWARE, facilitará la detección de fugas de manera más precisa y rápida, facilitando la actuación sobre estas y su prevención.

2 INTRODUCCIÓN

Estamos demasiado acostumbrados a atribuir a una sola causa lo que es el producto de varias, y la mayoría de nuestras controversias proceden de eso.

- Marco Aurelio -

En la actualidad, nos encontramos en un mundo globalizado donde abundan dispositivos de todo tipo y para diversos usos. Es por eso que surge la necesidad del uso de ecosistemas que permitan la integración de infinidad de datos heterogéneos en un sistema global, único y escalable.

Gracias al avance de la tecnología y a la capacidad de recolección y uso de datos para todo tipo de negocios, se acuña el nombre de "Internet de las Cosas" como plataforma donde todos los objetos la vida cotidiana pueden interactuar y comunicarse a través de la red.

A lo largo de este capitulo, se expondrá una definición del denomidado Internet de las Cosas, además de describir ciertos aspectos importantes de su historia hasta la actualidad y pasando por las diferentes tecnologías que lo hacen posible.

2.1 Introducción al Internet de las Cosas

De acuerdo al IoT European Research Cluster [1] , Internet de las Cosas (IdC, en inglés IoT) es: "una infraestructura de red dinámica y global con capacidades de auto-configuración basada en protocolos de comunicación estándar y que operan entre sí donde las "cosas" físicas y virtuales tienen identidad, atributos físicos y personalidades virtuales utilizando interfaces inteligentes, siendo integradas a la perfección en la red de información".

Otra definición extendida es: "la IoT es una infraestructura global para la sociedad de la información, habilitando servicios avanzados al interconectar (física y virtualmente) cosas basadas en el mundo real con modernas tecnologías de la información y comunicaciones. A través de la identificación, captura de datos y capacidades de comunicación y procesado, la IoT permite un uso pleno de los datos para ofrecer servicios a aplicaciones, a la misma vez de dotar de los requisitos de seguridad y privacidad pertinentes. Desde una perspectiva más amplia, la IoT puede ser percibida como una nueva visión con implicaciones tanto tecnológicas como sociales".

La IoT se basa en tres pilares básicos de conocimiento (Figura 2-1): el diseño de las cosas (hardware), el diseño de la infraestructura de comunicación (telemática) y la toma de decisiones (semántica y de razonamiento). Así, el fenómeno de la IoT se encontraría en la zona de intersección de estas tres áreas. La tercera definición proviene de una infografía [2], que resume de una manera rápida y eficaz los conceptos básicos de esta tecnología: "los Sistemas Inteligentes (Smart Systems) y la IoT son impulsados por una combinación de (1) sensores y actuadores, (2) conectividad y (3) personas y procesos". De acuerdo con esta visión, los sensores y actuadores crean un sistema nervioso digital y, gracias a las conexiones, se habilita la disponibilidad de los datos que son transmitidos, almacenados, combinados y analizados entre sistemas que integran datos, personas, procesos y otros sistemas. Aunque menos formal, esta última es una definición simple y clara que aborda los tres principios fundamentales de la IoT: las cosas, las conexiones y los procesos.

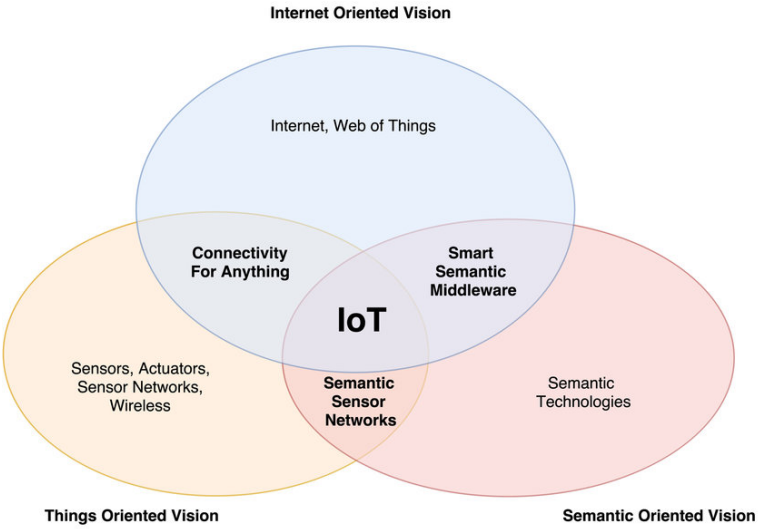


Figura 2-1. Pilares IoT de conocimiento.

2.2 Historia del Internet de las Cosas

Para ponernos en contexto, tendríamos que remontarnos a 1926 para hablar del gran Nikola Tesla cuyas patentes y trabajos teóricos conformaron la base de las comunicaciones inalámbricas y de radio.

El origen del Internet de las Cosas data del año 1969 cuando la red ARPANET [3] estableció comunicación entre las universidades de Stanford y Ucla, permitiendo no sólo el desarrollo de la IoT, sino de Internet propiamente dicho, la red de redes que actualmente conocemos y utilizamos diariamente. Diez años después se probó el TCP/IP, los protocolos de red en los que se basa Internet y que permiten la transmisión de datos entre computadoras. En 1990 Berners-Lee implementó la primera comunicación exitosa entre un cliente Hypertext Transfer Protocol (HTTP) y un servidor a través de Internet, había inventado la World Wide Web. Él mismo, un año más tarde, creó la primera página web. A partir de ese momento el desarrollo tecnológico es vertiginoso, comienza la revolución de Internet.

En el año 1990 Jhon Romkey y Simon Hacket [4] desarrollaron el primer objeto con conexión a Internet, la primera "cosa" con capacidad de manejo remoto. Esa "cosa" fue una tostadora inteligente que podía controlarse remotamente a través de cualquier ordenador, pudiendo encenderla, apagarla o incluso controlar el tiempo de tostado.

La primera aparición del IoT como hoy se conoce procede del Instituto Tecnológico de Massachusetts (MIT), en concreto del trabajo del Auto-ID Center. Este grupo, fundado en 1999, realizaba investigaciones en el campo de la identificación por radiofrecuencia en red (RFID) y las tecnologías de sensores emergentes. Fue ahí donde se empezó a formar el actual concepto que tenemos de la IoT.

En 2005 la agencia de las Naciones Unidas International Telecommunications Union ITU publica el primer estudio sobre el tema. A partir de ese momento Internet de las Cosas adquiere otro nivel.

"*Una nueva dimensión se ha agregado al mundo de las tecnologías de información y la comunicación (TIC): a cualquier hora, en cualquier lugar, ahora vamos a tener conectividad para cualquier cosa. Las conexiones se multiplican y crearán una nueva red dinámica de redes con redes, una Internet de las Cosas*".

En 2005 también comienza la aventura de Arduino, una plataforma de software y hardware abierto para el desarrollo de dispositivos digitales que puedan controlar objetos del mundo real. Esta plataforma cuenta en la actualidad con una comunidad bastante extensa en todo el mundo. Todos los productos que vende la compañía de la plataforma, son distribuidos como Hardware y Software libre bajo licencias GPL (Licencia Publica General de GNU).

Atendiendo a la Figura 2.2, se puede observar el crecimiento que ha seguido la cantidad de dispositivos conectados a Internet. En el año 2003, había aproximadamente 6,3 mil millones de personas en el mundo, con un número aproximado de 500 millones de dispositivos conectados a Internet. La relación entre estas variables era de 0,08 dispositivos por persona.

El auge en el desarrollo y la mejora de las tecnologías portátiles e inalámbricas como smartphones o tablets

elevó a 12,5 mil millones en 2010 la cantidad de dispositivos conectados a la red de redes, pasando la relación dispositivos/personas de 0,08 a 1,84, es decir, a partir de este año el número de dispositivos supera al de personas. Las predicciones apuntan a que en el año 2020 cerca de 50.000 millones de dispositivos electrónicos estarán conectados a Internet, cambiando radicalmente nuestro estilo de vida.

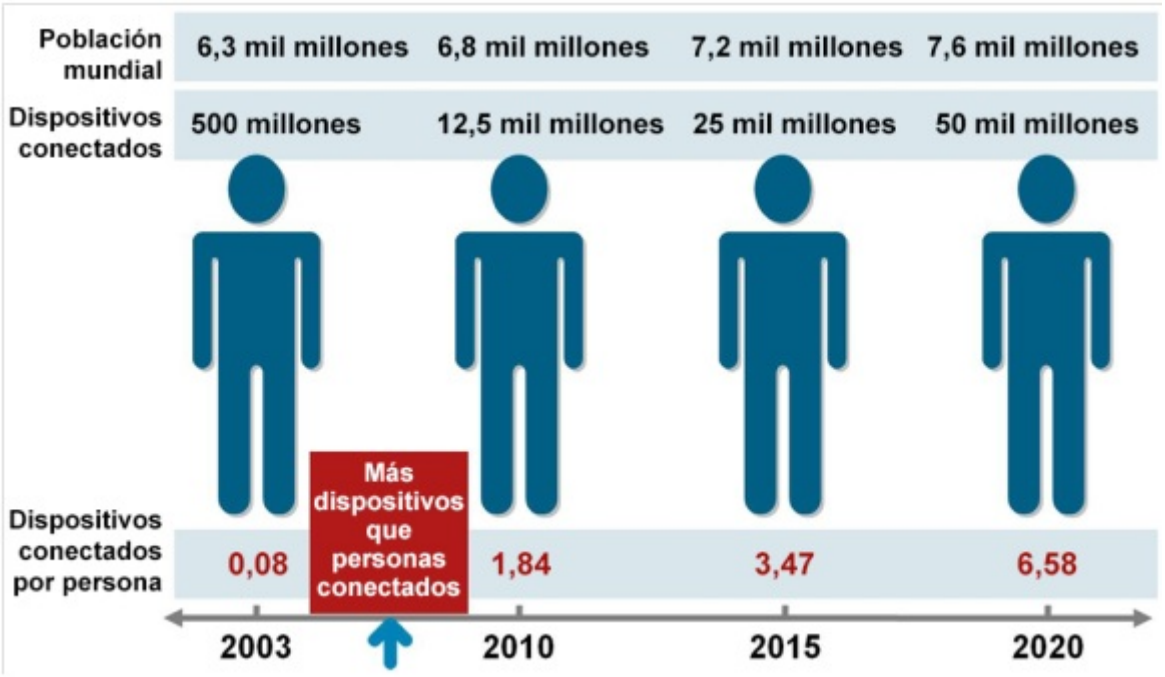


Figura 2-2. Crecimiento de numero de dispositivos.

2.3 Tecnologías Usadas en IoT

Los administradores del área de TI de las empresas y encargados de las redes y el funcionamiento del Internet de las Cosas tienen a su disposición muchas opciones distintas para conectar dispositivos de este tipo. Cada opción tiene distintas ventajas, aunque también desventajas. Aquí hablamos de algunas de ellas.

2.3.1 802.15.4

El estándar 802.15.4 define el nivel físico y el control de acceso al medio de redes inalámbricas de área personal con tasas bajas de transmisión de datos. El grupo de trabajo IEEE 802.15 es el responsable de su desarrollo. Es la tecnología en la que se apoyan protocolos como Bluetooth y arquitecturas basadas en ZigBee, un protocolo ampliamente usado en ecosistemas IoT ya que dichos ecosistemas no pueden manejar protocolos de red muy pesados debido a que esto afectaría seriamente el consumo de energía y requerirían de mayor poder de procesamiento [5].

Las características más importantes en este estándar son su flexibilidad de red, bajos costos, bajo consumo de energía.

La motivación para el uso de tecnología inalámbrica es la reducción en los gastos de instalación, ya que nunca es necesario cambiar el cableado. Las redes inalámbricas implican un gran intercambio de información con un mínimo de esfuerzo de instalación. Esta tendencia es impulsada por la gran capacidad de integrar componentes inalámbricos de una forma más barata y el éxito que tienen otros sistemas de comunicación inalámbrica como los celulares.

2.3.2 Bluetooth

Aunque muchos piensan que esta tecnología de red ya está muerta, el Bluetooth permite comunicaciones inalámbricas integradas para dispositivos de IoT.

Como puntos a favor de la tecnología Bluetooth, encontramos su reducido consumo de energía y su seguridad.

En contra, podemos decir que esta tecnología se ve limitada en temas de alcance y capacidad de la red, pasando a estar en determinados casos de uso donde intervengan comunicaciones de larga distancia [6].

La nueva versión de Bluetooth, la 5.0, viene a introducir varias mejoras [7]:

- Doble de capacidad con respecto a la versión Bluetooth 4.0: Antiguamente el pico de velocidad estaba establecido en 1Mbps (Megabit por segundo). Con esta versión, existe un modo con una capacidad de 2Mbps. Esta mejora, ofrece unos beneficios adicionales:
    - Reduce la potencia de transmisión ya que la misma cantidad de datos es transmitida en menos tiempo.
    - Mejora en la coexistencia de distintas comunicaciones inalámbricas gracias a la reducción del tiempo de emisión.
- Aumenta cuatro veces el rango de distancia: Esta versión añade un modo de alto alcance donde se utilizan técnicas de error de corrección de tramas llamada Corrección de Tramas hacia Delante (FEC por sus siglas en ingles). Gracias a este, en vez de requerir que el emisor reenvíe una trama en el caso de que esta contenga errores, el receptor puede reconstruir dicha trama a través de la redundancia que existe en los datos de la trama.
  Rangos de hasta ochocientos metros han sido testados con resultados satisfactorios durante los tests de el modo de largo alcance que Bluetooth 5.0 incorpora.

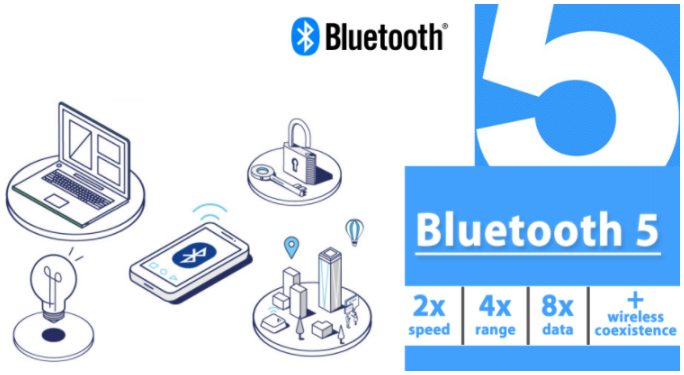


Figura 2-3. Bluetooth 5.0.

Como podemos ver, las nuevas implementaciones que Bluetooth 5.0 trae pueden ser beneficiosas para un amplio rango de aplicaciones IoT. Es por esto que no podemos hablar del internet de las cosas sin hablar de Bluetooth como un referente en la implementación del estándar 802.15.4 en estos casos de uso determinados.

2.3.3 WiFi

WiFi se basa en el estándar del IEEE 802.11 el cual define el uso de los dos niveles inferiores de la arquitectura o modelo OSI (capa física y capa de enlace de datos), especificando las normas de funcionamiento de una red de área local inalámbrica (WLAN). Este es un protocolo inalámbrico gestionado por la WiFi Alliance [8] diseñado para intentar reemplazar a los protocolos de red de área local cableados usando comunicaciones inalámbricas sobre bandas de frecuencia de uso libre ISM (bandas de uso para industrial, científico y medico) [9]. La idea de este protocolo era la de crear un protocolo fácil de implementar, fácil de usar en conexiones inalámbricas de corto alcance con interoperabilidad entre proveedores. Además, se pensó sin la implementación de una eficiencia en el uso de la frecuencia y obviando aspectos relativos a la eficiencia energética.

Las ventajas de usar este protocolo en una red IoT son el bajo costo de la infraestructura y dispositivos, la facilidad en el despliegue y eficiencia del espectro. En contra, el elevado consumo de potencia y la congestión del canal inalámbrico son grandes desventajas para el uso de dicho protocolo en una infraestructura del internet de las cosas.

Figura 2-4. Protocolo WiFi.

Las nuevas versiones del estándar 802.11 vienen a implementar desarrollos para una alta eficiencia en el consumo y uso del espectro electromagnético. En concreto, el nuevo estándar 802.11ax incluye la tecnología Target Wake Time (TWT). Esta permite mejorar la duración de la batería de smartphones y otros dispositivos móviles. Lo consigue optimizando el consumo de la batería cuando los dispositivos están inactivo.

Aunque el nuevo estándar Wi-Fi 802.11 ax parece ser la solución a nuestros problemas de conexión inalámbrica, no llegará en breve. La Wi-Fi Alliance estima que la adopción en masa de este nuevo estándar no se producirá, al menos, hasta 2019 [10].

Figura 2-5. 802.11ax

WiFi es el gran amigo y la herramienta más confiable de toda una generación de dispositivos fijos y móviles, pero exige mucha energía para una conectividad continua, algo que piden los dispositivos IoT, y aunque muchas casas inteligentes funcionan a la perfección con el WiFi, no todos los sistemas podrían sostenerse por completo de este modo.

2.3.4 4G LTE

El estándar Long Term Evolution o LTE sirve para medir la comunicación inalámbrica a alta velocidad. Esta tecnología es desarrollada por el 3GPP [11]. Las redes LTE ofrecen velocidades más bajas y más al alcance de los proveedores de servicio, a diferencia de la red 4G normal, que es más cara. Sin embargo, no todos los dispositivos son compatibles con esta red, tienen un gran consumo de energía y para la gran cantidad de datos manejados, puede resultar muy costoso.

Figura 2-6. LTE Cat M. [12]

LTE-M, también conocida por CAT-M1, viene del estándar LTE implementado en dispositivos IoT. Utiliza las antenas LTE instaladas y está optimizada para un ancho de banda más eficientes que permitirá prolongar la vida de las baterías.

Cuando hablamos de LTE-M nos podemos fijar en ciertos aspectos que lo diferencian de los demás:

- Velocidad de subida y bajada de hasta 1 Mbps.
- VoLTE: Tecnología que permite el envío de voz mediante codecs de transmisión sobre tecnologías de red IP.
- Movilidad: LTE-M está preparada para dispositivos en movimiento.
- Soportado por la mayoría de los equipamientos móviles.
- Las redes LTE-M pueden coexistir con redes 2G, 3G y 4G y beneficiarse de todas las características de privacidad y seguridad que estas ofrecen como autenticación de entidades, integridad de información, etc.

2.3.5 Ethernet

Ethernet es la tecnología de red de área local (LAN) más ampliamente instalada [13]. Ethernet es un protocolo de capa de enlace en la pila de TCP/IP, que describe cómo los dispositivos en red pueden formatear los datos para su transmisión a otros dispositivos de red en el mismo segmento de red y cómo colocarlos en la conexión de red. Toca tanto la Capa 1 (la capa física) como la Capa 2 (la capa de enlace de datos) en el modelo de protocolo de red OSI. Ethernet define dos unidades de transmisión, paquete y marco. El marco incluye no solo la "carga útil" de datos que se transmiten, sino también información de direcciones que identifica las direcciones físicas de "Control de acceso a medios" (MAC) del emisor y el receptor, el etiquetado de VLAN e información de calidad de servicio, así como información de corrección de errores para detectar problemas en la transmisión. Cada cuadro está envuelto en un paquete, que fija varios bytes de información utilizada para establecer la conexión y marcar donde comienza el cuadro.

Especificado en la familia de estándares conocida como IEEE 802.3, Ethernet fue desarrollado originalmente por Xerox en la década de 1970. Ethernet fue inicialmente diseñado para funcionar con cables coaxiales, pero una LAN Ethernet típica ahora usa grados especiales de cables de par trenzado o cableado de fibra óptica. Los estándares de Wi-Fi (IEEE 802.11a, b, g, n y ahora ac) definen el equivalente de Ethernet para LAN inalámbricas). Los estándares de Ethernet están evolucionando constantemente para abarcar nuevos medios, mayores velocidades de transmisión y cambios en el contenido del marco (por ejemplo, 802.3 ac para acomodar la VLAN y el etiquetado de prioridad) y los requisitos funcionales (por ejemplo, 802.3af, definiendo el *Power Over Ethernet* [POE] crucial para la mayoría de las implementaciones de telefonía IP y Wi-Fi) [14].

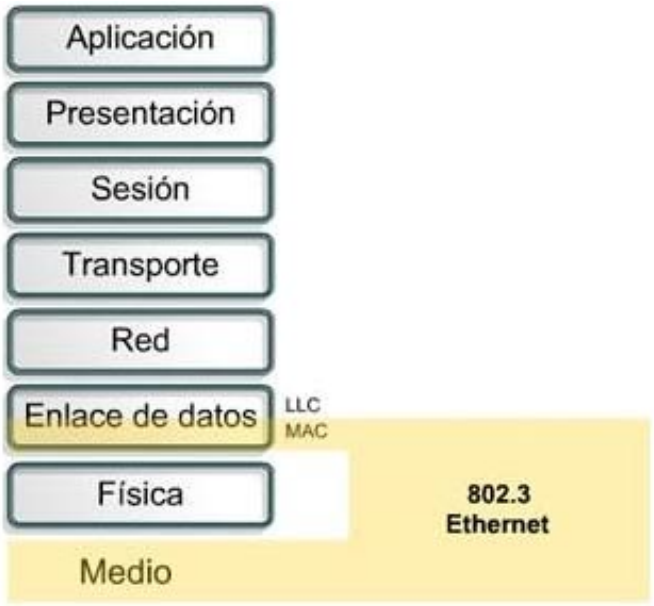


Figura 2-7. 802.3 en modelo OSI.

Si bien el Ethernet permite conexiones LAN de alta velocidad, tiene un gran defecto que lo hace poco beneficioso para IoT: requiere de un cable para conectarse a sus dispositivos, y eso puede convertirse en un despropósito cuando lo que se quiere es más libertad inalámbrica y de largo alcance.

2.3.6 Narrowband-IoT

Actualmente, diversas compañías están acelerando la implementación de Narrowband-IOT (NB-IOT). Esta tecnología es la primera centrada en conectar a internet objetos cotidianos que requieren pequeñas cantidades de datos en períodos de tiempo largos. Es una de las distintas tecnologías competidoras -Sig-Fox, LoraWan…- que se denominan, generalmente, low-power wide-area networking (LPWAN), redes de amplia área de baja potencia.

Esta tecnología ha sido desarrollada para permitir comunicaciones eficientes y una alta durabilidad de la batería, para dispositivos distribuidos masivamente [15]. Utiliza la ya existente red móvil para conectar todos esos objetos. NB-IOT está diseñado para ampliar el futuro de la conectividad IOT de una manera más segura y fiable. Es ideal para dispositivos que generan un tráfico de datos no muy alto y tienen un ciclo de vida largo [16].

Hay diversas características que pueden hacer que esta tecnología lidere el mercado de las comunicaciones IOT en un futuro a corto plazo, pero vamos a destacar cuatro de las más importantes:

- Bajo coste: Los módulos NB-IOT tiene menores costes que los de otras tecnologías de comunicación (como 3G, 4G, GPRS…) y también de LTE-M.
- Más celulares por antena: Los dispositivos NB-IOT utilizan un ancho de banda de 180KHz y se estima que en cada red pueden caber 100.000 conexiones.
- Excelente penetración en interiores y bajo tierra.
- Seguridad extremo a extremo: Seguridad y privacidad de información gracias a varios niveles de protección junto con direcciones IP privadas.

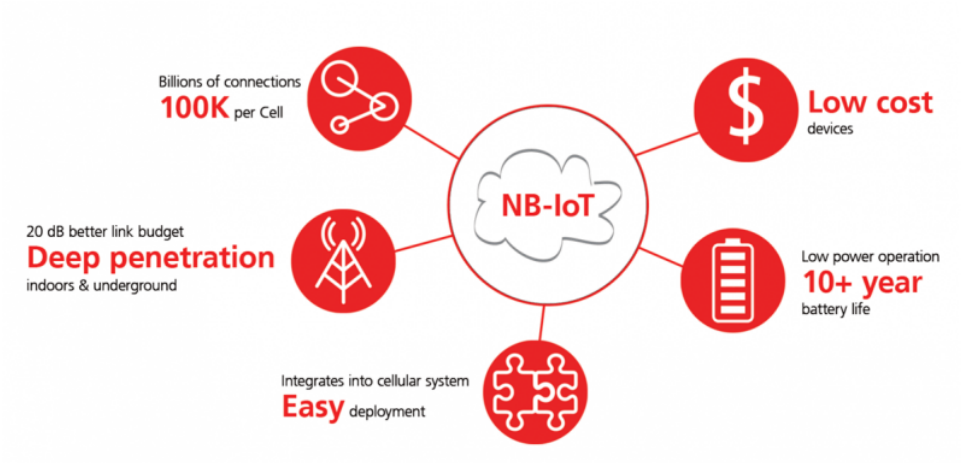


Figura 2-8. NB-IOT [17].

Estas son solo algunas de las tecnologías de conectividad disponibles para los dispositivos de IoT, aunque, de hecho, la industria está creando nuevas tecnologías de red que sirven específicamente para conectar este tipo de dispositivos, por lo que la conectividad personalizada puede, a la larga, funcionar mejor.

2.4 Protocolos Usados en IoT

Existe una amplia variedad de protocolos utilizados en el ámbito de Internet de las Cosas. Así, cada plataforma de IoT implementa una serie de protocolos u otros en función de los servicios que ofrece a los usuarios. Los protocolos más destacados son los siguientes:

2.4.1 Protocolos de Nivel de Red

2.4.1.1 IpV4

IPv4 es uno de los principales protocolos en el conjunto de protocolos TCP/IP. Es la primera versión usada para producir ARPANET. Este funciona en la capa de red del modelo OSI y en la capa de Internet del modelo TCP/IP. Por lo tanto, este protocolo tiene la responsabilidad de identificar hosts basados en sus direcciones lógicas y para dirigir los datos entre ellos a través de la red subyacente.

IP proporciona un mecanismo para identificar de forma única los hosts por un esquema de direccionamiento IP. Este no garantiza que los paquetes se entregarán al host destino, pero implementa funciones para maximizar la posibilidad para llegar al destino. IPv4 utiliza 32 bits dirección lógica las cuales están escaseando [18].

IPv4 usa direcciones de 32 bits, limitándola a 232 = 4.294.967.296 direcciones únicas, muchas de las cuales están dedicadas a redes locales (LANs). Por el crecimiento enorme que ha tenido Internet (mucho más de lo que esperaba, cuando se diseñó IPv4), combinado con el hecho de que hay desperdicio de direcciones en muchos casos, ya hace varios años se vio que escaseaban las direcciones IPv4.

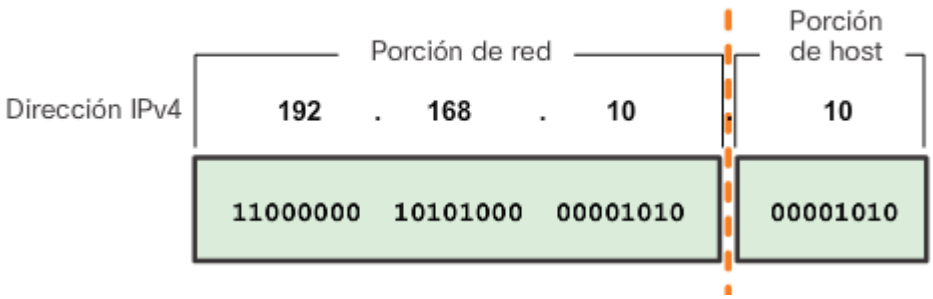


Figura 2-9. Cabecera IPv4

Las direcciones disponibles en la reserva global de IANA pertenecientes al protocolo IPv4 se agotaron el jueves 3 de febrero de 2011 oficialmente1. Los Registros Regionales de Internet deben, desde ahora, manejarse con sus propias reservas, que se estima, alcanzaran hasta septiembre de 2011.

El desperdicio de direcciones IPv4 se debe a varios factores. Uno de los principales motivos es que inicialmente no se consideró el enorme crecimiento que iba a tener Internet; se asignaron bloques de direcciones grandes (de 16,271 millones de direcciones) a países, e incluso a empresas.

Otro motivo de desperdicio es que, en la mayoría de las redes, exceptuando las más pequeñas, resulta conveniente dividir la red en subredes. Dentro de cada subred, la primera y la última dirección no son utilizables; de todos modos, no siempre se utilizan todas las direcciones restantes. Por ejemplo, si en una subred se quieren acomodar 80 hosts, se necesita una subred de 128 direcciones (se tiene que redondear a la siguiente potencia de base 2); en este ejemplo, las 48 direcciones restantes ya no se utilizan.

Esta limitación ayudó a estimular el impulso hacia Ipv6, que está actualmente en las primeras fases de implantación, y se espera que termine reemplazando a IPv4.

En la organización que se ocupa de la estandarización de los protocolos de Internet, el IETF (Internet Engineering Task Force) hace unos 15 años que se iniciaron los trabajos para desarrollar el nuevo protocolo IPv6, y desde aproximadamente el año 2002 podemos decir que la base del protocolo, comparada con IPv4, está totalmente terminada y probada. ISOC (Internet Society), de quien depende administrativamente el IETF, también ha apoyado el desarrollo y despliegue de IPv6.

2.4.1.2 IpV6

Versión del protocolo Internet Protocol (IP) y diseñada para reemplazar a IPv4, otorgando un mayor abanico de direcciones para la interconexión de dispositivos a través de Internet. Además del direccionamiento, resuelve grandes problemas de seguridad que anteriormente no se plantearon [19].

IPv6 o Internet Protocol Versión 6, es el protocolo más actual de IP y se posiciona como la actualización de Ipv4 en términos de capacidad, cubrimiento y seguridad.

Las direcciones IPv6 están basadas en 128 bits y este protocolo IPv6 está compuesto por ocho secciones de 16 bits, separadas por dos puntos (:).

IPv6 está pensado como una solución a largo plazo debido al auge de las redes que cada día están en un crecimiento exponencial y si solo estuviéramos con Ipv4 quedaríamos cortos a la hora de asignar direcciones IP.

- No se requiere hacer uso de NAT (Network Address Translation). Este método ha sido usado para poder solucionar la limitación de direcciones [20].
- Elimina la posibilidad de colisiones de direcciones privadas.
- Mayor número de direcciones IP disponibles, ya que mientras IPv4 soporta hasta 2 ^ 32 a nivel de direcciones, IPv6 soportara hasta 2 ^ 128.
- Su formato de encabezado es mucho más simple, lo cual permite un enrutamiento más simple y eficiente de los paquetes.
- Mejoras en la calidad de servicio (QoS).
- Evitará el uso del ya conocido por todos DHCP, un gestor de direcciones IP para los equipos. Se encarga de asignar direcciones a los equipos al iniciarse.
- Mayor seguridad de capa de red incorporada (IPsec).
- Configuración automática de direcciones sin estado para facilitar la administración de la red.

En el siguiente ejemplo, podemos ver una dirección IP en los dos diferentes estándares que existen actualmente de IP:

- IPv4: 192.168.0.25
- IPv6: 2001:0db8:3c4d:0015:0000:0000:1a2f:1a2b

A simple vista podemos ver algunos de los cambios más drásticos entre estos dos protocolos, pero en la siguiente tabla veremos en detalle los cambios específicos entre estos dos tipos de protocolos IP:

Características	IPv4	Ipv6
Dirección	32 bits de longitud (4 bytes).	128 bits de longitud (16 bytes)
Tiempo de vida máximo de la dirección	Gestionadas por DHCP	Las direcciones IPv6 tienen dos tiempos de vida: el preferido y el válido. El tiempo de vida preferido siempre es <= válido.
Máscara de dirección	Es usada para designar la red desde la parte del sistema principal.	No aplica
Tamaño de los paquetes	576 bytes requeridos	280 bytes requeridos
Fragmentación de los paquetes	A través de routers y host de envío	Solo a través de host de envío
Broadcast	Si	No
Cabecera IP	Longitud variable de 20-60 bytes	Longitud fija de 40 bytes
Conexión LAN	Hace uso de la conexión LAN para acceder a la capa física	IPv6 puede ser usado con cualquier adaptador Ethernet y también soporta conexión a través de Ethernet virtual entre particiones lógicas
Filtrado de paquetes	Es un conjunto de funciones básicas de cortafuegos integradas en TCP/IP	El filtrado de paquetes no da soporte a IPv6
Administración de subredes locales	Internet Group Management Protocol (IGMP)	Multicast Listener Discovery (MLD)

Tabla 2–1 Diferencias IPv4 e IPv6 [21].

Podemos ver las notables diferencias entre ambos protocolos y todo es gracias al creciente número de redes y podemos estar seguros que con IPv6 tendremos direccionamiento para mucho tiempo con las prestaciones adecuadas y los mejores niveles de transmisión.

2.4.1.3 ZigBee

Basada en el estándar de la capa de enlace IEEE 802.15.4, ZigBee es una tecnología de bajo costo, baja energía y bajo rendimiento. Opera principalmente en la banda ISM de 2,4 GHz, aunque la especificación también admite las bandas ISM de 868 MHz y 915 MHz. ZigBee puede entregar hasta 250 KBps de rendimiento de datos, pero normalmente se usa a tasas de transferencia de datos mucho menores. También tiene la capacidad de mantener intervalos de suspensión muy largos y ciclos de trabajo de operación baja que se alimentan mediante baterías de celda tipo botón durante años. Los nuevos dispositivos ZigBee que vienen al mercado pueden habilitar técnicas de recolección de energía para una operación sin batería [22].

El estándar ZigBee es mantenido por la ZigBee Alliance. El standard define las capas superiores de red sobre la capa de enlace 802.15.4 y diversos perfiles de aplicaciones permiten implementaciones interoperables del sistema completo. ZigBee puede usarse en múltiples aplicaciones, pero ha adquirido el mayor impulso y éxito en aplicaciones de energía inteligente, domótica y control de iluminación, cada una de las cuales tiene un perfil y certificación específicos de ZigBee. Otra razón por la que le ha ido tan bien al estándar ZigBee en estas áreas de aplicaciones es por la topología de red en malla que puede incluir hasta miles de nodos.

Aunque el estándar ZigBee tiene una especificación IP, está separada de los perfiles populares de enlace de energía inteligente, domótica e iluminación y no ha generado mucha tracción en la industria. Para conectarse a IoT, las redes ZigBee requieren una puerta de enlace de nivel de aplicación. La puerta de enlace participa como uno de los nodos en la red ZigBee y ejecuta en paralelo una pila TCP/IP y aplicación por Ethernet o Wi-Fi para conectar la red ZigBee al Internet [23].

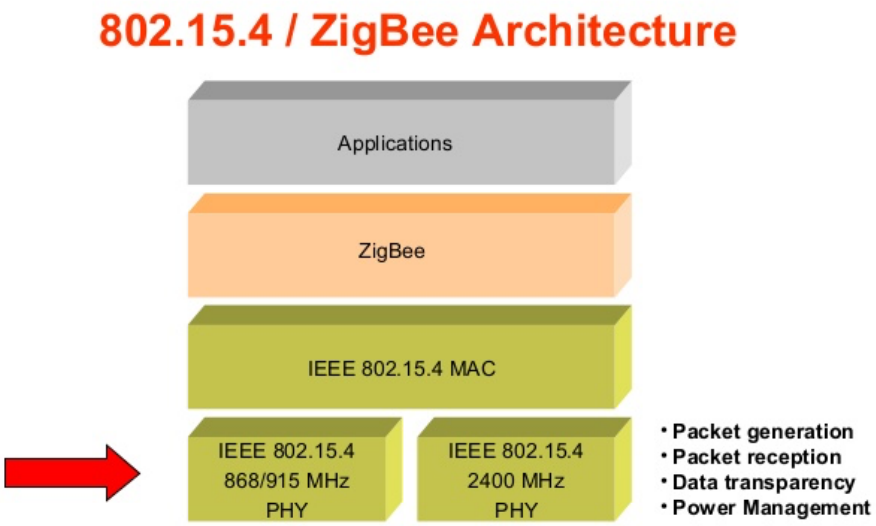


Figura 2-10. Arquitectura ZigBee [24].

2.4.1.4 6LoWPAN

6LoWPAN es un acrónimo para IPv6 over Low power Wireless Personal Area Networks (IPv6 para redes de área personal inalámbricas de baja energía). La promesa de 6LoWPAN es aplicar IP al dispositivo de energía de procesamiento más limitado, más pequeño y con la menor energía. 6LoWPAN es realmente el primer estándar de conectividad inalámbrica que se creó para IoT.

El estándar se creó por el grupo de trabajo 6LoWPAN del IETF y se formalizó bajo la RFC 6282 [25], en septiembre de 2011. El estándar 6LoWPAN solo define una capa de adaptación eficiente entre la capa de enlace 802.15.4 y una pila TCP/IP.

Un dispositivo 6LoWPAN puede comunicarse con cualquier otro servidor o dispositivo basado en IP en Internet, incluidos los dispositivos WiFi y Ethernet.

Se eligió a IPv6 como el único IP admitido en 6LoWPAN (excluido IPv4) porque admite un mayor espacio de direcciones, por lo tanto, redes mucho mayores, y también porque tiene un soporte incorporado para la configuración automática de la red.

Las redes 6LoWPAN requiere una puerta de enlace Ethernet o WiFi para acceder a Internet. De forma similar a Wi-Fi, la puerta de enlace es una puerta de enlace de capa de IP y no una puerta de enlace de capa de aplicaciones, que permite a los nodos y las aplicaciones 6LoWPAN un acceso directo a Internet. Debido a que la mayor parte de la Internet implementada en la actualidad todavía usa IPv4, una puerta de enlace 6LoWPAN generalmente incluye un protocolo de conversión de IPv6 a IPv4.

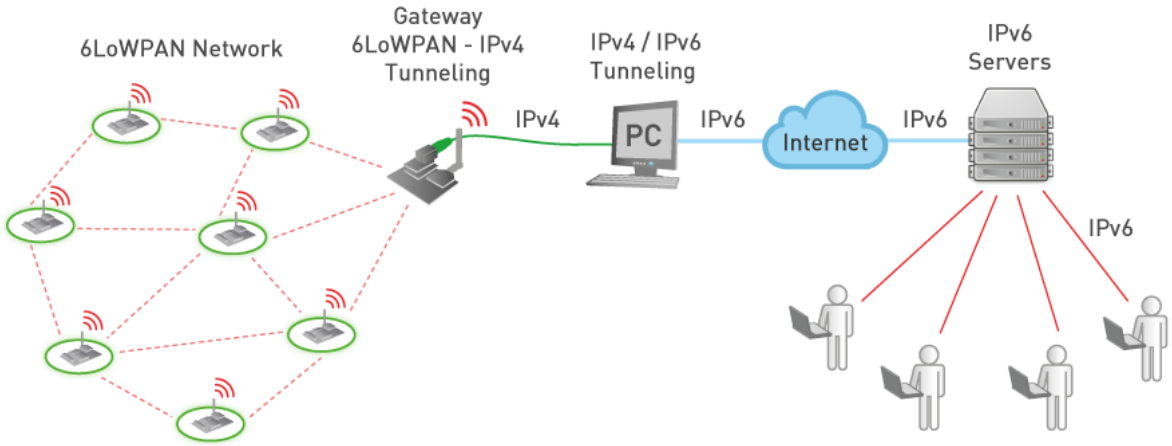


Figura 2-11. Modelo 6LoWPAN [26].

6LoWPAN es bastante nuevo para el mercado. Las implementaciones iniciales usan ambas bandas ISM de 2,4 GHz y 868 MHz/ 915 MHz. Basándose en las ventajas de 802.15.4, topología de red en malla, gran tamaño de red, comunicación confiable y bajo consumo de energía, y en los beneficios de la comunicación IP, 6LoWPAN está en condiciones de alimentar al mercado en expansión de sensores conectados a Internet y otras aplicaciones de bajo rendimiento de datos y operadas con baterías.

2.4.2 Protocolos de Transporte

2.4.2.1 UDP

Protocolo básico de la capa de transporte del modelo OSI usado en aplicaciones cliente/servidor basado en el protocolo IP. UDP es la principal alternativa a TCP y uno de los protocolos de red más antiguos, introducido en 1980. Se utiliza fundamentalmente para aplicaciones en tiempo real [27].

Este protocolo es muy simple ya que no proporciona detección de errores (no es un protocolo orientado a conexión).

Otro escenario de uso de UDP está en programas que requieren transmisión de datos sin pérdida cuando el programa está configurado para administrar la retransmisión de paquetes perdidos y organizar los paquetes recibidos correctamente.

Figura 2-12 UDP

Por lo tanto, el encabezado del segmento UDP es muy simple:

Puerto de origen (16 bits)	Puerto de destino (16 bits)
Longitud total (16 bits)	Suma de comprobación del encabezado (16 bits)
Datos (longitud variable)	

Tabla 2–2 Cabecera UDP

El significado de los campos UDP es el siguiente:

- Puerto de origen: es el número de puerto relacionado con la aplicación del remitente del segmento UDP. Este campo representa una dirección de respuesta para el destinatario. Por lo tanto, este campo es opcional. Esto significa que, si el puerto de origen no está especificado, los 16 bits de este campo se pondrán en cero. En este caso, el destinatario no podrá responder (lo cual no es estrictamente necesario, en particular para mensajes unidireccionales).
- Puerto de destino: este campo contiene el puerto correspondiente a la aplicación del equipo receptor al que se envía.
- Longitud: este campo especifica la longitud total del segmento, con el encabezado incluido. Sin embargo, el encabezado tiene una longitud de 4 x 16 bits (que es 8 x 8 bits), por lo tanto, la longitud del campo es necesariamente superior o igual a 8 bytes.
- Suma de comprobación: es una suma de comprobación realizada de manera tal que permita controlar la integridad del segmento.

2.4.2.2 TCP

Es otro de los protocolos básicos en internet. Su acrónimo, indica que es un protocolo de control de transmisión. Permite que los datos puedan llegar al destino sin errores y en el mismo orden que se transmitieron. Además, ofrece un control de flujo para que la aplicación se vaya adaptando a las condiciones del canal de transmisión [28].

El propósito primordial de TCP es proporcionar circuitos lógicos confiables o servicios de conexión entre parejas de procesos. Esto no implica confiabilidad desde protocolos de más bajo nivel (como IP) así que TCP debe garantizar esto por sí mismo.

TCP reside en el nivel de transporte del modelo de niveles convencionales. Está situado entre IP y los niveles superiores. TCP no está cargado en las pasarelas. Está diseñado para residir en los computadores o en las máquinas que se ocupan de conservar la integridad de la transferencia de datos entre extremos. Lo más común es que TCP resida en los computadores de usuario. Como IP es una red no orientada a conexión, es TCP quien se debe encargar de las tareas de fiabilidad, control de flujo, secuenciamiento, aperturas y cierres. Aunque TCP e IP estén tan relacionados que incluso se les denomine juntos "TCP/IP", TCP puede soportar otros protocolos. Por ejemplo, otro protocolo no orientado a conexión como el ISO 8473 (Protocolo de Redes No Orientado a Conexión o CNLP) podría funcionar con TCP (si se realizan algunos ajustes de los interfaces entre módulos). Además, los protocolos de aplicación, como el Protocolo de Transferencia de Correo Simple (de siglas en inglés, SMTP) se apoyan en muchos servicios que proporciona TCP.

El término asociado con estos aspectos de los protocolos orientados a conexión es el de circuito virtual. Cada octeto transmitido lleva asignado un número de secuencia. El módulo TCP receptor utiliza una rutina de checksum para comprobar la posible existencia de daños en los datos producidos en el proceso de transmisión. Si los datos son aceptables, TCP envía una acepción positiva (ACK) al módulo TCP remitente. Si los datos han resultado dañados, el TCP receptor los descarta y utiliza un número de secuencia para informar al TCP remitente del problema. Como muchos otros protocolos orientados a conexión, TCP emplea temporizadores para garantizar que no transcurre un lapso de tiempo demasiado grande antes de la transmisión de aceptaciones desde el nodo receptor y/o de la transmisión de datos desde el nodo transmisor.

TCP recibe datos de un protocolo de nivel superior de forma orientada a cadenas. Esto es diferente a muchos otros protocolos empleados en la industria. Los protocolos orientados a cadenas se diseñan para enviar caracteres separados y no bloques, tramas, datagramas, etc. Los datos son enviados por un protocolo de nivel superior en forma de cadenas, byte a byte. Cuando llegan al nivel TCP, los bytes son agrupados para formar segmentos TCP. Dichos segmentos se transfieren a IP (o a otro protocolo de nivel inferior) para su transmisión al siguiente destino. La longitud de los segmentos de la determina TCP, aunque el realizador de un determinado sistema puede determinar la forma en que TCP toma su decisión.

TCP comprueba también la duplicidad de los datos. En el caso de que el remitente decida retransmitir los datos, el receptor descarta los datos redundantes. Estos datos redundantes podrían aparecer en la interred, por ejemplo, cuando el receptor no acepta el tráfico de manera temporizada, en cuyo caso el remitente decidirá retransmitir los datos. Además de la capacidad de transmisión de cadenas, TCP soporta también el concepto de función push. Esta función se utiliza cuando una aplicación desea asegurarse de que todos los datos que han pasado al nivel inferior se han transmitido. Para hacer eso, gobierna la gestión del buffer de TCP. Para obtener esta función, el protocolo de nivel superior envía una orden a TCP con un identificador de parámetro de push a 1. Esta operación implica que TCP envía todo el tráfico almacenado en forma de segmento o segmentos hacia su destino. Además de utilizar los números de secuencia para aceptaciones, TCP los utiliza para la reordenación de los segmentos que llegan a su destino fuera de orden. Como TCP descansa sobre un protocolo no orientado a conexión, es bastante posible que en la interred se creen datagramas duplicados.

TCP también elimina los segmentos duplicados. TCP emplea un esquema de aceptación inclusiva. El número de aceptación acepta todos los octetos hasta (e incluyendo) el del número de aceptación menos uno. Este esquema es un método muy sencillo y eficiente de aceptar tráfico, pero presenta una desventaja. Por ejemplo, supongamos que se han transmitido diez segmentos y debido a las operaciones realizadas durante el proceso de encaminamiento llegan desordenados. TCP está obligado aceptar sólo el mayor número de bytes contiguos recibidos sin error. No está permitido aceptar el byte de mayor número recibido hasta que hayan llegado todos los bytes intermedios.

Por tanto, como en cualquier otro protocolo orientado a conexión, podría transcurrir el periodo de temporización de aceptaciones y TCP transmisora retransmitiría el tráfico no aceptado todavía. Esas retransmisiones podrían introducir una considerable sobrecarga en la red. El módulo TCP receptor se ocupa también de controlar el flujo de los datos del transmisor, lo que es muy útil para evitar el desbordamiento de los dispositivos de almacenamiento y la saturación de la máquina receptora. La idea que utiliza TCP es algo poco usual en protocolos de comunicación. Se basa en enviar el dispositivo transmisor un valor de "ventana". Se permite que el transmisor envíe un número máximo de bytes igual al valor de su ventana. Cuando se ha llegado a ese valor, la ventana se cierra y el transmisor debe interrumpir el envío de datos. Además, TCP posee una facilidad muy útil que permite multiplexar varias sesiones de usuario en un mismo computador. Esta operación se realiza definiendo algunas convenciones para compartir puertos y sockets entre usuarios.

TCP proporciona transmisión en modo dúplex integral entre las entidades que se comunican. De esta forma, la transmisión se puede efectuar en ambos sentidos sin necesidad de esperar a la señal de indicación de cambio de sentido, necesaria en las transmisiones semidúplex. Además, TCP permite a los usuarios especificar niveles de seguridad y prioridades de las conexiones. Aunque esas opciones no están incluidas en todos los protocolos TCP, están definidas en el estándar TCP. TCP proporciona el cierre seguro de los circuitos virtuales (la conexión lógica entre dos usuarios). El cierre seguro se ocupa de que todo el tráfico sea reconocido antes de la desactivación del circuito virtual.

Las PDU que se intercambian entre dos módulos TCP se denominan segmentos. El segmento se divide en dos partes, la parte de cabecera y la parte de datos. La parte de datos sigue a la parte de cabecera. Los primeros dos campos del segmento se denominan puerto de fuente y puerto de destino. Esos campos de 16 bits identifican a los programas de aplicación de nivel superior que utilizan la conexión TCP [29].

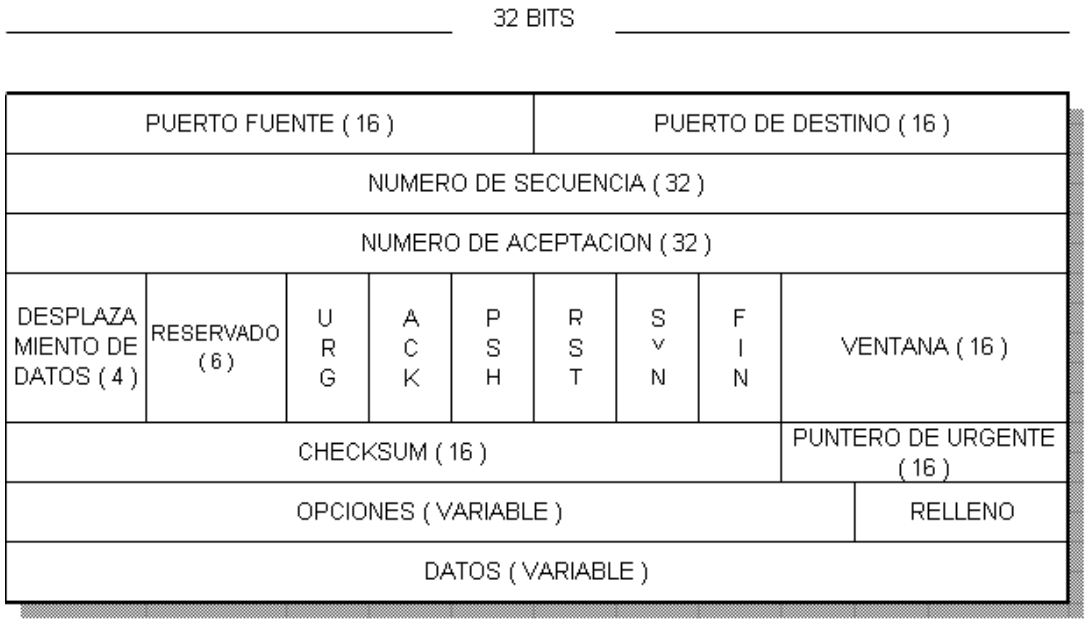


Figura 2-13. Cabecera TCP.

El siguiente campo se denomina número de secuencia. Este campo contiene el número de secuencia del primer octeto del campo de datos de usuario. Su valore especifica la posición de la cadena de bits del módulo transmisor. Dentro del segmento especifica el primer octeto de datos de usuario.

El número de secuencia se utiliza también durante la operación de gestión de la conexión. Si dos entidades TCP utilizan el segmento de solicitud de conexión, entonces el número de secuencia especifica el número de secuencia de envío inicial (ISS) que se utilizará para la numeración subsiguiente de los datos de usuario.

El valor del número de aceptación permite aceptar los datos previamente recibidos. Este campo contiene el valor del número de secuencia del siguiente octeto que se espera recibir del transmisor. Con esa definición permite la aceptación inclusiva, en el sentido de que permite la aceptación de todos los octetos hasta, e incluyendo, el valor de este número menos 1.

El campo de desplazamiento de datos especifica el número de palabras alineadas de 32 bits de que consta la cabecera de TCP. Este campo se utiliza para determinar dónde comienza el campo de datos [30].

Como puede esperarse, el campo reservado está reservado. Consta de 6 bits que deben valer cero. Estos bits están reservados para usos futuros.

Los seis bits siguientes se denominan indicadores (flags). Son bits de control de TCP y se utilizan para especificar ciertos servicios o utilidades que se pueden emplear durante la sesión. El valor de algunos de esos bits indica cómo interpretar otros campos de la cabecera. Los seis bits mencionados llevan la siguiente información.

- URG indica que el campo de puntero de urgencia es significativo.
- ACK indica si el campo de aceptación es significativo.
- PSH significa que el módulo va a utilizar la función push.
- RST indica que la conexión se va a inicializar.
- SYN indica que se van a sincronizar los números de secuencia; se utiliza en los segmentos de establecimiento de conexión como indicación de que se van a realizar algunas operaciones de preparación.
- FIN indica que el remitente no tiene más datos para enviar. Es comparable a la señal de fin de

transmisión (EOT) en otros protocolos.

El campo siguiente, denominado ventana, se pone a un valor que indica cuántos octetos desea aceptar el receptor. Este valor se establece teniendo en cuenta el valor del campo de aceptación (número de aceptación). La ventana se establece sumando los valores del campo de ventana y del campo de número de aceptación.

El campo de checksum contiene el complemento de 1 a 16 bits del complemento a 1 de la suma de todas las palabras de 16 bits del segmento, incluyendo la cabecera del texto. El propósito de este cálculo es determinar si el segmento procedente del transmisor ha llegado libre de errores.

El siguiente campo del segmento, denominado puntero de urgente, se utiliza sólo si el indicador de URG está a 1. El objeto de este puntero es identificar el octeto de datos al que siguen datos urgentes. Los datos urgentes se denominan datos fuera de vida. TCP no dice lo que hay que hacer con los datos urgentes. Depende de la implementación. Dicho de otro modo, sólo se indica el lugar donde empiezan los datos urgentes, no lo que hay que hacer con ellos. El valor de este campo es un desplazamiento del número de secuencia y apunta al octeto a partir de cual siguen los datos urgentes. El campo de opciones está concebido para posibilitar futuras mejoras de TCP. Está diseñado de forma semejante al campo de opción de los datagramas de IP, en el sentido de que cada opción se especifica mediante un byte que especifica el número de opción, un campo que contiene la longitud de la opción y finalmente, los valores de la opción propiamente dichos.

Finalmente, el campo de relleno asegura que la cabecera TCP ocupa un múltiplo par de 32 bits.

2.4.3 Protocolos de Aplicación

2.4.3.1 MQTT

MQTT (Message Queue Telemetry Transport), que está construido sobre la pila de TCP/IP, se ha convertido en uno de los estándares para las comunicaciones de IoT [31].

Originariamente, MQTT fue inventado y desarrollado por IBM a finales de los 90. Su aplicación original era conectar sensores de los oleoductos con satélites. Tal como sugiere su nombre, es un protocolo de mensajería que soporta la comunicación asíncrona entre las partes. Un protocolo de mensajería asíncrona disocia al emisor y al receptor de los mensajes tanto en espacio como en tiempo, y, por lo tanto, es escalable en entornos de red no confiables. A pesar de su nombre, no tiene nada que ver con las colas de mensajería, y, en cambio, utiliza un modelo de publicación y suscripción. A finales del 2014, se convirtió oficialmente en un estándar abierto de OASIS, y se soporta en lenguajes de programación populares mediante la utilización de varias implementaciones de código abierto [32].

MQTT habilita el modelo Publicador/Suscriptor de una manera ligera. Permite la comunicación con servidores con un uso reducido de ancho de banda. Este protocolo está orientado a la comunicación de sensores, debido a que consume muy poco ancho de banda y puede ser utilizado en la mayoría de los dispositivos empotrados con pocos recursos (CPU, RAM, …).

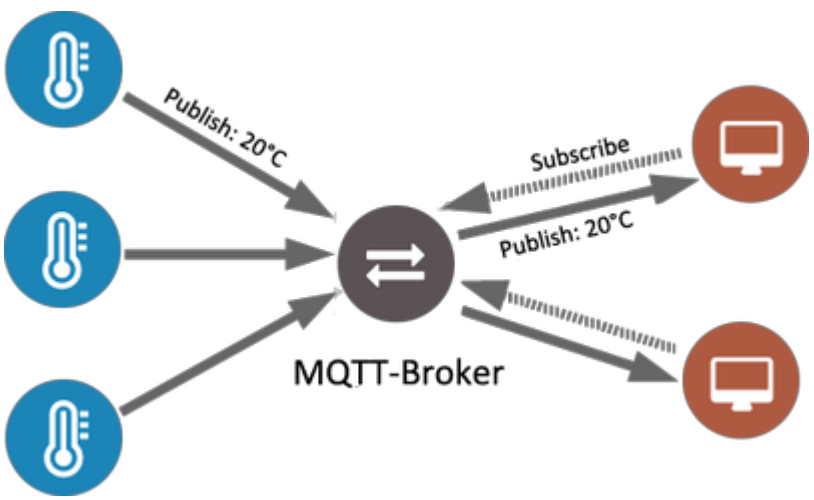


Figura 2-14. Protocolo MQTT.

El protocolo MQTT define los tipos de entidades en la red: un intermediario de mensajes y un número de clientes. El intermediario es un servidor que recibe todos los mensajes de los clientes y luego los redirige a clientes de destinos relevantes. Un cliente es cualquier cosa que pueda interactuar con el intermediario para enviar y recibir mensajes. Un cliente puede ser un sensor de IoT en el campo o una aplicación del centro de datos que procesa datos de IoT.

1. El cliente se conecta con el intermediario. Se puede suscribir a cualquier "tema" de mensajes de intermediario. Esta conexión puede ser una conexión TCP/IP simple o una conexión TLS cifrada para mensajes confidenciales.
2. El cliente publica el mensaje, sobre un tema, enviando el mensaje y el tema al intermediario.
3. Después, el intermediario redirige el mensaje a todos los clientes que están suscritos a ese tema.

Ya que los mensajes MQTT están organizados por temas, el desarrollador de aplicaciones tiene la flexibilidad de especificar que ciertos clientes sólo puedan interactuar con determinados mensajes. Por ejemplo, los sensores publicarán sus lecturas sobre del tema "sensor_data" y se suscribirán al tema "config_change". Las aplicaciones de procesamiento de datos que guardan datos del sensor en una base de datos backend se suscribirán al tema "sensor_data". Una aplicación de consola de administración puede recibir los comandos del administrador del sistema para ajustar las configuraciones de los sensores, como la sensibilidad y la frecuencia de la muestra, y publicar esos cambios en el tema "config_change".

2.4.3.2 XMPP

Tecnología abierta para comunicación en tiempo real. Es el mejor protocolo para conectar dispositivos con personas. Se trata de un caso especial del patrón D2S (Devices to Servers). Sus siglas son una abreviación para el término *Extensible Messaging Presence Protocol* [33].

XMPP es un protocolo abierto que se creó para ser usado en sistemas de mensajería instantánea originalmente, está basado en XML. Originalmente se conocía como Jabber, y el proyecto fue iniciado en 1998 por Jeremie Miller. Actualmente XMPP y sus múltiples extensiones soportan mensajería instantánea, videoconferencia, vista del estado en línea de los usuarios, y transferencia de archivos a través de clientes.

Debido a que es un protocolo abierto y confiable, muchas empresas lo adoptaron como el estándar para sus servicios de chat, entre las que podemos mencionar servicios tan populares como Whatsapp, Facebook Messenger, y Gtalk. XMPP funciona de manera completamente descentralizada, está documentado, y puede ser usado en cualquier proyecto. Existen muchos clientes y servidores libres que se pueden usar de manera gratuita [34].

2.4.3.3 DDS

El primer estándar abierto e internacional que acoge el patrón Publicador/Suscriptor para comunicaciones en tiempo real de dispositivos embebidos. DDS usa una arquitectura sin Broker o servidor. Se puede ver como un bus de datos que integra y comunica máquinas inteligentes [35].

Este protocolo usa la difusión de contenido para proveer una elevada calidad de servicio a las aplicaciones que hagan uso del mismo.

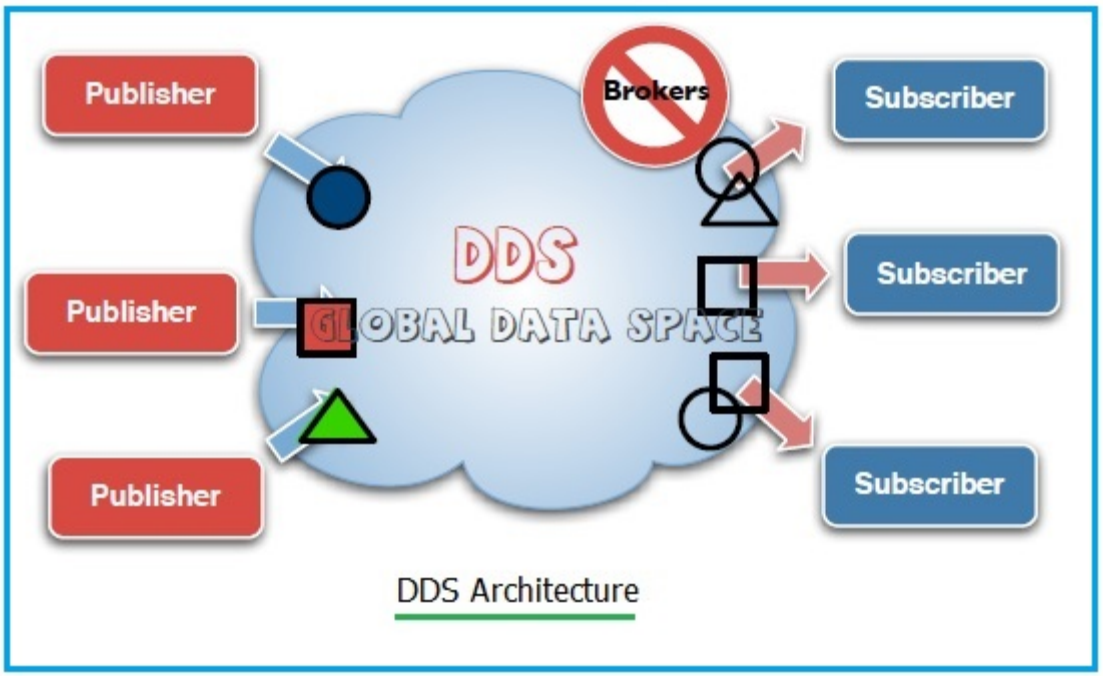


Figura 2-15. Arquitectura DDS [36].

DDS es un espacio global de datos distribuido. Las especificaciones de un espacio global de datos hacen de este protocolo de una buena solución para las aplicaciones que demandan la reducción de los cuellos de botella.

En dicha arquitectura, las aplicaciones pueden leer o escribir de forma autónoma y asíncrona. Además, estas pueden unirte o dejar el espacio de datos en cualquier instante ya que son descubiertos en dicho espacio de forma dinámica.

El resto del funcionamiento del protocolo DDS es muy parecido a MQTT. Los publicadores y suscriptores expresan su intención de producir o consumir información de determinado tipo según el teman escogido [37].

2.4.3.4 AMQP

Sistema de colas diseñado para la conexión entre servidores. Sus características fundamentales son el encolamiento, enrutamiento, seguridad y fiabilidad [38].

Este protocolo manda mensajes de transacciones entre servidores. Actúa como un middleware centrado en mensaje cuyo origen es la industria bancaria. Este puede procesar miles de colas de transacciones.

AMQP está centrado en evitar la pérdida de paquetes. Las comunicaciones de los publicadores a los servidores y desde las colas de los mismos a los suscriptores se apoyan sobre TCP, el cual provee una conexión punto a punto de confianza. Además, los puntos finales deben de enviar un acuse de recibo de cada mensaje.

Actualmente, dicho protocolo se usa en aplicaciones de banca. Este protocolo en IoT es usado principalmente para el plano de control de las funciones de análisis basada en servidores [39].

2.4.3.5 DTLS

Datagram Transport Layer Security (DTLS) [40], es un protocolo de comunicaciones diseñado para proteger la privacidad en las comunicaciones y prevenir su intercepción y manipulación. Está basado en el protocolo Transport Layer Security (TLS), el cual es un protocolo que provee seguridad a redes de comunicaciones basadas en computadoras. La principal diferencia entre DTLS y TLS es que el DTLS usa UDP y TLS utiliza TCP. Es usado para navegación web, correo electrónico, mensajería instantánea y voIP [41].

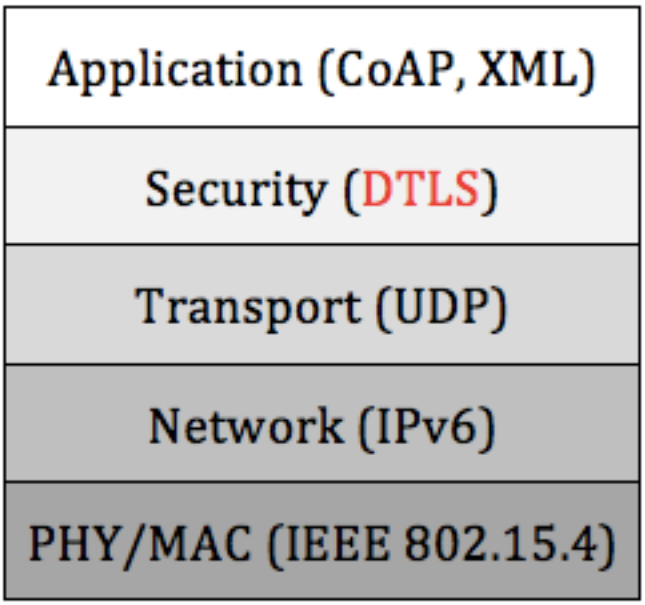


Figura 2-16. DTLS sobre UDP.

2.4.3.6 HTTP

Hypertext Transfer Protocol (HTTP) (o Protocolo de Transferencia de Hipertexto en español) es un protocolo de la aplicación para la transmisión de documentos hipermedia, como HTML [42]. Fue diseñado para la comunicación entre los navegadores y servidores web, aunque puede ser utilizado para otros propósitos también. Sigue el clásico modelo cliente-servidor, en el que un cliente establece una conexión, realizando una petición a un servidor y espera una respuesta del mismo. Se trata de un protocolo sin estado, lo que significa que el servidor no guarda ningún dato (estado) entre dos peticiones. Aunque en la mayoría de casos se basa en una conexión del tipo TCP/IP, puede ser usado sobre cualquier capa de transporte segura o de confianza, es decir, sobre cualquier protocolo que no pierda mensajes silenciosamente, tal como UDP.

Con HTTP se establecen criterios de sintaxis y semántica informática (forma y significado) para el establecimiento de la comunicación entre los diferentes elementos que constituyen la arquitectura web: servidores, clientes, proxies. Fue creado en 1999 por el World Wide Web Consortium en colaboración con la Internet Engineering Task Force [43].

El funcionamiento del http se basa en un esquema de petición-respuesta entre el servidor web y el "agente usuario" (del inglés user agent) o cliente que realiza la solicitud de transmisión de datos. Un cliente puede ser un explorador determinado, cuando intentamos abrir una página web, o los rastreadores web (webcrawlers o arañas web) que las inspeccionan.

A ellos el servidor brinda una respuesta estructurada de modo puntual y dotada de una serie de metadatos, que establecen las pautas para el inicio, desarrollo y cierre de la transmisión de la información. Estos son los "métodos de petición", es decir, los comandos que disparan la ejecución de recursos determinados, cuyos archivos residen en el servidor.

2.4.3.7 CoAP

Protocolo de nivel de aplicación cuyo uso se da en dispositivos de Internet con recursos limitados, como nodos WSN. CoAP está diseñado para una fácil traducción a HTTP con el fin de simplificar la integración con la web, y al mismo tiempo satisfacer necesidades especializadas como soporte multicast, bajo coste y simplicidad ya que son muy importantes para el Internet de las cosas [44].

CoAP implementa el modelo REST de HTTP (con las primitivas GET, POST, PUT y DELETE), usa cabeceras reducidas, y limita el intercambio de mensajes, añadiendo soporte UDP y otras modificaciones como mecanismos de seguridad específicos.

2.5 Plataformas IoT

De la misma forma que hay numerosas aplicaciones basadas en la IoT, existen multitud de plataformas que permiten que el desarrollo de aplicaciones sea más amigable. A modo de introducción, se mencionan las características generales de cinco plataformas de IoT.

2.5.1 FiWare

FiWare es una iniciativa open source (código abierto en español), financiada por la Unión Europea, que pretende impulsar la creación de estándares necesarios para desarrollar aplicaciones Smart en diferentes dominios: Smart Cities, Smart Ports, Smart Logistics, Smart Factories, entre otros. Cualquier aplicación Smart se caracteriza por recoger información relevante para la aplicación de diferentes fuentes sobre lo que está pasando en un momento dado. Esto se conoce como "información de contexto". La información de contexto actual e histórica se procesa, visualiza y analiza a gran escala. De esta forma, se produce el comportamiento inteligente esperado [45].

FiWare quiere impulsar un estándar que describe cómo recopilar, gestionar y publicar información de contexto y adicionalmente aporta elementos que permiten explotar esta información una vez recopilada. Ese estándar no existe en la actualidad y resulta clave para construir un mercado digital único para las aplicaciones inteligentes donde las apps y soluciones pueden portarse de un cliente a otro sin grandes cambios. También resuelve de manera sencilla cómo capturar información procedente de redes de sensores, aunque se comunican usando diferentes protocolos y lenguajes IoT. En ese sentido es capaz de resolver la complejidad de tratar la información recogida por los sensores y traducirlos a un lenguaje común.

Figura 2-17. Plataforma FiWare [46].

2.5.2 Amazon Web Service

Amazon Web Service para IoTm provee una colección de servicios de computación que en conjunto forman una plataforma de computación en la nube, ofrecida a través de Internet por Amazon.com. Es usado en aplicaciones populares como Dropbox o en reconocidas organizaciones como la NASA [47].

Como vemos en la siguiente imagen, Amazon Web Service cuenta con amplitud de servicios que van desde el borde de los dispositivos IoT hasta la nube. Esto proporciona una flexibilidad para poder combinar nuevos servicios y poder desplegar nuevos dispositivos. Además, este servicio IoT puede ser integrado con otros servicios que Amazon Web Service ofrece, pudiendo crear aplicaciones IoT completas.

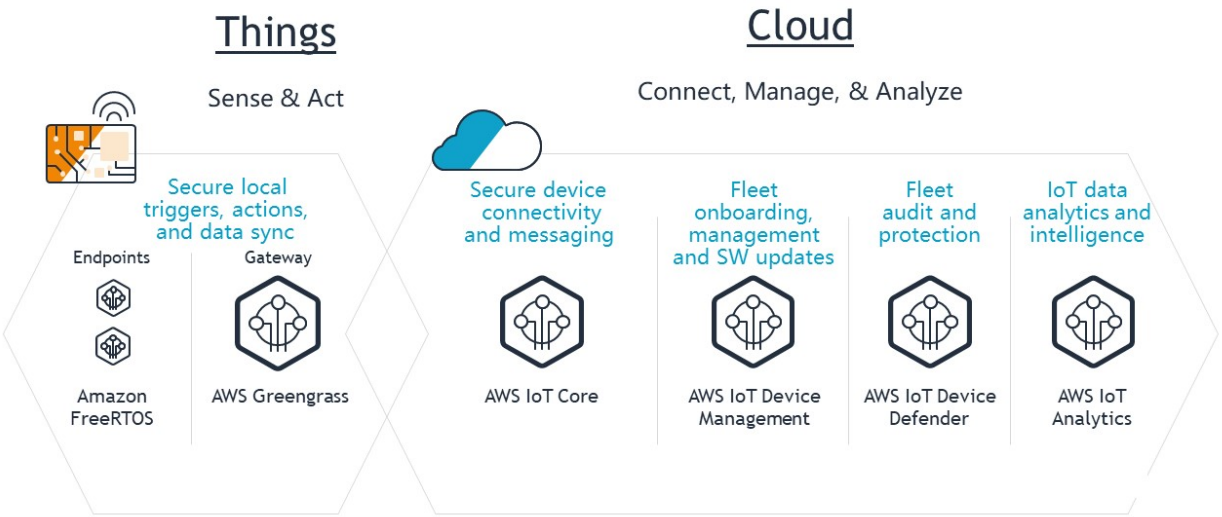


Figura 2-18. Amazon Web Service IoT [47].

El funcionamiento del servicio IoT de Amazon Web Service es bastante trivial, primero conecta una amplia gama de dispositivos como sensores, bombillas y robots a la nube. Una vez que los dispositivos están conectados de forma segura, AWS IoT direcciona los mensajes de los dispositivos a los servicios en la nube y a otros dispositivos de borde. También almacena los datos de IoT para que pueda consultarlos y generar modelos analíticos y de aprendizaje automático.

2.5.3 Google Cloud

Google Cloud IoT es un conjunto de servicios completamente administrado e integrado que te permite conectar, administrar e ingerir datos a gran escala, así como de forma fácil y segura, a partir de dispositivos repartidos por todo el mundo. También te ayuda a procesar, analizar y ver datos en tiempo real, implementar cambios operacionales y llevar a cabo las acciones que creas pertinentes [48].

Cloud IoT Core recopila datos de los dispositivos, que más tarde se publican en Cloud Pub/Sub para su análisis.

Entre sus funcionalidades se encuentran la elevada implementación de seguridad completa del ecosistema mediante técnicas de claves asimétricas y otros certificados de firma; la capacidad de integración con otros servicios Cloud distribuyendo los datos para su manipulación y almacenamiento; la capacidad de realizar cambios en el estado de los dispositivos en tiempo real obteniendo una optimización de los procesos empresariales; la compatibilidad con servicios Cloud de análisis de datos avanzados pudiendo derivar a métodos de inteligencia artificial; y la implementación de un solo sistema global al cual se le pueden conectar dispositivos siguiendo protocolos estándar (como HTTP o MQTT) administrando todos estos como si fuese un sistema único y global.

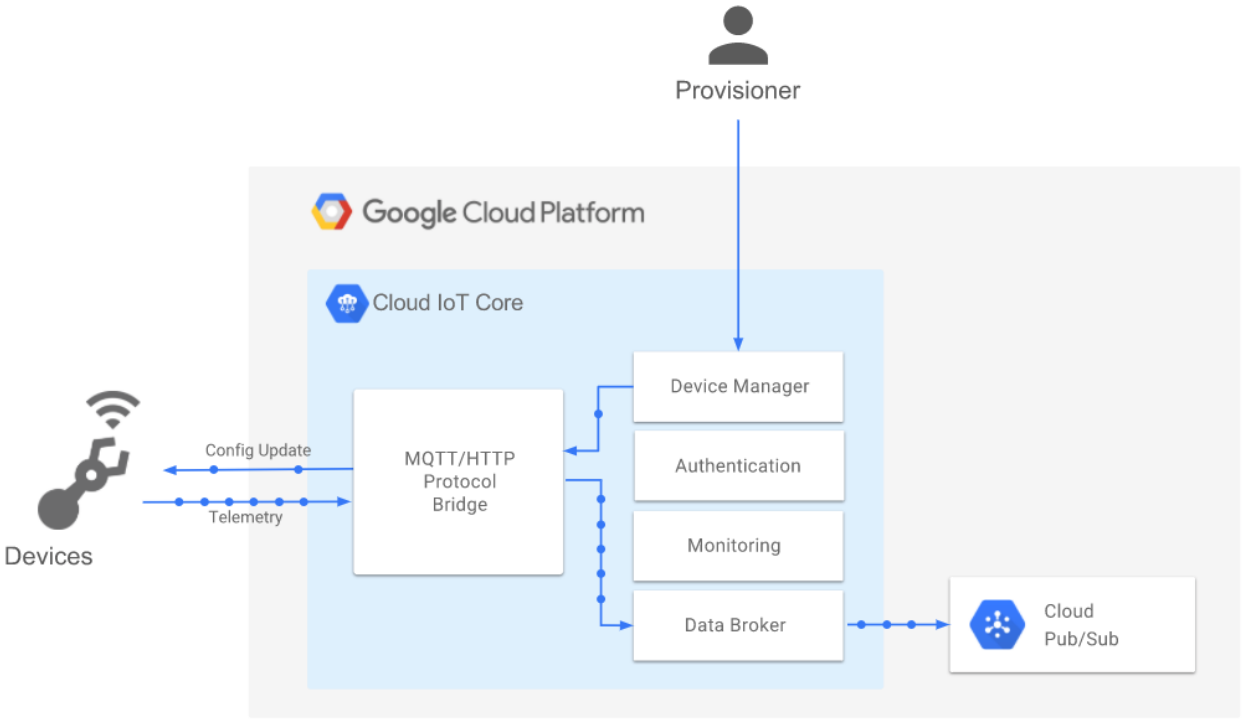


Figura 2-19. Google Cloud Platform IoT [48].

2.5.4 Azure

Como Amazon y Google han hecho, Azure también ha desplegado una extensa variedad de servicios Cloud específicos para IoT. Grandes empresas como Rolls-Royces y Scheneider confían en Microsoft y sus soluciones Cloud para la implementación de sus ecosistemas IoT [49].

Este servicio, ofrece un potente acelerador de soluciones en la nube de IoT la cual permite conectar los dispositivos hacia una infraestructura cloud de manera ágil y flexible. Además, Azure ofrece una compatibilidad perfecta con Stream Analytics, lo que permite almacenar analíticas de los datos obtenidos de los dispositivos para usarlos en desarrollos web o en mecanismos de inteligencia de negocio.

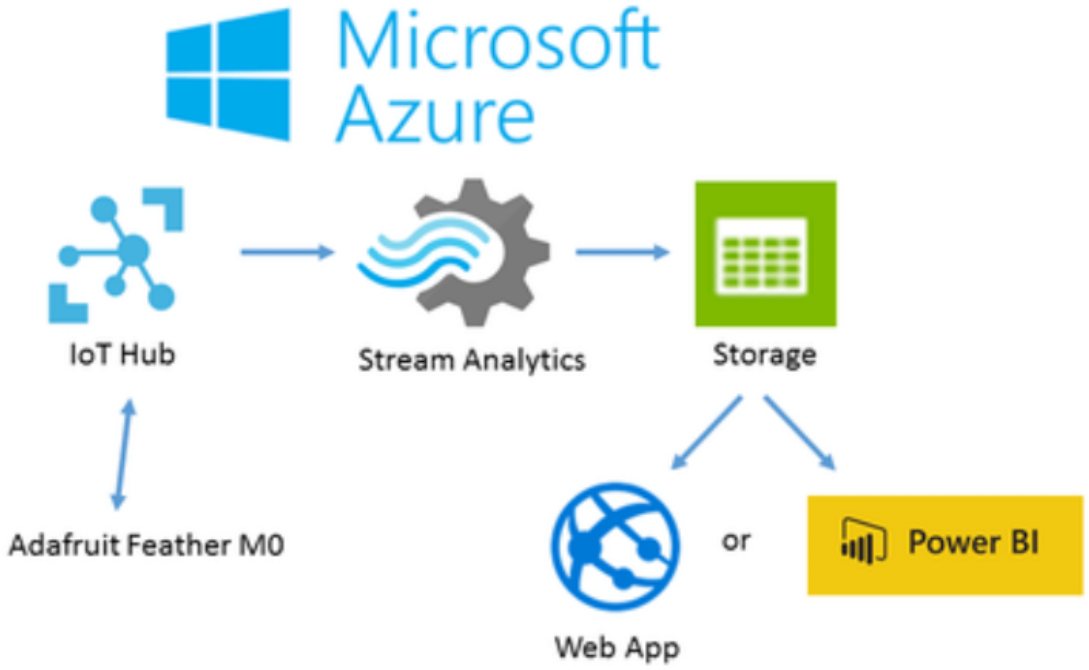


Figura 2-20. Microsoft Azure IoT [49].

2.5.5 IBM Watson IoT

Watson IoT es la plataforma de IBM para el internet de las cosas. Esta plataforma ingiera datos de dispositivos y los transforma para proveer de inteligencia en la optimización de procesos y ciclos de vida de productos. Por su naturaleza, Watson IoT está pensada principalmente para procesos industriales [50].

IBM ha trabajado muy duro en implementar una conexión y gestión de dispositivos seguros en su plataforma. Provee de capacidades de registro y conexión de dispositivos de manera rápida y secura por diseño y basado en estándares abiertos para poder satisfacer todo tipo de necesidades. Además, es capaz de capturar, procesar y almacenar los datos para rápidamente transformar estos en bienes valiosos.

Figura 2-21. IBM Watson [50].

3 ESTADO DE LA TECNOLOGÍA

En este mundo hay que aprender a observar primero y hacer las preguntas adecuadas después

- Santiago Posteguillo. Los asesinos del emperador -

3.1 Requisitos Ecosistema IoT

La complejidad de las redes de sensores hace necesaria la utilización de plataformas de gestión de los dispositivos que formen parte de dicha estructura. Estas plataformas deben exhibir una serie de características, principalmente: deben de permitir la interconexión de dispositivos heterogéneos, deben ser escalables con el tamaño de la red, deben permitir el registro y búsqueda de dispositivos y servicios, deben ser tolerantes a fallos e incluir utilidades para la estimación o interpolación de datos en aquellas áreas no cubiertas por los sensores.

3.2 Patron Diseño Publicador/Suscriptor

Internet ha cambiado considerablemente la escala de los sistemas distribuidos. Actualmente dichos sistemas abarcan cientos de entidades cuyas localizaciones y comportamientos pueden variar significativamente. Estas restricciones hacen necesario modelos y sistemas de comunicación más flexibles que sean capaces de reflejar el dinamismo y el desacople de las características entre aplicaciones. Como consecuencia directa, el paradigma de comunicación Publicador/Suscriptor está recibiendo progresivamente mayor atención. En los sistemas basados en los esquemas de interacción Publicador/Suscriptor, los suscriptores registran su interés en un evento o en un grupo de eventos, siendo notificados posteriormente de los eventos generados por los publicadores. Así, han aparecido una gran cantidad de variaciones basadas en esta idea, cada una adaptada a unos requisitos o características específicas del modelo de aplicación o red.

Básicamente, los suscriptores tienen la capacidad de expresar su interés en un evento o eventos de características determinadas, siendo posteriormente notificados por cualquier evento cuyas características cumplan los requisitos de la suscripción previa.

Un evento se propaga de manera asíncrona a todos los suscriptores que han registrado un interés en él. La potencia de este estilo de interacción basado en eventos radica en el desacople total en tiempo, espacio y sincronización entre publicadores y suscriptores.

Muchos sistemas industriales apoyan este estilo de interacción, existiendo actualmente un notable auge en investigaciones basadas en los esquemas de interacción Publicador/Suscriptor.

3.3 FiWare

FiWare es una nueva infraestructura en la nube (cloud) para la creación y despliegue global de servicios y aplicaciones en la Internet del Futuro, surgida a partir de la Comisión Europea y las principales empresas TIC europeas.

El objetivo esencial de FiWare es ayudar al desarrollo e implantación de nuevos servicios, proporcionando un conjunto de APIs para el desarrollo rápido de aplicaciones en numerosos sectores, facilitando la reutilización e introduciendo estándares.

Separa la cadena de valor de las aplicaciones, de forma que la plataforma, el desarrollo de los servicios, su despliegue, etc. puedan ser proporcionados por entidades diferentes, ayudando de esta forma a su expansión y mejorando la competitividad para acelerar la expansión comercial.

El ecosistema de FiWare está formado por varios componentes que se encargan de diversas funcionalidades

que aportan valor a lo que al final FiWare quiere disponer, que no es otra cosa que, un ecosistema en el que se puedan crear aplicaciones de forma gratuita y que cuente con la implementación de una serie de estándares que sirvan a los usuarios para facilitar la creación de aplicaciones inteligentes en múltiples sectores.

3.3.1 Platform FiWare

La plataforma FiWare ofrece al usuario una serie de potentes APIs que facilitan el desarrollo de aplicaciones inteligentes (Smart). Las especificaciones de estas APIs son públicas, así como las APIs son gratuitas y de código abierto.

3.3.2 FiWare Lab

FiWare Lab es un entorno no comercial donde se pueden desarrollar las aplicaciones FiWare, ofrece la posibilidad de que los usuarios pueden testear sus aplicaciones, usando datos abiertos publicados por ciudades y organizaciones.

3.3.3 FiWare Accelerate

El programa FiWare Accelerator es un programa destinado a fomentar el desarrollo de aplicaciones FiWare, enfocándose esencialmente en empresas pequeñas.

3.3.4 Fiware Mundus

Aunque el programa FiWare naciera en Europa es un programa que tiene una vocación mundial, el programa Mundus ha sido diseñado para ofrecer una cobertura global del proyecto, ofreciéndolo a entidades y gobiernos de países alrededor del mundo.

3.3.5 Fiware iHubs

El rol principal de esta plataforma es el desarrollo de una comunidad de desarrolladores que actúen a nivel local para dar soporte y poder ayudar a los nuevos desarrolladores, esto aplicado a nodos alrededor del mundo es el objetivo de iHubs.

3.4 Plataforma FiWare

La documentación del funcionamiento de la plataforma FiWare puede encontrarse en su página web [51]. En ella, encontraremos la documentación de todos los integrantes de la arquitectura, junto con casos de uso y documentaciones de buenas prácticas para el despliegue de los mismos.

En la web de FiWare también se encuentran los repositorios necesarios para descargar el entorno en el que se desarrolla FiWare, así como una serie de guías en las que se explican diferentes aplicaciones de la plataforma y diversos proyectos en los que la herramienta ha sido usada. Las guías disponibles siguen una temática diferente, cada una de ellas hace una explicación de algunas de las herramientas y funcionalidades disponibles haciendo una presentación de esta, seguida con una descripción de las herramientas disponibles y como se usan, ofreciendo ejemplos para que la comprensión de lo que se quiere transmitir sea lo más completa posible y que usuario final pueda reusar lo aprendido para implementarlo en sus propias aplicaciones.

Dispone en paralelo de una plataforma de e-learning de la que su objetivo es ayudar al usuario que se quiere iniciar en FiWare a seguir una línea de conocimientos mediante los que pueda obtener unos conocimientos fuertes que le sirvan para el aprovechamiento en sus proyectos personales/profesionales de los conocimientos adquiridos.

De manera colateral se consigue mediante la ayuda al aprendizaje que la plataforma adquiera una mayor base de usuarios que repercutirá en una mejor base de desarrolladores en los foros, que puedan guiar y ayudar a los nuevos usuarios, así como ayudar a crear una base de proyectos y partes reutilizables del software que en conjunto mejoren el nivel de la plataforma en su conjunto convirtiéndola en una plataforma versátil.

FiWare es una estructura que facilita el desarrollo de aplicaciones destinadas a Smart Cities y garantiza una

serie de características que la hacen apetecible a la hora de ser elegida como plataforma de soporte del proyecto, en otras palabras proporciona una plataforma Open Source de temática a elegir por el usuario a partir de estándares propios que permite integrar aplicaciones desarrolladas por empresas de distinta índole, practicando la técnica de reutilización de software que permite a los desarrolladores no tener que empezar de cero y poder reaprovechar desarrollos anteriores, repercutiendo esto en la mejora de la calidad, debido a que permite dejar trabajos específicos en manos de expertos que se encargan de perfeccionar ciertos módulos, que dispondrán de una interfaz sencilla para los usuarios.

FiWare ofrece una serie de ventajas que son importantes a la hora de desarrollar aplicaciones vinculadas a las Smart Cities que se enumeran y explican a continuación.

- Fácil de implementar: Ofrece una plataforma versátil capaz de abarcar las distintas áreas, lo que es muy útil a la hora de crear e implementar servicios destinados a Smart Cities.
- Garantiza la interoperabilidad: Al hablar de FiWare se habla de una plataforma abierta que proporciona herramientas y conforma un ecosistema destinado a la investigación que garantiza la interoperabilidad y el establecimiento de modelos.
- Fácilmente escalable: Se trabaja en un ecosistema donde el software es abierto, por lo cual va evolucionando y es posible mejorar módulos ya implementados por anteriores usuarios, dándoles de nuevas y mejores funcionalidades sin tener que enfrentarse nuevamente a los problemas que el desarrollador anterior se enfrentó.
- Normas abiertas API: FiWare ofrece una biblioteca con una serie de componentes con un conjunto de funciones y herramientas y librerías, cuyas interfaces son públicas y sin derechos de autor, lo que permite que puedan ser usadas por cualquiera que las necesite, solo las tiene que incluir en sus proyectos y podrá disfrutar de las bondades de la abstracción de software que le permiten estas.
- Permite el desarrollo de aplicaciones muy potentes y alimentadas en tiempo real: La plataforma en su concepción ha sido pensada para soportar aplicaciones en tiempo real capaces de tratar grandes cantidades de información sin perder los datos y ofrecer un respaldo para los mismos. Ahondando en el tema de las aplicaciones en tiempo real por regla general suelen tratarse de aplicaciones IoT, para las cuales ofrece conectores con el fin de normalizar el tráfico de estos datos, que pueden ser utilizados por usuarios finales de forma transparente sin importar el método de obtención, ni el protocolo de comunicaciones empleando para el transporte de los datos.
- Lanzamiento más rápido y eficiente: Es una de las grandes ventajas de la reutilización del software, permite no tener que enfrentarse a los mismos problemas que anteriores desarrolladores se han enfrentado a la hora de desarrollar la tecnología, centrando los esfuerzos en las mejoras y disminuyendo el tiempo necesario para la implementación, ya que siempre es más fácil conectar algo implementado con tu aplicación que crearlo desde cero. Las herramientas que el sistema pone a disposición del usuario en el apartado gráfico le permiten que la muestra de los datos y la creación de servicios web le sean más fáciles al hacerle transparentes muchas de las conexiones y servicios.
- Fácilmente replicable: Frente a soluciones propietarias en las que los servicios suelen estar ubicados en distintos sitios con conexiones complejas entre ellos, FiWare se ofrece como una herramienta fácilmente replicable.
- Facilita la información de contexto de manera masiva: FiWare dispone de la posibilidad de la muestra de los datos en tiempo real, aunque no solo de eso sino también de la posibilidad de interactuar con esos datos en tiempo real, debido a su construcción, principalmente a un elemento de su arquitectura como es el Context Broker se consigue generar, recoger, analizar en tiempo real y publicar esa información como datos abiertas que pueden ser utilizados por otras aplicaciones

3.5 Generic Enablers

La plataforma FiWare está formada por un conjunto de elementos llamados “Generic Enablers” los cuales son considerados de propósito general e independientes de alguna otra área de uso. Estos elementos, ofrecen interfaces abiertas tanto para el desarrollo de aplicaciones mediante APIs, como para la interoperabilidad con otros “Generic Enablers”.

Para el despliegue del ecosistema FiWare, se ensamblan una serie de elementos o “Generic Enablers” para

facilitar la creación de aplicaciones web inteligentes. Estas ofrecen funciones comunes y reusables usadas en múltiples casos de uso en varios sectores.

Las especificaciones para nuevos GE son de código abierto pudiendo ser modificadas por cualquier persona para su propio uso.

Según el proyecto FiWare, los "Generic Enablers" se dividen en siete partes técnicas.

3.5.1 Cloud Hosting

Los componentes que, enmarcados en esta parte técnica, son los encargados de ofrecer recursos cloud desde infraestructura y parte hardware, hasta software. Son los encargados del despliegue y provisión de máquinas virtuales, de almacenamiento robusto y escalable, etc. Además, ofrece elementos para la gestión de arquitecturas de despliegue y de gestión de políticas basadas en reglas. Por último, ofrecen soporte al despliegue automático de software y mecanismos de monitorización de los mismos.

Figura 3-1. Cloud Hosting.

3.5.2 Data/Context Management

Los elementos denominados como "Data/Context Management" son los encargados del procesamiento de datos, desde el despliegue de motores de análisis Big Data mediante un ecosistema Hadoop, hasta procesamiento de eventos en tiempo real y la gestión de información de contexto. Por último, en esta parte técnica, encontramos elementos responsables de la inclusión de capacidades multimedia al ecosistema para añadir componentes interactivos a las aplicaciones.

Figura 3-2. Data/Context Management.

3.5.3 Interfaces to Network and Devices

Estos elementos ofrecen Control e información sobre la red IoT. Permiten la abstracción y virtualización de los recursos y funcionalidades de la red.

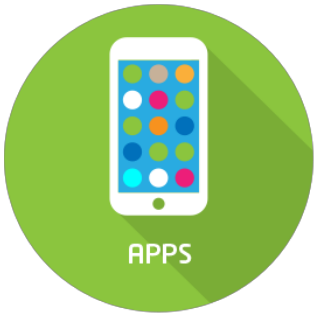


Figura 3-3. Interfaces to Network and Devices.

3.5.4 Advanced Web-Based User Interface

Es necesario en un ecosistema IoT, la inclusión de elementos que permitan la comunicación con plataformas web. Esta parte cuenta con numerosas interfaces para proveer información de manera genérica mediante APIs Web de alto nivel Estandarizadas. Estas APIs comprenden desde para Realidad Virtual, datos GIS, etc.

Figura 3-4. Advanced Web-Based User Interface

3.5.5 Security

La seguridad es un factor clave en cualquier proyecto tecnológico. La parte técnica "Security" se encarga de la gestión de herramientas que aumenten la seguridad del sistema mediante gestión de políticas de seguridad, gestión de usuarios y funciones de proxy.

Figura 3-5. Security.

3.5.6 Internet Of Things

Para la inserción de información hacia el ecosistema se usan los elementos de la parte "Internet of Things" la cual cuenta con elementos para proveer APIs para comunicaciones M2M (Maquina a Máquina); para el registro a la red de nuevos dispositivos y sensores; para procesar datos en tiempo real; o para actuar sobre los dispositivos o actuadores del ecosistema.

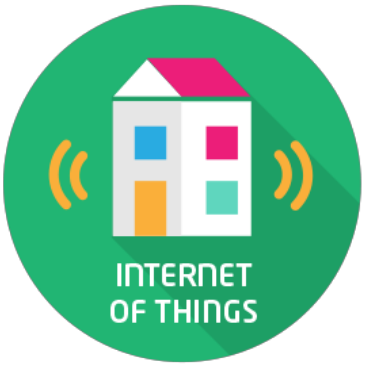


Figura 3-6. Internet Of Things.

3.5.7 Applications/Services and Data Delivery

Todos los datos que obtenemos del ecosistema deben de tener un consumidor final. Los elementos de esta parte técnica se encargan de proveer de servicios y APIs a los usuarios finales de los datos que fluyen dentro del ecosistema FiWare.

Figura 3-7. Applications/Services and Data Delivery.

3.6 NGSI

NGSI es una especificación propuesta por la OMA (Open Mobile Alliance) que se basa en una API Rest para comunicarse mediante peticiones HTTP de tipo GET y POST. Estas especificaciones están definidas en con los nombres NGSI-9 y NGSI-10 (FI-WARE NGSI Open RESTful API Specification (PRELIMINARY)). Sin embargo, el gestor de contexto FiWare NGSI no implementa todas las especificaciones de la OMA para no ser innecesariamente complejo. A pesar de eso, resuelva algunas ambigüedades en las especificaciones de la OMA y las extiende para una visión orientada a IoT.

El aspecto central del modelo de información de NGSI es el concepto de entidades. Las entidades son la representación virtual de toda clase de objeto físico del mundo real. Ejemplos como mesas, habitaciones o personas pueden ser entidades. Las entidades virtuales tienen un identificador y un tipo. Por ejemplo, una entidad virtual representa a una persona llamada “José” puede tener el identificador “José” y el tipo “persona”.

Cualquier información disponible de las entidades físicas es expresada en forma de atributos de las entidades virtuales. Los atributos también tienen un nombre y un tipo. Por ejemplo, la temperatura del cuerpo de José puede ser representado como un atributo con nombre “temperatura corporal” y el tipo “temperatura”. Los valores de estos atributos se definen en los contenedores de valores. Esta clase de contenedores no solo consisten en actual atributo-valor, si no que contienen una serie de metadatos. Los metadatos son los datos de los datos; en nuestro ejemplo de temperatura corporal estos metadatos pueden contener el tiempo de la medida, las unidades de esta, y otra información acerca del valor del atributo.

También está el concepto de dominios de atributos en la OMA NGSI 9/10. Un dominio de atributo agrupa de manera lógica a un conjunto de atributos. Por ejemplo, el dominio de atributos “estado_salud” puede estar

compuesto de los atributos “temperatura_corporal” y “presion_sanguinea”.

La estructura de datos usada para el intercambio de información sobre entidades son los elementos de contexto. Un elemento de contexto contiene información sobre múltiples atributos de una entidad. El dominio de estos atributos también puede ser especificado dentro del elemento de contexto; en este caso todos los valores de atributos incluidos pertenecen a ese dominio.

Un elemento de contexto contiene la siguiente información:

- Un id de entidad y tipo.
- Una lista de tripletas <nombre de atributo, tipo de atributo, valor de atributo> que contienen información sobre atributos de la entidad.
- (Opcional) El nombre del dominio del atributo.
- (Opcional) Una lista de tripletas <nombre de metadato, tipo de metadato, valor de metadato> que se aplica a todos los valores de atributos del dominio dato.

La OMA NGSI define dos interfaces para intercambiar información basada en el modelo. La interfaz NGSI-10 es usada para intercambiar información sobre las entidades y atributos. La interfaz NGSI-9 es usada para información de estado de entidades y atributos. En esta interfaz, en lugar de intercambiar valores de atributos, la información acerca de quien ha suministrado cierto atributo es intercambiada.

Las APIs NGSI solo soportan XML como formato de serialización de datos. XML es un lenguaje normalizado de marcado de datos similar a HTML.

La representación de los recursos es transmitida entre cliente y servidor usando el protocolo HTTP 1.1, definido por el IETF en la RFC-2616 [52].

3.6.1 FiWare NGSI-9

La interfaz NGSI-0 es una API Rest que funciona sobre HTTP. Su propósito es intercambiar información acerca de la disponibilidad de la información de contexto. Las tres principales interacciones son:

- Peticiones para descubrir nuevos dispositivos o agentes donde cierta información de contexto está disponible.
- Subscripciones para actualizaciones sobre información de disponibilidad de contextos (y sus correspondientes notificaciones).
- Registro de información de contexto como notificaciones de disponibilidad de cierta información de contexto.

El árbol de recursos de funcionalidades sigue dos aproximaciones. Por una parte, hay unos recursos para operaciones que soportan las respectivas funcionalidades mediante mensajes POST. Por otra parte, existes recursos adicionales que realizan funcionalidades de conveniencia entre dispositivos.

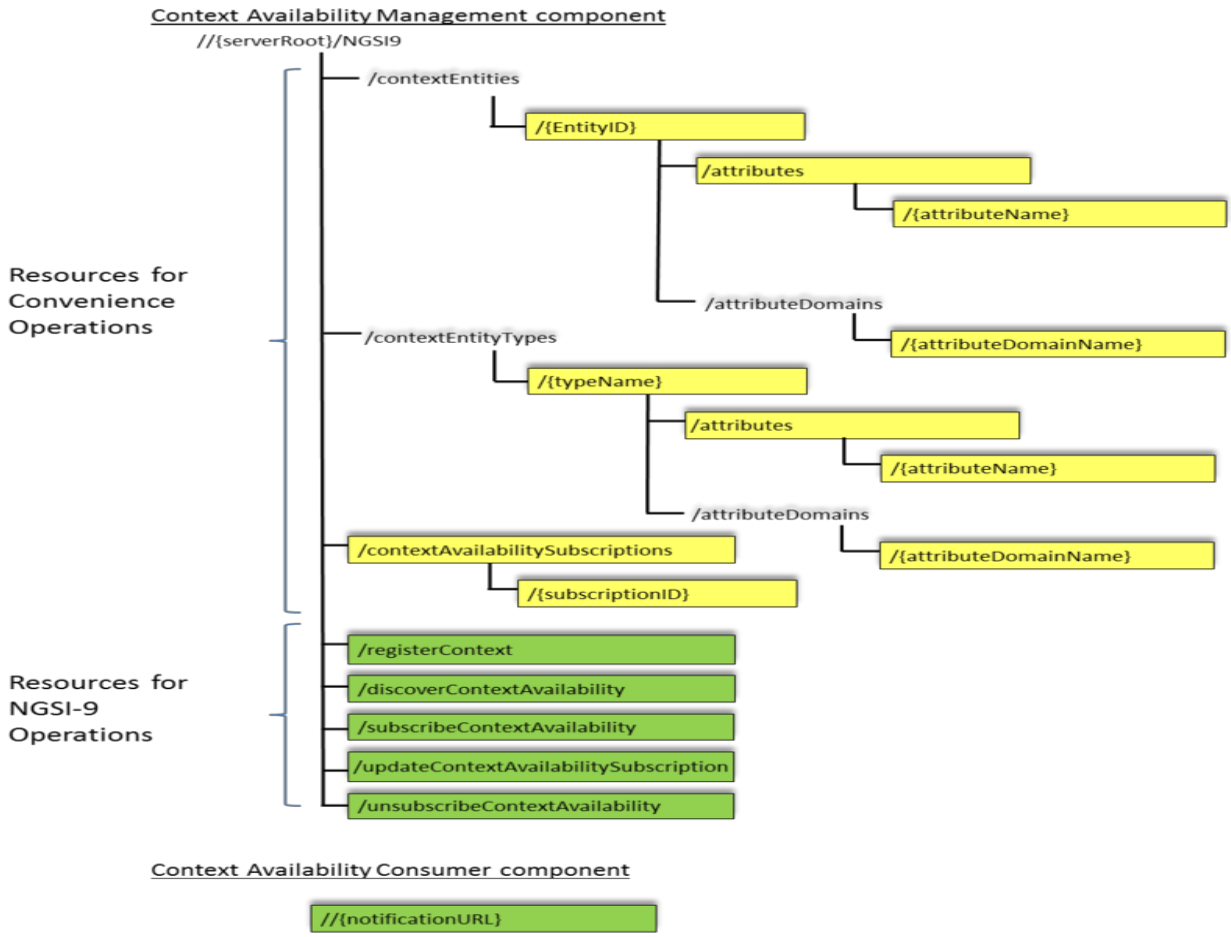


Figura 3-8. Árbol de Recursos NGSI-9.

3.6.2 FiWare NGSI-10

La interfaz NGSI-10 también se comunica mediante una API Rest a través de HTTP. Su propósito es intercambiar información de contexto. Las tres interacciones principales son:

- Peticiones de información de contexto
- Suscripciones de actualizaciones de información de contexto.
- Actualizaciones no solicitadas invocadas por el proveedor de contenido.

El árbol de recursos de funcionalidades sigue la misma arquitectura del árbol de NGSI-9.

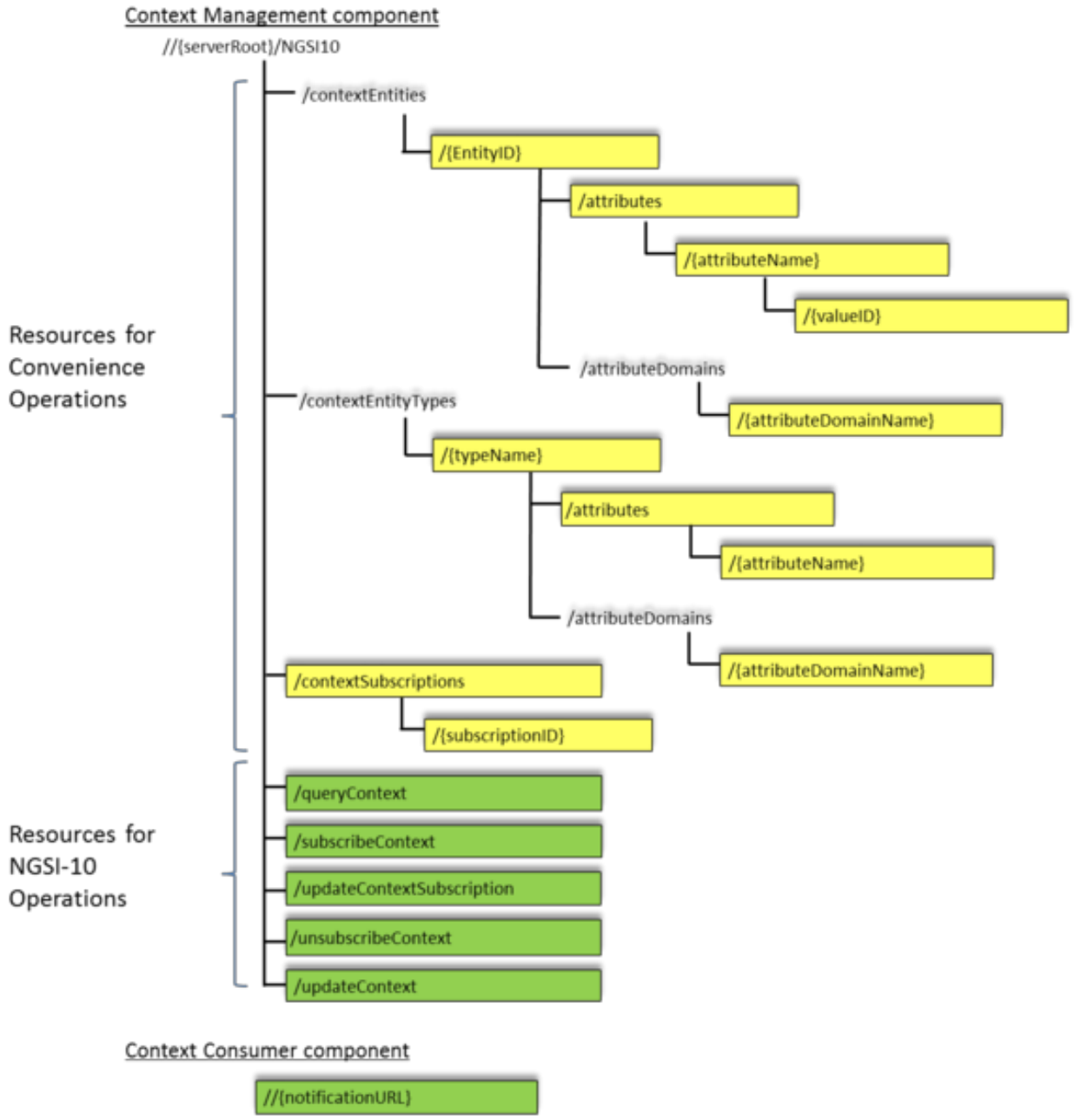


Figura 3-9. Árbol de Recursos NGSI-10

4 ARQUITECTURA DEL SISTEMA

Sólo cuando conoces cada detalle de la condición del terreno puedes maniobrar y luchar.

- Sun Tzu -

Cualquier ecosistema IoT debe de cumplir unos requerimientos que permita que dicho ecosistema sea escalable y distribuido, que los componentes que forman el ecosistema puedan interactuar entre ellos, y que todo esto se realice de manera eficiente y segura.

La implementación de todos estos requerimientos bajo el estandar FiWare es lo que lo hace idóneo para el desarrollo del ecosistema bajo esta tecnología.

A continuación, se exponen las características del ecosistema desplegado en el proyecto apoyándonos en FiWare como estandar.

4.1 Arquitectura General del Sistema

El esquema de la arquitectura se presenta en la Figura 4-1, donde de abajo hacia arriba en la arquitectura se pueden observar los distintos módulos, desde la obtención de las medidas de calidad, la transmisión de éstas a través de la red MAN, su agregación e integración gracias haciendo uso de FiWare, el post-procesado de la información con distintas técnicas de análisis Big Data y su posterior representación en la plataforma web.

Como se ha comentado, dicha arquitectura se fundamenta en el Context Broker Orion, por el cual circula toda la información del sistema y orquesta la interrelación con los distintos componentes de la arquitectura.

Partiendo desde la base de la arquitectura, aquí se encuentran los productores de la información que son la fuente de datos de toda plataforma. Estos productores son los distintos sensores, actuadores, APIs, entre otras, que componen el sistema. Algunos de estos productores, necesitan la ayuda de determinados agentes que se encargan de la conversión de la información generada, a un formato legible para el Context Broker el cual es NGSI, que se aglutinan en el IoT Agent.

Junto al Context Broker, se sitúan los diferentes simuladores, que se utilizan en los entornos de pruebas para realizar test de funcionalidad, test de esfuerzos del sistema, test de comprobaciones de la capacidad de transacción del sistema, etc. Además, permite comprobar la escalabilidad del sistema y el comportamiento ante nuevas fuentes de información conectadas. Por último, destacar que los simuladores son fundamentales para permitir evaluar el comportamiento del sistema ante los datos de productores de información simulados, que en el futuro se sustituyen por productores reales.

Por encima de Orion, se encuentran los diferentes consumidores de la información producida por el sistema. Esta información, al igual que en la parte de producción, pasara por un proceso de conversión al formato estándar de las diferentes bases de datos que pueden almacenar estos datos. Este proceso de conversión será realizado por lo que se denomina conector, el cual puede variar la implementación en función de la tecnología usada por la base de datos. Además, existen componentes específicos que se encargan de la publicación de datos abiertos y del procesamiento en tiempo real y procesamiento de grandes flujos de datos.

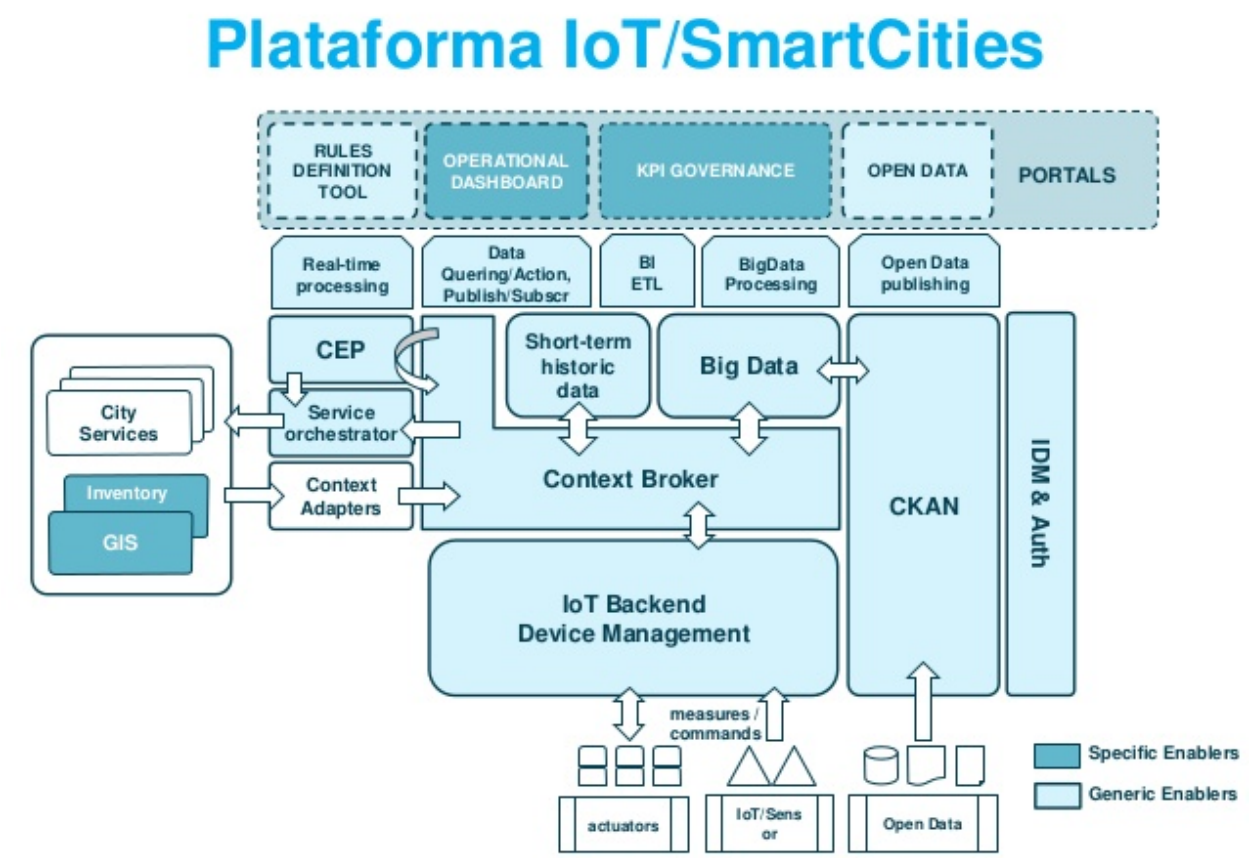


Figura 4-1. Arquitectura Plataforma Fiware

4.2 Orion Context Broker

El principal motor de FiWare es Orion Context Broker. Este es el encargado de intermediar entre los productores (publicadores) y consumidores (subscriptores). Orion Context Broker va de la mano de una base datos MongoDB la cual guarda toda la información del broker. Por esto motivo se dice que el broker no tiene memoria interna. Es la base de datos MongoDB la que guarda toda información relativa a los publicadores, es decir, toda la información relacionada a sus entidades, atributos, tiempo de actualización… y de los suscriptores, los cuales almacenan información de las entidades suscritas, los atributos, cuando mandar a esta la notificación, etc. La importancia de esta base de datos reside en que, si por alguna razón el broker tuviese algún problema, toda la información persistirá en esta MongoDB.

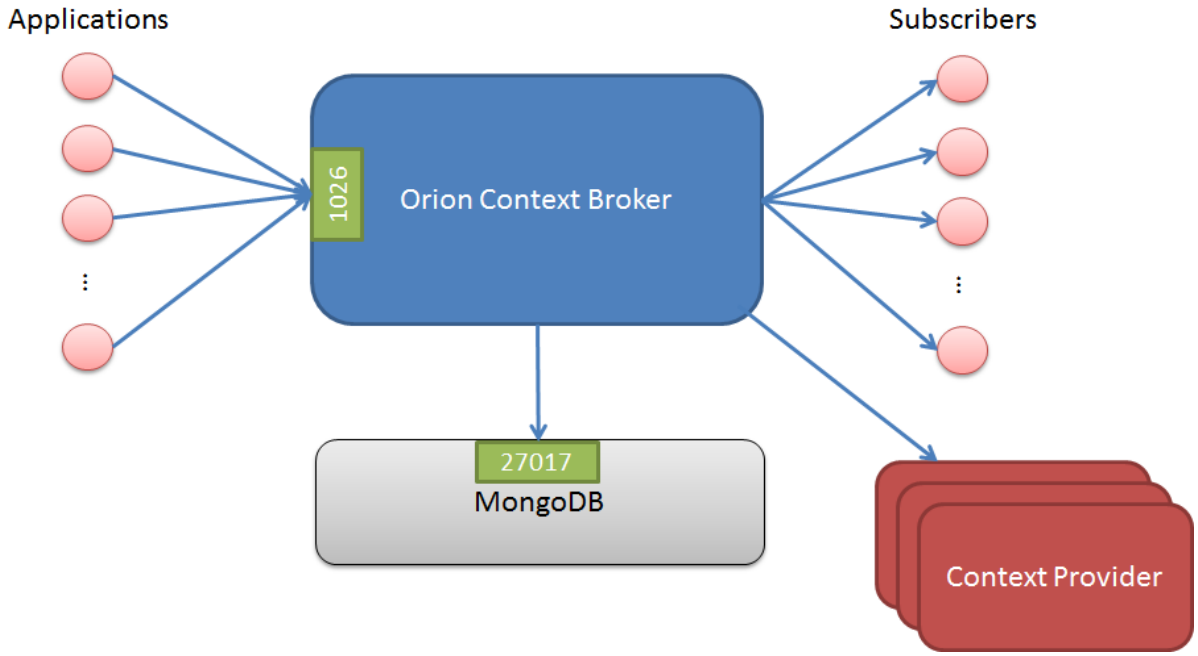


Figura 4-2. Arquitectura Orion Context Broker.

FiWare Orion está desarrollado en C++. Gracias a Orion Context Broker se puede manejar al completo el ciclo de vida de la información de contexto incluidos las actualizaciones, registros y suscripciones. A través de Orion context Broker, se podrá crear elementos de contexto nuevos y administrarlos a través de actualizaciones y peticiones. Además, se podrá suscribir a determinada información para que sea mandada cuando se cumplan ciertas condiciones o la información de los elementos cambien.

En la Fig 4-2 podemos ver como el Context Broker es el epicentro de toda acción donde recibe información a través del puerto estándar 1026. Además, podemos ver la interacción de este con la base de datos MongoDB la cual se encarga de la persistencia de la información. El Context Broker se comunica con la base de datos MongoDB a través del puerto estándar de mongo 27017.

También se puede observar que ambas entidades, Orion y base de datos MongoDB se pueden ubicar en distintas maquinas. Con esto, se puede dotar fácilmente al sistema con redundancia, teniendo varios replicasets de la base de datos en diferentes máquinas, mejorando la fiabilidad global del sistema.

De la misma manera que se pueden instanciar varias bases de datos en diferentes maquinas, se puede realizar una instanciación del bróker en varias máquinas para que, en caso de fallo, no se pierda el servicio de este durante mucho tiempo. Con la redundancia como punto clave en el despliegue de estos servicios, podemos hablar de fiabilidad del sistema. Para el aumento de dicha fiabilidad del sistema, se realiza una instanciación de varios Context Broker formando una confederación. Dicha confederación puede ser de dos tipos: push y pull. En las confederaciones de tipo push, uno de los context broker notifica al resto de las modificaciones realizadas por los productores de información. En cambio, en las confederaciones de tipo pull, el context broker receptor de la información, reenvía la misma información recibida al resto de instancias [53].

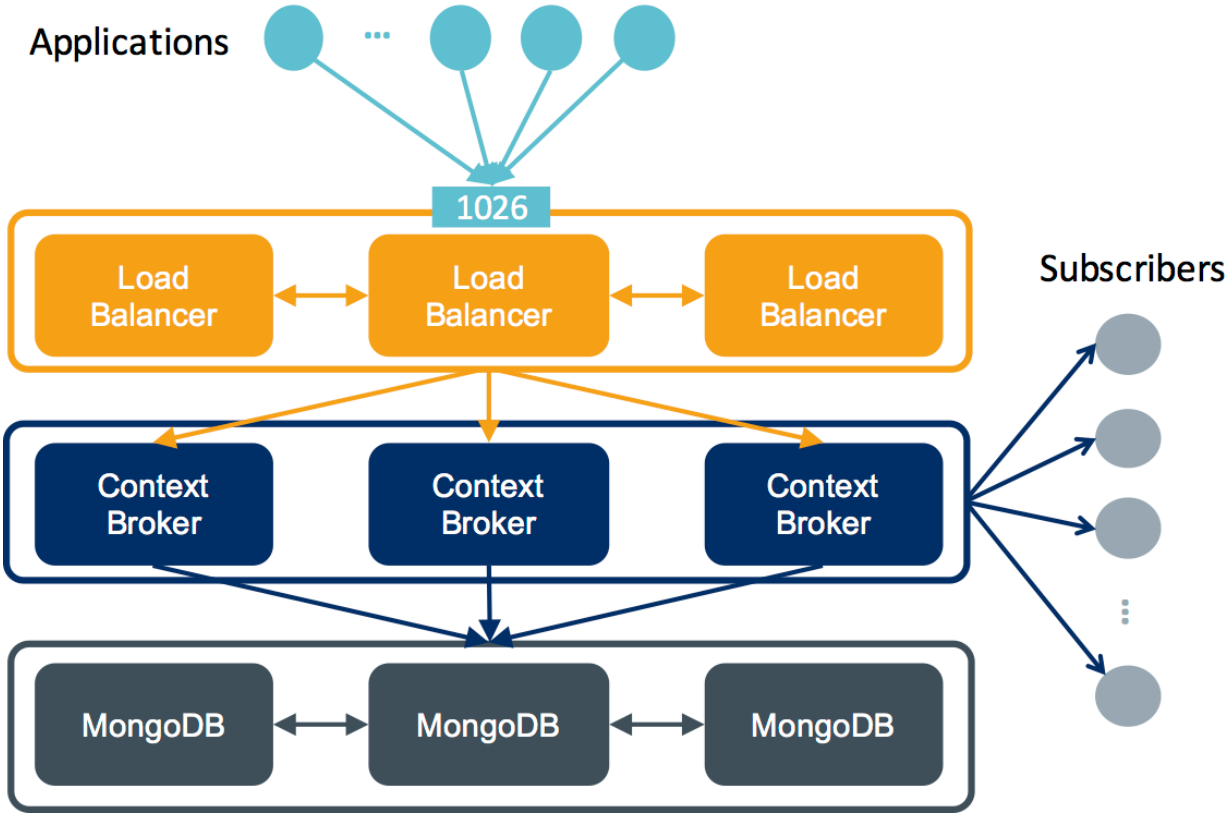


Figura 4-3. Redundancia en Orion [53].

En cuanto al context broker, su dinámica es sencilla. Este interactúa fundamentalmente con tres tipos de primitivas o mensajes: UPDATE, SUBSCRIBE y NOTIFY.

El Context Broker, recibe mensajes UPDATE de los IoT Agents los cuales están registrados en dicho broker. Los IoT Agents, son los dispositivos que actúan como intermediarios entre los sensores o dispositivos productores del sistema, y el broker. Se profundizará en estos agentes más adelante.

Además de los mensajes de tipo UPDATE, el context broker recibe mensajes de suscripción (SUBSCRIBE) de las aplicaciones consumidoras los cuales sirven para poder obtener información relativa a las entidades que queramos monitorizar.

Los mensajes de tipo NOTIFY son los que se encargan de enviar la información del bróker a las aplicaciones consumidoras que demandan la información.

En cuanto a los mensajes UPDATE, estos son enviados desde IoT Agent al broker con la información sobre los valores de los atributos de las entidades.

Estos son las tres primitivas que usa el Context Broker para llevar a cabo sus funciones. Otras características de Orion Context Broker que son útiles es la gestión de tokens de autenticación para aportar seguridad y que solo puedan acceder los consumidores que cuenten con uno de estos tokens.

4.3 Dispositivos Productores de Información

Los dispositivos productores de información son aquellos que proveen los datos al sistema. El IoT está formada por la red de este tipo de dispositivos físicos que son capaces de intercambiar datos. Cada dispositivo está formado por un microcontrolador embebido y un software el cual es capaz de actuar como sensor o actuador. Los sensores son capaces de enviar información del estado de ciertos elementos alrededor de él. Los actuadores son responsables de alterar el estado del sistema tras ciertas señales de control recibidas por estos.

Cada dispositivo es identificado unívocamente a través del sistema de computación embebido, pero también es capaz de identificarse perteneciendo a una existente infraestructura de internet o red de dispositivos.

Ya que FiWare es un sistema para administrar información de contexto. Para una solución inteligente del internet de las cosas, el contexto será el conjunto de elementos relacionados entre sí. Cada elemento IoT, será

representado como una única entidad dentro del contexto.

Los dispositivos IoT pueden ser desde dispositivos simples hasta otros bastante complejos. Algunos ejemplos de dispositivos IoT pueden ser:

- Puerta inteligente: Una puerta electrónica que puede recibir comandos para ser cerrada o abierta remotamente. También puede reportar su estado actual.
- Una campana: Puede recibir comandos para activarla o sonar durante un determinado intervalo de tiempo.
- Un sensor de movimiento: Puede enviar el número de personas que han pasado desde el ultimo reset del dispositivo.

Como se puede ver, la campana es un ejemplo de un actuador puro ya que solo actúa a los comandos que recibe. Por otra parte, el sensor de movimiento es un ejemplo de sensor puro, ya que solo envía información de su estado. La puerta inteligente es capaz de responder ante comandos recibidos y de enviar información del estado.

4.3.1 UltraLight 2.0

UltraLight 2.0 es un protocolo de texto ligero diseñado para dispositivos con limitado ancho de banda y escasa memoria. El cuerpo de las medidas enviadas es una lista de clave-valor separados por el carácter '|'.

Un ejemplo de cuerpo seria: 't|5|k|as'. Este contiene dos atributos, uno con nombre 't' y valor '15' y otro con nombre 'k' y valor 'as'. Todos los valores de este protocolo son categorizados como cadenas de texto.

Ultralight 2.0 define cuerpos para describir medidas y comandos intercambiados entre dispositivo y servidor, pero no especifica un solo protocolo de transporte. Este estándar, define protocolos de transporte como HTTP, MQTT o AMQP. Estos mensajes serán transportados hasta el agente IoT, el encargado de la conversión y tipado de este tipo de comunicación a una legible por el Context Broker, la cual es NGSI. El agente IoT será descrito en siguientes apartados.

4.3.2 Tráfico Descendente (Comandos)

Los mensajes HTTP generados por el Context Broker y enviados a los dispositivos IoT son denominados tráficos descendente. Este tipo de trafico consiste en comandos que provocan que le actuador cambie su estado mediante una acción. Por ejemplo, un comando para para modificar el estado de la puerta a 'abierto', abriría la puerta en la vida real. Esta acción, puedo influir en las lecturas de otros sensores cercanos.

La configuración descendente entre un agente IoT con un dispositivo IoT es conocida como aprovisionamiento. Esta se asegura de que el agente tiene suficiente información para ser capaz de contactar con el dispositivo.

El cuerpo del comando Ultralight tiene el siguiente formato:

<nombre_dispositivo>@<comando>|<parametro|<parametro>

Donde el <nombre_dispositivo> es el id de la entidad en el context broker, <comando> es uno de los comandos soportados por el dispositivo y cualquier información adicional requerida se escribe a continuación.

4.3.3 Tráfico Ascendente (Medidas)

Las peticiones generadas por un dispositivo IoT y enviadas hacia el Context Broker son conocidas como trafico ascendente. Este tipo de trafico consiste en medidas hechas por el sensor y forman parte del estado del mismo en un contexto determinado. Una suscripción puede hacerse para informar de nuevas medidas y provocar otras acciones.

Un dispositivo puede enviar nuevas medidas al agente IoT usando una petición HTTP GET hacia el mismo. En la cabecera del mensaje viajaran los siguientes parámetros:

- I: ID del dispositivo.
- K: API Key del servicio del dispositivo.
- T: Instante de la medida.
- D: Datos. Cuerpo del mensaje Ultralight 2.0.

También puede usarse HTTP POST para el envío de medidas. En este caso, el parámetro 'd' no será necesario, el atributo-valor de la medida será pasado en el cuerpo de la petición.

4.4 Agente IoT

Un agente IoT es un componente que permite a un grupo de dispositivos enviar sus datos y ser administrado por el Context Broker usando sus propios protocolos nativos. Los agentes IoT deben de encargarse de aspectos de seguridad de la plataforma FIWARE como autenticación y autorización.

El Broker usa solo NGSI para todas sus interacciones con el resto de dispositivos. Cada agente IoT ofrece una interfaz NGSI usado por el context broker y otra interfaz del protocolo nativo de los dispositivos conectados al agente.

En la realidad, esto ofrece una interfaz estándar a todas las interacciones IoT al nivel de gestión de información de contexto. Cada grupo de dispositivos IoT son capaces de usar sus protocolos propietarios mientras que el agente hace la función de apoderado para la conversión del tipado NGSI.

En definitiva, este componente simplifica la administración e integración de dispositivos. Este recoge información de protocolos heterogéneos y convierte estos en el estándar NGSI, permitiendo también mandar comandos a los dispositivos.

La plataforma actualmente soporta varios protocolos IoT con una arquitectura modular. Entre los ejemplos, se incluyen los siguientes:

- Agente JSON [54]: Actúa de agente entre mensajes HTTP/MQTT, con un cuerpo con formato JSON, y NGSI.
- Agente LWM2M [55]: Actúa como puente entre el protocolo Lightweight M2M y NGSI.
- Agente UL [56] : Puente entre un mensaje HTTP/MQTT, con un cuerpo UltraLight2.0, y NGSI.
- Agente LoRaWAN [57]: Puente entre el protocolo LoRaWAN [58]y NGSI.

Si el dispositivo usa distintos protocolos a los suministrados, será necesario una traducción entre la plataforma especifica del dispositivo y el modelo estándar de la plataforma (NGSI). Para ello, FiWare cuenta con varios componentes para el desarrollo de nuevos agentes.

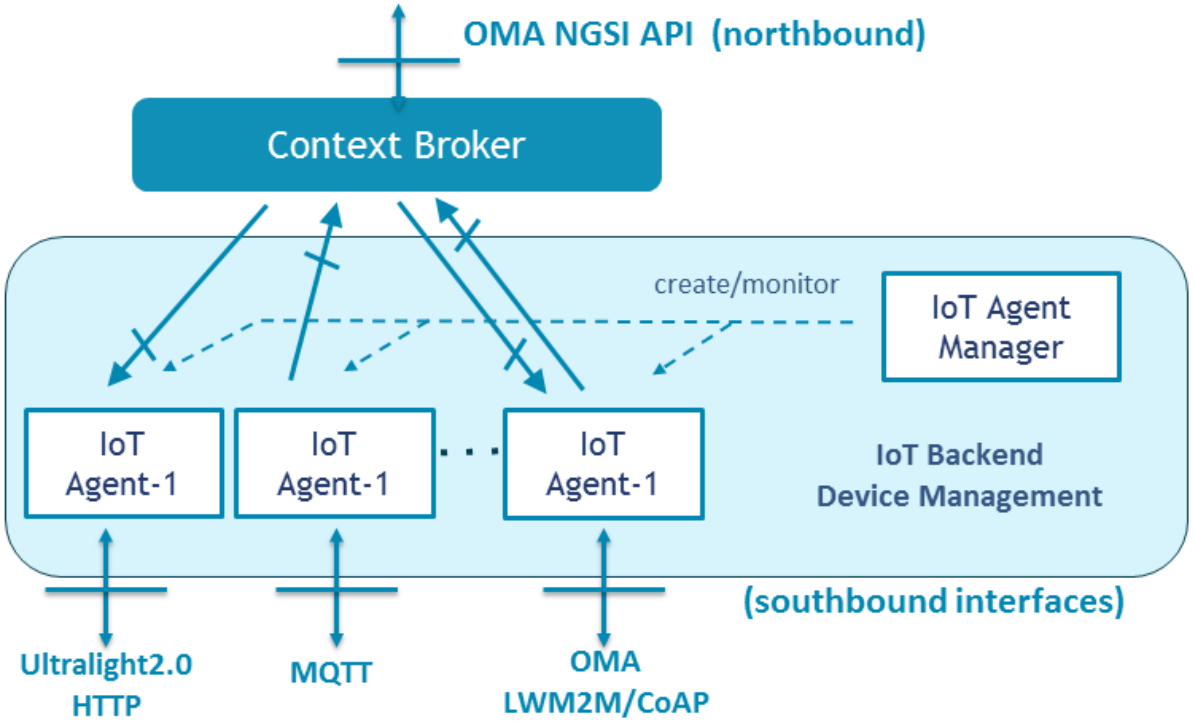


Figura 4-4. Agente IoT.

4.5 Cygnus

La plataforma FiWare ha sido diseñado para manejar información del contexto actual, es decir está diseñado para mantener los datos de las entidades en el momento dato.

El contexto trata solo del estado actual del sistema, no es responsabilidad de ninguno de los componentes del sistema de informar del histórico de estados del sistema. El contexto se basa en la última medida que cada sensor ha enviado al context broker.

Para hacer esto, necesitamos extender la arquitectura existente para que la información histórica de estados sea almacenada en una base de datos.

La persistencia de los datos de contexto es útil para análisis Big Data ya que puede ser usada para detectar tendencias. Sin embargo, en cada tipo de solución IoT, las entidades y atributos necesitaran ser muestreados en diferentes intervalos de tiempo.

Los requerimientos para usar información de contexto han variado entre distintas aplicaciones, es decir, no ha habido un estándar para la persistencia de la información ya que cada situación era única y una única solución no encajaba con las demás. Por ello para la persistencia de la información de contexto, se diseñó un componente altamente configurable llamado Cygnus.

Cygnus, como parte de una plataforma de código abierto, es una tecnología pensado para la implementación de distintas bases de datos para la persistencia de la información en Fiware. La base de datos que se elige puede varias en función de las necesidades de negocio de un proyecto determinado.

Sin embargo, hay un coste para ofrecer esta flexibilidad. Cada parte del sistema necesita estar configurada de forma independiente y las notificaciones necesitaran estar configuradas para solo escribir en la base de datos la mínima cantidad de datos necesaria.

Cygnus es un conector que permite la persistencia de información de ciertas fuentes de datos en una base de datos de terceros, creando una vista histórica de dicha información.

Internamente, Cygnus está basado en Apache Flume [59], una tecnología destinada a facilitar la agregación de

grandes cantidades de dato a través de agentes de persistencia. Un agente está formado por un recolector de información de la fuente, un canal donde la fuente pone los datos para ser transformados en un evento Flume, y un sumidero, donde el evento Flume se deposita en una base de datos de terceros.

Cygnus está diseñada para recolectar determinadas fuentes de datos. Actualmente, Cygnus es capaz de permitir la persistencia de información de las siguientes fuentes de datos en las siguientes bases de datos:

- Información de datos de contexto NGSI en:
    - Sistemas distribuidos HDFS [60]
    - Base de datos relacionales MySQL [61].
    - Plataforma de datos abiertos CKAN [62].
    - Base de datos no relacionales como MongoDB [63].
    - Base de datos de carácter temporal STH Comet [64].
    - El broker Kafka [65].
    - Base de datos basada en cloud DynamoDB [66].
    - Base de datos relacional PostgreSQL [67].
    - Base de datos especializada en datos geo Carto [68].
- Datos provenientes de Twitter en:
    - Sistemas distribuidos de ficheros HDFS.

4.5.1 Arquitectura de Cygnus

Cygnus usa agentes Flume para su funcionamiento. La arquitectura de los agentes de Cygnus es la misma que los agentes de Flume.

Según está descrito en flume.apache.org:

Un evento es una unidad de datos que fluye a través de un agente de Flume. El evento fluye desde el origen al canal al receptor, y se representa mediante una implementación de la interfaz del evento. Un evento lleva una carga útil (matriz de bytes) que está acompañada por un conjunto opcional de encabezados (atributos de cadena). Un agente de Flume es un proceso (JVM) que aloja los componentes que permiten que los Eventos fluyan desde una fuente externa a un destino externo.

Una Fuente consume Eventos que tienen un formato específico, y esos Eventos son entregados a la Fuente por una fuente externa como un servidor web. Por ejemplo, un AvroSource se puede usar para recibir Avro Events de los clientes o de otros agentes de Flume en el flujo. Cuando una Fuente recibe un Evento, lo almacena en uno o más Canales. El Canal es una tienda pasiva que contiene el Evento hasta que un Receptor consume ese Evento. Un tipo de canal disponible en Flume es FileChannel, que utiliza el sistema de archivos local como su almacén de respaldo. Un sumidero es responsable de eliminar un evento del canal y colocarlo en un repositorio externo como HDFS (en el caso de un HDFSEventSink) o enviarlo a la Fuente en el siguiente salto del flujo. La Fuente y el Fregadero dentro del agente dado se ejecutan de forma asíncrona con los Eventos organizados en el Canal.

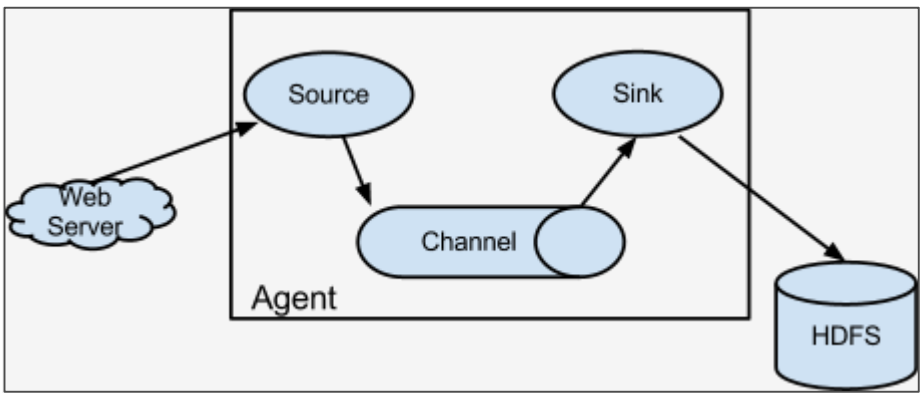


Figura 4-5. Arquitectura Apache Flume [69].

La manera más simple de usar Cygnus es adoptar el flujo fuente-canal-sumidero descrito en la documentación de Apache Flume. Puede haber tantos flujos básicos como elementos de persistencia de información haya.

Por cada uno de estos flujos, una fuente HTTP tiene que ser usada. La manera en la que esta fuente nativa procesa las notificaciones del Context Broker es mediante una manejador de eventos REST. A pesar de esto, esta aproximación requiere que cada fuente reciba sus propias notificaciones. Esto no es un problema si la arquitectura define claramente que flujo debe de acabar en una base de datos Carto o en una base de datos MongoDB. Pero si un evento debe de ser almacenado en dos bases de datos a la misma vez, la arquitectura debe de ser cambiada para que los eventos sean replicados a cada canal conectado a la fuente.

4.5.2 Alta Disponibilidad en Cygnus

Cygnus no implementa ningún mecanismo de alta disponibilidad en sí. De todos modos, dotar de alta disponibilidad es tan fácil como tener dos instancias del software y contar con un balanceador de carga de red entre ellos y la fuente de datos.

El balanceador de carga por si solo es un punto crítico de fallos. Para esto hay soluciones como redundancia de balanceadores y otras soluciones de prevención de fallos mediante software específico.

4.6 Consumidores de Información

Se entiende como consumidores de información a todos aquellos componentes de la arquitectura, capaces de hacer de la información de contexto fácil de procesar, de analizar o visualizar para así implementar el comportamiento inteligente en cualquier aplicación.

Existen varios componentes de la arquitectura de FiWare que se encargan de dicho procesamiento y análisis de información de contexto:

- Wirecloud [70]: Componente que se provee de una potente plataforma web para ofrecer gráficas y tablas de cualquier tipo de información de contexto altamente configurable por los usuarios finales.
- Knowage [71]: Knowage ofrece una potente plataforma de inteligencia de negocio que permite realizar analíticas de negocio a través de sistemas Big Data y fuentes tradicionales de recolección de información.
- Kurento [72]: Este componente permite el procesamiento en tiempo real de flujos de datos de video ofreciendo la transformación de cámaras de video en sensores, así como la incorporación de funciones avanzadas de integración de comunicaciones audiovisuales, de realidad aumentada, etc.
- Cosmos [73]: Cosmos ofrece un análisis Big Data de manera sencilla a través de la integración con las plataformas Big Data más populares en la actualidad.
- FogFlow [74]: Ofrece un framework de ejecución distribuida para apoyar el procesamiento dinamico de flujos mediante infraestructuras Cloud.

Además de todos los componentes que forman parte de la arquitectura desarrollada por FiWare, podemos tener en cuenta cualquier sistema de base datos, tanto relacional como no relacional, como elementos consumidores de la información ya que en ellos se puede almacenar la información de contexto extraída de los consumidores del mismo.

Como se ha comentado en el apartado anterior referente a Cygnus, este puede actuar como puente entre Orion Context Broker y las bases de datos que este soporta.

Una vez que la información de contexto sea almacenada en cualquier base de datos, esta puede ser usada por motores Big Data y aplicaciones Web que sean capaces de extraer inteligencia del uso de dicha información de contexto.

Todo tipo de motor capaz de extraer inteligencia de la información de contexto, es denominado sistema Backend. Dichos sistemas puedes ser desarrollados en multitud de lenguajes de programación y son responsables de la generación de informes y gráficas para ayudar a la interpretación del contexto. Además,

dichas aplicaciones pueden ser capaces de actuar frente a diferentes aspectos del contexto, generando alertas y actuando sobre ellas, o capaces de comunicarse con el usuario mediante diferentes formas de acceder a este como correo electrónico, mensajes de texto, etc.

5 APLICACIÓN: CASO DE USO

Pon tu mano de hierro en un guante de terciopelo.

- Napoleon Bonaparte -

En este apartado se va a describir los elementos que han sido desarrollados para el despliegue de una solución inteligente para la gestión y administración del consumo y distribución de agua. Esta solución recoge el nombre de AQUASIG y en este trabajo fin de grado se describen los trabajos desarrollados por el autor del mismo para el despliegue de esta.

5.1 Introducción

AQUASIG se beneficia enormemente del uso de la plataforma FiWare, que es la base para garantizar una interoperabilidad de datos y tareas de integración desde los datos en tiempo real generados por el sistema de adquisición de la medida hasta las bases de datos (Big Data y con componente geográfica) que son la base para que la plataforma AQUASIG provea de servicios inteligentes dirigidos a distribuidoras y consumidores de agua a nivel urbano.

Orion se encarga de recolectar toda la información de los productores, de las fuentes de información, en nuestro caso, los dispositivos de medida de calidad del agua, de consumo y de presión, y de transmitirla a los subscriptores que lo necesiten, en nuestro caso, las bases de datos Big Data.

El esquema de la arquitectura de la solución diseñada para AQUASIG se presenta en la Figura 5-1, donde de abajo hacia arriba en la arquitectura se pueden observar los distintos módulos, desde la obtención de las medidas de calidad, la transmisión de éstas a través de la red MAN, su agregación e integración gracias haciendo uso de FiWare, el post-procesado de la información con distintas técnicas de análisis Big Data y su posterior representación en la plataforma web

Como se ha comentado, dicha arquitectura se fundamenta en el Context Broker Orion, por el cual circula toda la información del sistema y orquesta la interrelación con los distintos componentes de la arquitectura.

Partiendo desde la base de la arquitectura, aquí se encuentran los productores de la información que son la fuente de datos de la plataforma. Estos productores son los distintos sensores (caudal, presión, calidad, etc.), APIs, entre otras, que componen el sistema. Algunos de estos productores, necesitan la ayuda de determinados agentes que se encargan de la conversión de la información generada, a un tapado legible para el Context Broker, que se aglutinan en el IoT Agent Manager.

.

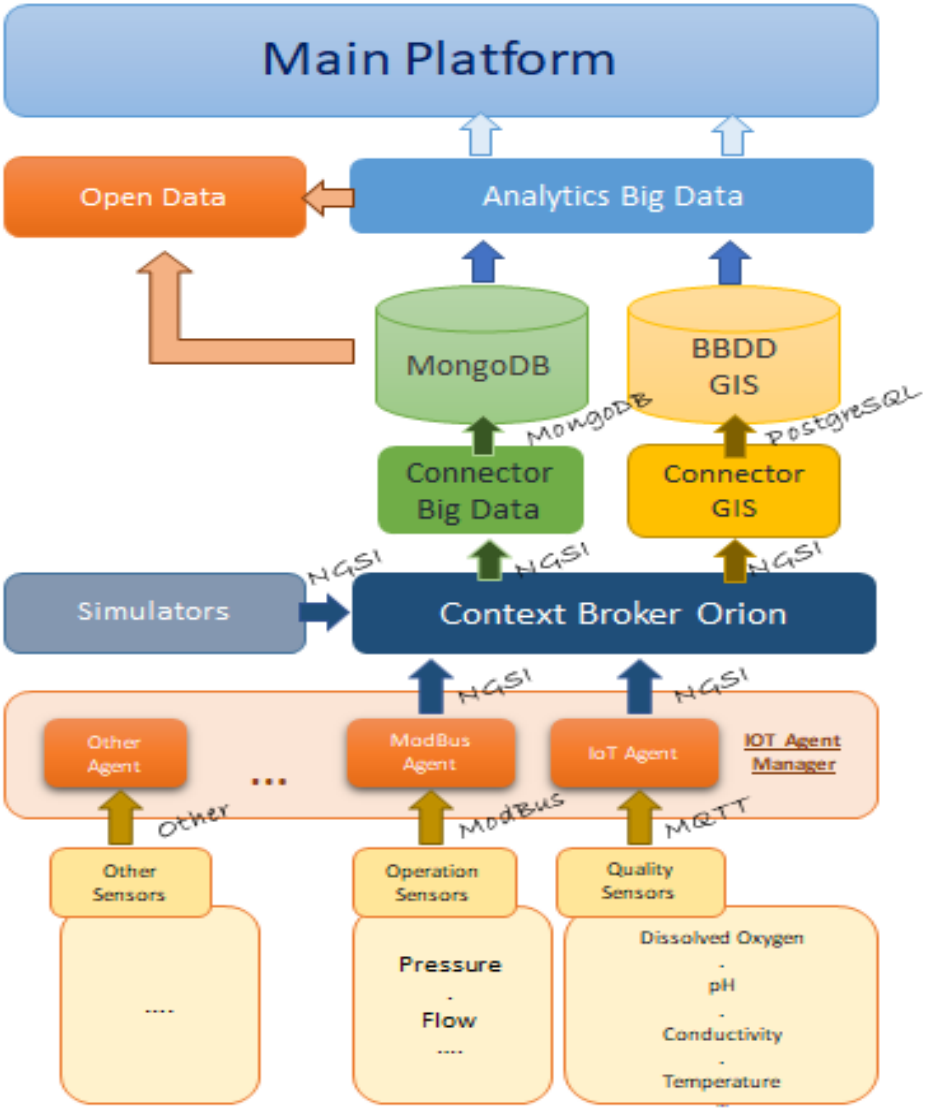


Figura 5-1. Arquitectura de la solución.

Junto al Context Broker, se sitúan los diferentes simuladores, que se utilizan en los entornos de pruebas para realizar test de funcionalidad, test de esfuerzos del sistema, test de comprobaciones de la capacidad de transacción del sistema, etc. Además, permite comprobar la escalabilidad del sistema y el comportamiento ante nuevas fuentes de información conectadas a la plataforma. Por último, destacar que los simuladores son fundamentales para permitir evaluar el comportamiento del sistema ante los datos de sensores simulados, que en el futuro se sustituyen por sensores reales.

Por encima de Orion, se encuentran los diferentes consumidores de la información producida por el sistema. Esta información, al igual que en la parte de producción, pasara por un proceso de conversión al formato estándar de las diferentes bases de datos que pueden almacenar estos datos. Este proceso de conversión será realizado por lo que se denomina conector, el cual puede variar la implementación en función de la tecnología usada por la base de datos.

Una vez se tengan la información en las bases de datos correspondiente, se puede acceder a estas para su análisis y posterior aportación a los módulos de generación de servicios dirigidos a las distribuidoras y dirigidos a los consumidores.

Para el despliegue de los componentes de la arquitectura FiWare usados en este proyecto, se usará la tecnología Docker [75]. Docker es una tecnología de contenedores software la cual permite aislar a los diferentes componentes en sus ecosistemas respectivos.

La herramienta de Docker que usaremos se llama Docker Compose. Docker Compose fue creada para definir y desplegar aplicaciones Docker. Se usa un fichero de configuración donde se definen los servicios requeridos para la aplicación. Esto quiere decir que con un comando podemos desplegar todos los contenedores.

5.2 Productores de Información

En primer lugar, se analizan los distintos tipos de datos que debe ser capaz de procesar la plataforma, con las particularidades de cada uno, para obtener una información homogénea, fiable y completa a partir de todos esos datos, con el fin de almacenar en la base de datos dicha información para ser explotada por los servicios de la plataforma.

El total de información que alimentará a la base de datos proviene del sistema de adquisición de datos, donde fundamentalmente hay tres grandes bloques de fuentes de datos que son:

- Sensores de calidad de agua: Dichos sensores, estarán ubicados por sectores en función de la caracterización de la red de la distribuidora de agua.
- Sensores de control de presión de tuberías de agua: Estos sensores se encuentran en troncales de la distribuidora para permitir identificar ciertas fugas de agua mediante los mismos.
- Sensores de consumo de agua: Cada vivienda contara con un sensor de consumo que monitorizara en tiempo real el consumo de cada cliente.

En los sub-apartados siguientes se describe la información obtenida por cada una de las fuentes de sensorizacion del proyecto, destacando aquellos procedimientos de desarrollo de sensores y pre-procesamiento previos a la incorporación al sistema de procesado de datos de AQUASIG que se describe en los siguientes apartados.

5.2.1 Interfaces Adquisición de la medida

Los equipos terminales o end-device, como su mismo nombre implica, son los dispositivos de final de red, quienes están directamente integrados con los sensores instalados, sean del tipo que sean, es decir, se ha diseñado y desarrollado una interfaz de medida para equipos terminales o end-device que sea compatible con diversos tipos de sensores, como son los de calidad, consumo de agua, presión o consumo eléctrico. Estos equipos terminales están formados por los propios sensores, un componente de adquisición de medida y otro componente de comunicación inalámbrico el cual se encarga de convertir las medidas ofrecidas por los sensores en magnitudes y comandos aptos para su envío a través de la red inalámbrica implementada en el ecosistema IoT.

El equipo terminal implementado debe ser capaz de conectarse con la red inalámbrica desplegada y además extraer la información del sensor o dispositivo al que se conecte, por lo que debe ser compatible con las diferentes interfaces que dispongan los equipos desplegados en la red de distribución de aguas.

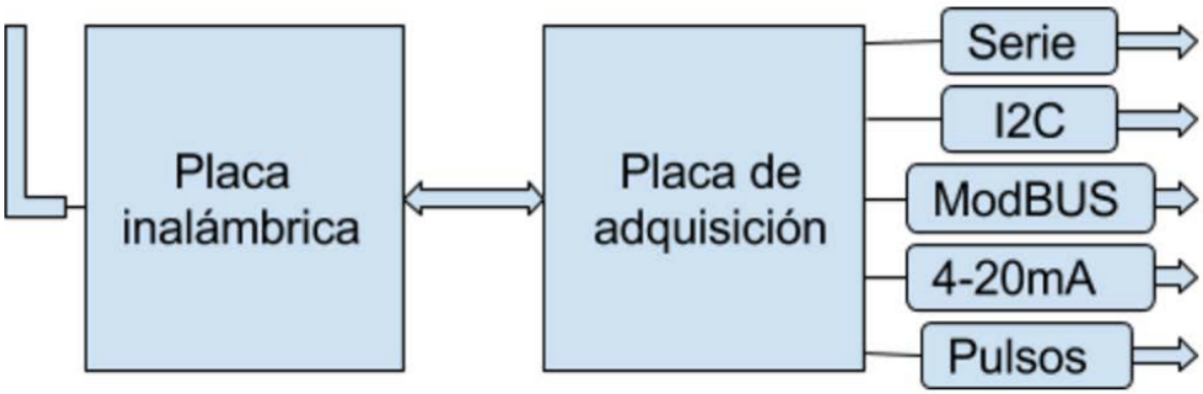


Figura 5-2. Modelo Genérico Interfaz de Medida

En la siguiente figura se puede ver el diagrama de bloques que compone a los End Devices:

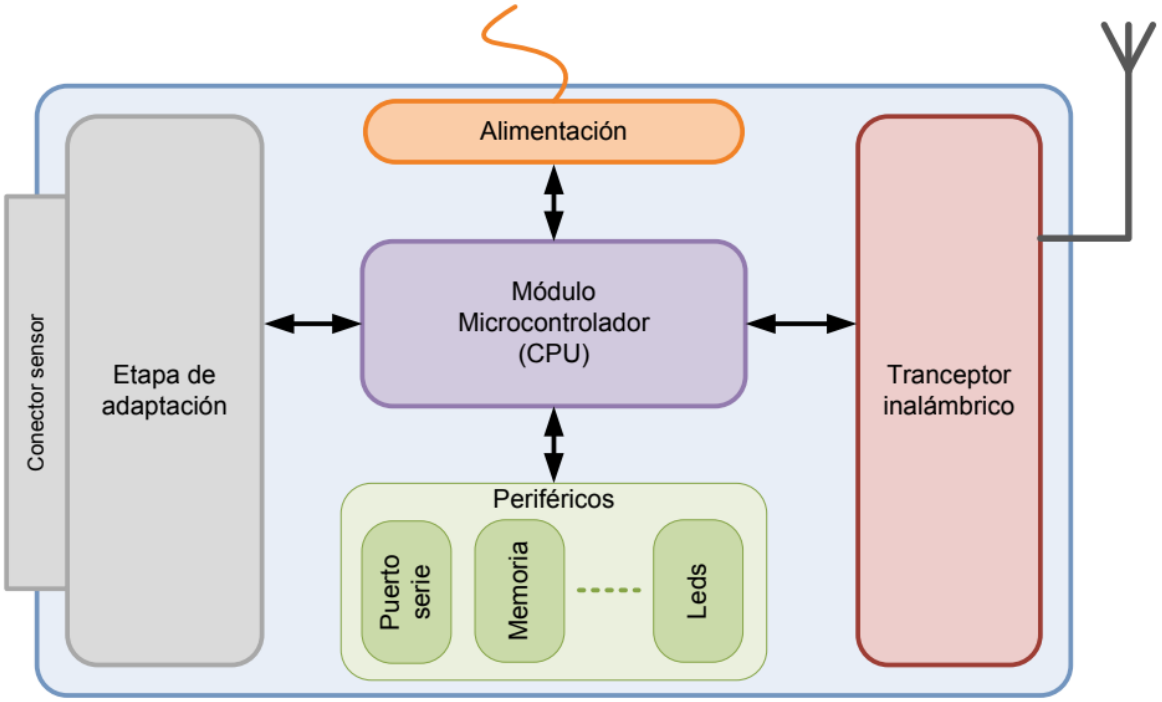


Figura 5-3. Arquitectura Dispositivo Final.

La mayoría de los bloques son comunes a todos los dispositivos que se encuentran en el proyecto, tal y como se ha comentado con anterioridad. Estos bloques son: Módulo microcontrolador, etapa de alimentación y etapa de periféricos.

En primer lugar, el Módulo Microcontrolador, también llamado CPU, es el encargado de procesar la información extraída de los sensores y transmitirla a través del transceptor inalámbrico.

La Etapa de Alimentación del End Device (ED), es la encargada de distribuir la corriente a los componentes del dispositivo. Esta recibe a través de una batería una corriente de 5V.

En cuento a la Etapa de Periféricos, esta se encarga de la interconexión del microcontrolador con la memoria, los leds, la salida a pantalla, y todos los componentes externos que sean conectados al dispositivo.

La Etapa de Adaptación y su conector es el módulo principal de este dispositivo ya que es el que extrae información para alimentar el sistema. En este proyecto los ED no deben tener capacidad de actuación sobre el elemento al que monitorizan, con lo que la etapa de adaptación será únicamente una interfaz de lectura, a excepción de la estación de sensores de consumo eléctrico, los cuales sí disponen de un elemento actuador para el encendido y apagado de bombas.

En este sentido y debido a la gran cantidad de equipos existentes, se desarrollan bajo un enfoque de compatibilidad con respecto a otros estándares habituales en industria, como puede ser Modbus, o el uso de equipos comerciales que puedan ser integrados en la red con el objetivo de ampliar el número de magnitudes disponibles en el sistema.

Involucra información procedente del sistema de adquisición de medidas de AQUASIG, donde se incluyen los sistemas de adquisición de la medida para obtener información relacionada sobre:

- Caudal: asociado al consumo de agua.
- Presión: asociado a la presión de agua en la distribución.
- Calidad: asociado a sondas de medición de parámetros de calidad, como son:
    - Conductividad: mide de forma general todas las sales que tiene disueltas el agua.
    - pH: mide la acidez o alcalinidad del agua.
    - Oxígeno disuelto: mide el contenido de oxígeno disuelto.
    - Temperatura: mide la temperatura del agua, que influye en la solubilidad de iones, gases, etc.

A continuación, en los próximos apartados se describen de forma específica cada una de las estaciones de

medida desarrolladas, incluyendo la descripción de los diversos prototipos diseñados y desarrollados.

Todos los sensores del sistema de adquisición de medida proveen la información a la plataforma a través del envío de lecturas inalámbricas. La mayoría de las medidas que se toman en el proyecto proceden de una serie de procedimientos que resultan al final en las medidas, la magnitud que inicialmente se mide no es la que al final se representa, ya que necesita ser tratada y puesta en contexto para entender como esa medida representa la realidad que cuantifica.

La finalidad de esta sección es explicar los procedimientos que se llevan a cabo en las distintas medidas desde que se obtiene la primera medida física hasta que se representa el dato finalmente como el usuario final lo ve. Así, se hace una descripción del procedimiento para conseguir la medida.

5.2.2 Estación de Medida de Parámetros de Calidad

Como se ha comentado, la estación de medida diseñada puede incorporar multitud de sensores para medición de parámetros de calidad. De forma concreta, para la estación de medida de parámetros de calidad prototipo de AQUASIG se ha diseñado y desarrollado una estación con cuatro conexiones para elementos o sensores, incorporando sensores para la medición de cuatro parámetros clave para la calidad del agua, como son: pH, oxígeno disuelto, temperatura y conductividad eléctrica.

En la imagen siguiente se muestran las principales características medidas por los sensores seleccionados:

Figura 5-4. Sondas de parámetros de calidad.

Así, la estación de medida de parámetros de calidad consta de los siguientes elementos:

- Kit de medición de pH.
- Kit de medición de oxígeno disuelto.
- Kit de medición de temperatura.
- Kit de medición de conductividad eléctrica.

- Placa de control MSP432P401R.
- Módulo de comunicaciones XBEE PRO S2C.

La lógica de funcionamiento de la estación será la de obtener de los sensores individualmente y mediante el protocolo de comunicaciones I2C medidas de cada uno de los sensores, para luego aglutinarlas en una cadena de texto, la cual será enviada mediante el módulo de comunicaciones al coordinador de la red que se encargará de procesar esos datos.

5.2.2.1 Medida de PH

La medida del pH se hace mediante un método potenciométrico, este método se basa en que entre dos soluciones con distinta concentración de iones de hidrogeno se establece una diferencia de potencial. Esta diferencia de potencial determina que cuando se ponen en contacto haya un flujo de iones de hidrógeno o lo que es lo mismo una corriente eléctrica. En la medida real del pH no se miden directamente los iones de hidrógeno, sino que se hace de forma relativa, se compara con el de una disolución de la que se conoce el pH.

Para ello se usa un electrodo de pH, cuando el electrodo entra en contacto con la disolución se establece un potencial a través de la membrana de vidrio que recubre el electrodo. El potencial varía en función del pH. Para determinar el valor del pH se necesita un electrodo que no varíe su potencial para tenerlo de referencia.

La diferencia de potencial (E) es proporcional a los iones de hidrógeno, y viene definida por la ecuación de Nernst.

$$E_{medido} = E_{referencia} + 2.3 \cdot \frac{RT}{NF} pH$$

E_{medido} = Potencia en Voltios detectado a través de la membrana de vidrio

$E_{referencia}$ = Potencial en Voltios del electrodo de referencia

$2.3 \cdot \frac{RT}{NF}$ = Factor de Nernst

R = Constante de los gases

F = Constante de Faraday

N = Carga del ion

T = Temperatura en grados Kelvin

Figura 5-5. Medida de PH.

Para la realización de la calibración de las medidas es importante que la disolución se encuentre a temperatura ambiente ya que el comportamiento del electrodo depende de la temperatura.

El valor del factor de Nernst a una temperatura de 25oC vale aproximadamente 0.06 y el potencial de referencia se considera igual a 0, la ecuación anterior queda reducida a:

$$E_{medido} = -0.06\, pH$$

Este tratamiento de la medida se hace en la etapa de adquisición de la medida de pH, que una vez seguido el procedimiento de tratamiento de los datos los entrega mediante el método elegido al controlador, para que los datos enviados por la etapa de adquisición se puedan mostrar de forma que se entienda es necesario procesarlos para que puedan ser mostrados como una cadena de texto que sea legible para cualquier equipo.

Los datos en el modo I2C vienen dados en modo ASCII, debido a las particularidades del microcontrolador y

de la librería para el manejo del puerto I2C que se utilizan los caracteres son leídos de uno en uno y almacenados en una tabla de tipo caracteres. El carácter que se encarga de separar la parte decimal de la parte entera es el punto (“.”).

Una vez se han almacenado todos los caracteres correspondientes a la medida del pH se procede a almacenar la cadena que se entrega finalmente al sistema para que la medida se procesada por otros procesos tales como el de envío de información.

A continuación, se muestra una serie de datos obtenidos de una muestra real de los datos de pH procesados y preparados para ser mostrados.

pH (Unidad de pH)
8.494
8.495
8.495
8.493
8.493
8.491
8.493
8.494
8.494
8.494

Figura 5-6. Muestra Medidas de PH.

5.2.2.2 Medida de Oxígeno Disuelto

La medida de oxígeno disuelto se realiza mediante una sonda galvánica, que una vez pasan las moléculas de oxígeno a través de la membrana de la sonda se genera una corriente eléctrica, a medida que el oxígeno aumenta también lo hace la salida en mV de la sonda, consiguiendo así la medida. El nivel de mV que puede dar una sonda puede ser diferente para cada sonda siendo los 0 mV el único valor igual para todas que representa cuando no existe oxígeno en la disolución.

Los datos en el modo I2C vienen dados en modo ASCII, debido a las particularidades del microcontrolador y de la librería para el manejo del puerto I2C que se utilizan los caracteres son leídos de uno en uno y almacenados en una tabla de tipo caracteres, una vez recibidos esos datos es necesario interpretarlos para saber la información que contienen, si se detecta que en la cadena de datos se incluye el carácter “,” se entiende que es una cadena CSV que entrega dos valores, el primero de ellos es la saturación y el segundo el valor de oxígeno disuelto, cada uno de ellos va a una cadena separada.

Por regla general solo se va a usar el dato de oxígeno disuelto, pero se deja preparado el sistema para que sea capaz de usar el dato de saturación en caso de ser necesario.

A continuación, se muestra una serie de datos de oxígeno disuelto de una muestra real.

Oxigeno Disuelto (mg/L)
8.95
8.96
8.98
8.99
8.99
8.97
8.94
8.92
8.91
8.92

Figura 5-7. Muestra Medidas de Oxígeno Disuelto.

5.2.2.3 Medida de Conductividad Eléctrica

Dentro de la sonda de conductividad se encuentran dos electrodos posicionados uno enfrente del otro, se les aplica un voltaje AC a los electrodos causando que los cationes se muevan hacia la parte

negativamente cargada mientras que los aniones se mueven a la parte positivamente cargada, mientras más electrones libres contenga la disolución mayor será la conductividad eléctrica.

Los datos en el modo I2C vienen dados en modo ASCII, debido a las particularidades del microcontrolador y de la librería para el manejo del puerto I2C que se utilizan los caracteres son leídos de uno en uno y almacenados en una tabla de tipo caracteres.

Existen dos posibilidades a la hora de recepción de los datos, que se tenga solo el dato de conductividad que es el que se va a usar en la mayoría de los casos o que se disponga de una cadena CSV en la que se encuentren los datos de EC, TDS, SAL y SG.

- EC = conductividad eléctrica.
- TDS = total de sólidos disueltos.
- SAL = salinidad.
- SG = gravedad específica.

La opción de que devuelva el dispositivo la cadena CSV solo está disponible a partir de la versión 2.10 del firmware, pero se deja preparado para su soporte en caso de que sea necesario.

A continuación, se muestra una serie de datos de conductividad eléctrica de una muestra real.

Conductividad Electrica (uS/cm)
152.6
152.6
152.5
152.2
152.5
152.6
152.7
152.4
152.6
152.6

Figura 5-8. Muestra Medidas de Conductividad Eléctrica.

5.2.2.4 Medida de Temperatura

El dispositivo de sensorización está basado en una resistencia que varía su valor en función de la temperatura, dispone de un valor de tensión a la salida proporcional a la temperatura medida a la entrada, de forma que a mayor temperatura mayor será la tensión a la salida.

Para este sensor se tiene una referencia de 1kΩ a 0oC, la temperatura se calcula mediante la siguiente:

$$T = \frac{\sqrt{(-0.00232(R) + 17.59246)} - 3.908}{0.00116}$$

T = temperatura en grados Celsius.

R = resistencia medida por el sensor PT-1000 temperature probe.

Figura 5-9. Formula Calculo Temperatura.

En la imagen siguiente se muestra una pequeña tabla de la relación temperatura y resistencia del dispositivo.

°C		Ω	°C		Ω	°C		Ω
-10	=	960.9	7	=	1027.3	24	=	1093.5
-9	=	964.8	8	=	1031.2	25	=	1097.3
-8	=	968.7	9	=	1035.1	26	=	1101.2
-7	=	972.6	10	=	1039	27	=	1105.1
-6	=	976.5	11	=	1042.9	28	=	1109
-5	=	980.4	12	=	1046.8	29	=	1112.8
-4	=	984.4	13	=	1050.7	30	=	1116.7
-3	=	988.3	14	=	1054.6	31	=	1120.6
-2	=	992.2	15	=	1058.5	32	=	1124.5
-1	=	996.1	16	=	1062.4	33	=	1128.3
0	=	1000	17	=	1066.3	34	=	1132.2
1	=	1003.9	18	=	1070.2	35	=	1136.1
2	=	1007.8	19	=	1074	36	=	1139.9
3	=	1011.7	20	=	1077.9	37	=	1143.8
4	=	1015.6	21	=	1081.8	38	=	1147.7
5	=	1019.5	22	=	1085.7	39	=	1151.5
6	=	1023.4	23	=	1089.6	40	=	1155.4

Figura 5-10. Relación Temperatura-Resistencia.

Los datos en el modo I2C vienen dados en modo ASCII, debido a las particularidades del microcontrolador y de la librería para el manejo del puerto I2C que se utilizan los caracteres son leídos de uno en uno y almacenados en una tabla de tipo caracteres.

A continuación, se muestra una serie de datos de temperatura de una muestra real.

Temperatura (Grados)
26.196
26.196
26.195
26.194
26.193
26.192
26.191
26.189
26.187
26.185

Figura 5-11. Muestra Medidas de Temperatura.

5.2.2.5 Configuración de Sensores

Los sensores seleccionados son de carácter comercial, concretamente de la marca ATLAS SCIENTIFIC. Los sensores se configuran por defecto en modo UART, por lo que ha sido necesario un trabajo de reingeniería para el cambio y adaptación al modo I2C, que sea compatible con la estación de medida.

El cambio a modo I2C se ha llevado a cabo mediante la programación de una placa Arduino UNO, que se conecta del puerto serie de la placa al puerto serie del Arduino siguiendo un nuevo esquema de conexiones para poder comunicar los sensores con la etapa de adquisición.

Esto ha sido posible gracias al datasheet del fabricante, mediante la configuración con comandos programados

a bajo nivel para el cambio de protocolo de comunicaciones. Se han llevado a cabo varias pruebas, donde se ha identificado como óptimo el comando “I2C, n”, siendo el más apropiado para las características buscadas. De esta forma, para su programación, se introduce el comando teniendo en cuenta que la dirección del protocolo I2C (n) que se asigne a la etapa de adquisición debe de ser la misma que se introduce en el código a bajo nivel, y que la dirección no puede ser mayor que 127, siendo una restricción marcada.

Este proceso ha sido necesario realizarlo para todos los sensores, consiguiendo tenerlos todos preparados para la comunicación I2C, teniendo la ventaja que solo se necesita dos pines de la placa principal en vez de las múltiples UARTs que se hubieran tenido que crear para conectar los dispositivos, haciendo las estaciones de medida mucho más eficientes en cuanto a su configuración, simplificación del circuito y rendimiento de operación.

La conexión de los dispositivos es la siguiente:

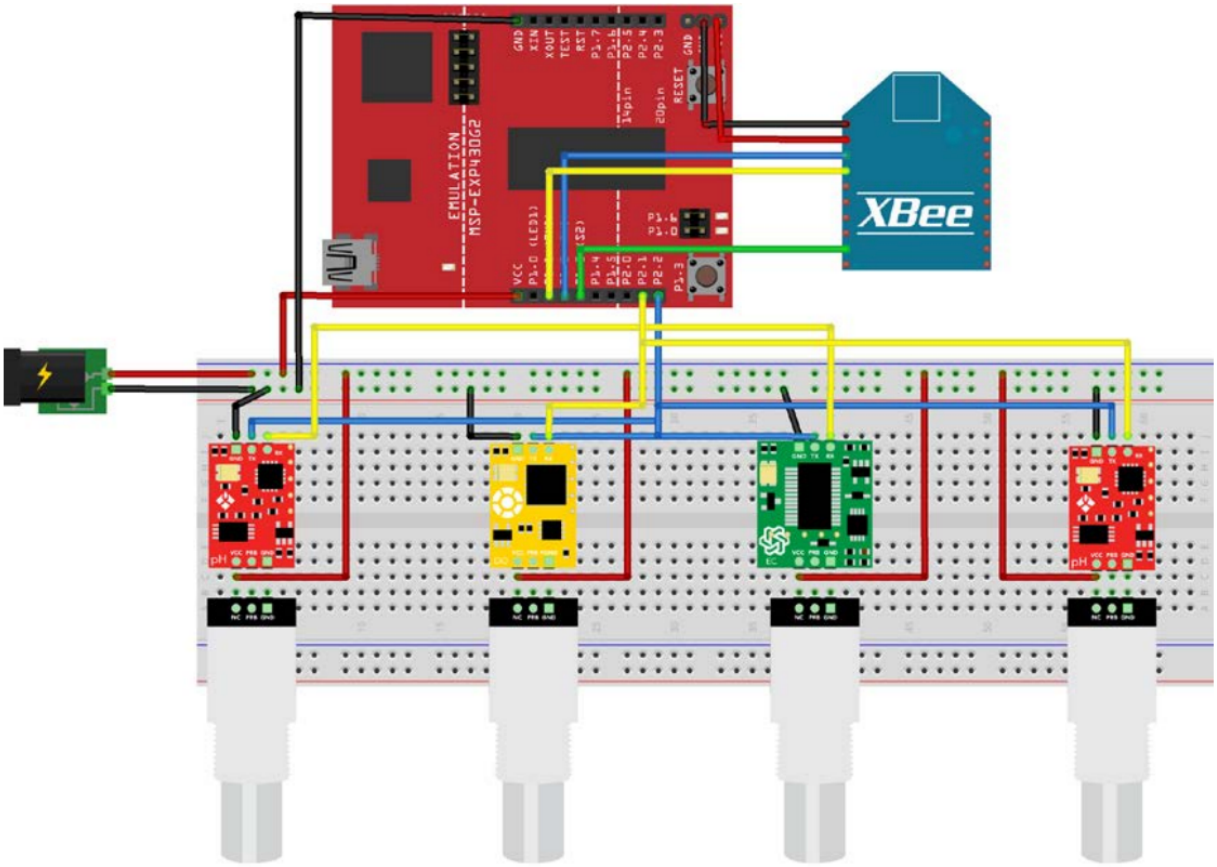


Figura 5-12. Composición Interfaz de medida de parámetros de calidad.

Una vez se han conectado los dispositivos es necesario configurar la placa principal para la adquisición de esos datos y el montar la trama de datos que va a ser enviada. Se configuran correctamente los dispositivos y las direcciones mediante las que serán accesibles.

5.2.2.6 Cadena de Datos

La cadena de datos que se ha creado aglutina la información proveniente de los sensores para su envío.

Esta cadena de datos contiene los datos obtenidos de todos los sensores de calidad desplegados. Este desarrollo ayuda a incrementar la compactación de tramas ya que no necesita que el dispositivo final envíe información de cada medida de forma independiente hacia la plataforma. Esto además reducirá el ancho de banda utilizado por los dispositivos finales y aumentará drásticamente el ahorro de batería.

El formato que tiene es el siguiente:

$$T(°C) * DO(mg/L) * pH(UPH) * EC(\mu S/cm)$$

Figura 5-13. Cadena de Datos Parámetros de Calidad.

Los "*" son los separadores para que a la hora de interpretar esos datos se sepa dónde empieza y donde termina cada dato.

Esta cadena será enviada al coordinador a través del módulo de comunicaciones. Posteriormente, el coordinador procesara esta información y la enviara de forma correcta al ecosistema IoT desplegado.

5.2.2.7 Desarrollo Software Interfaz de Medida de Calidad

El entorno de desarrollo donde se ha desarrollado el código necesario para el correcto funcionamiento de la aplicación ha sido ENERGIA [76], que es una plataforma creada por Texas Instrument para el manejo de sus launchpads, y está formado a partir de la plataforma Arduino, lo que hace que muchas de sus librerías y funcionalidades puedan ser utilizadas para optimizar el código y simplificar su codificación, así como su lenguaje de programación.

El funcionamiento del dispositivo se basa en extraer mediante el puerto I2C los datos de cada uno de los sensores de calidad (pH, oxígeno disuelto, temperatura y conductividad eléctrica) y almacenarlos en una variable en memoria, una vez se han conseguido los datos de todos los sensores se conforma una trama en modo texto para ser enviada mediante el módulo de comunicaciones. Así, cada vez que se tiene una nueva medida, ésta se envía.

Para la extracción de los datos se ha desarrollado un código específico para cada uno de los sensores, ya que se ha tenido que adecuar para que secuencialmente se vayan extrayendo medidas y se envíen posteriormente de forma conjunta.

Como método de comprobación para ver si la extracción y posterior creación de trama se ha utilizado una herramienta disponible en los launchpads que es el puerto serie que se conecta al PC, mediante una herramienta disponible de ENERGIA llamada Monitor Serie, donde se permite ver lo que se escriba en ese puerto, usándose como método de depuración.

Figura 5-14. Interfaz de medida de parámetros de calidad.

En las siguientes imágenes, se puede ver una muestra del código utilizado para la adquisición de medida de parámetros de calidad. Además, se puede ver cómo se han tenido en cuenta las consideraciones para el correcto funcionamiento de la placa.

```
#define address_ph 99           //default I2C ID number for EZO pH Circuit.
#define address_temp 102        //default I2C ID number for EZO RTD Circuit.
#define address_do 97           //default I2C ID number for EZO D.O. Circuit.
#define address_ec 100          //default I2C ID number for EZO EC Circuit.
#define SLEEP_PIN 6             //pin for sleep xbee
#define SLEEP_TIME 10000        //In miliseconds

void setup() {
  Serial.begin(9600); // Puerto serie activo para depuracion
  Serial1.begin(9600); //Puerto serie para dispositivo Xbee

  delay(2000);  // Dormimos 2s

  Serial.println("Iniciamos bus I2C");
  Wire.begin();       // Iniciamos bus I2C

  //iniciaSerial();

  Serial.println("Inicializamos lectura Analogica");
  analogReadResolution(14);

  Serial.println("Inicializamos referencia analogica");
  analogReference(INTERNAL2V5);

  Serial.println("Configuramos puerto serie para Xbee");
  xbee.setSerial(Serial1); //Indicamos Xbee puerto seria a usar

  Serial.println("Arrancamos Xbee");
  xbee.begin(Serial1);

  Serial.println("Configuramos pin para dormir dispositivo");
  pinMode(SLEEP_PIN,OUTPUT);

  delay(2000); // Dormimos 2s

  //calibracion de los dispositivos
  bool ec = false;
  bool ph = false;
  bool d_o = false;

  //EC
  if(ec){
    Serial.println("Calibramos Sensor Conductividad");
    calibrate_ec();
    Serial.println("Calibracion Conductividad Finalizado");
  }

  //Ph
  if(ph){
    Serial.println("Calibramos Sensor Ph");
    calibrate_ph();
    Serial.println("Calibracion Ph Finalizado");
  }

  //DO
  if(d_o){
    Serial.println("Calibramos Sensor Oxigeno Disuelto");
    calibrate_do();
    Serial.println("Calibracion Oxigeno Disuelto Finalizado");
  }
  Serial.println("Setup Finalizado");
}
```

Figura 5-15. Código Interfaz de Medida de Parámetros de Calidad (I).

```
void loop() {

  Serial.println("loop");

  // Configuramos el parametro 'computerdata' con valor "r"
  // Valor que sera mandado al sensor via i2c para que
  // nos devuelva el valor de la lectura
  computerdata[0] = 'r';
  computerdata[1] = 0;

  Serial.println("Leemos pH:");
  lee_ph(); // Funcion que se comunica con el sensor de PH via i2c

  Serial.println("Leemos temperatura:");
  lee_temp(); // Funcion que se comunica con el sensor de temperatura via i2c

  Serial.println("Leemos oxigeno disuelto:");
  lee_do(); // Funcion que se comunica con el sensor de oxigeno disuelto via i2c

  Serial.println("Leemos conductividad electrica:");
  lee_ec(); // Funcion que se comunica con el sensor de conductividad electrica via i2c

  Serial.println("Leemos voltaje bateria:");
  lee_voltaje_bateria(); // Funcion que mide el voltaje de la bateria

  Serial.println("Transmitimos datos a coordinador");
  tx_data(); // Transmitimos informacion via Xbee

  Serial.println("Dormimos");
  sleep_(); // Dormimos los dispositivos // Ahorro energia

  }
```

Figura 5-16. Código Interfaz de Medida de Parámetros de Calidad (II).

5.2.3 Estación de Medida de Parámetros de Consumo

En este apartado se describen los desarrollos llevados a cabo para conseguir obtener el consumo de un cliente en tiempo real sin tener que ir a ver el contador físicamente, lo que configura la estación de consumo. Si bien, actualmente la mayoría de los contadores digitales muestran información en tiempo real del consumo, todavía hay algunas carencias para disponer de medidas individualizadas.

Los elementos de los que consta la estación de consumo son los siguientes:

- Sensor de caudal FS300A 1 – 60 L.
- Placa de control MSP432P401R.
- Módulo de comunicaciones XBEE PRO S2C.

El esquema sigue la misma arquitectura de la estación de calidad, utilizando la misma placa de control y el mismo módulo de comunicaciones, con un funcionamiento similar al descrito en el apartado anterior, con la diferencia de que el sensor es uno específico para medir caudal, no parámetros de calidad.

5.2.3.1 Configuración del Sensor

El funcionamiento del sensor FS300A se basa en una espiral que se mueve con el paso del agua, la cual lleva incorporado un imán, cada vez que ese imán da una vuelta se genera un pulso en la salida, se basa en el efecto hall para detectar cada vez que ha dado una vuelta.

En este sensor es necesaria una calibración ya que dependiendo de la presión que haya tendremos que ajustar cada ciertos pulsos es una unidad de medida de volumen, se puede calcular el caudal que está pasando en cada momento con el factor proporcionado por el fabricante, pero para extraer del caudal los datos de consumo se necesita contar tiempo de forma exacta para realizar una calibración correcta.

Cualquiera que sea el método elegido es necesaria una calibración posterior para ajustarlo al contador que se esté usando para que las medidas sean los más fieles posibles y no haya una desviación excesiva del valor de la medida.

Pensando ya en una implementación real se hace más importante la calibración del dispositivo debido a que una desviación excesiva podría conllevar cobros por encima o por debajo del consumo de los clientes, por lo

cual este será un tema importante y en el que se han ajustado los factores para que sea lo más fiel posible a la realidad, teniendo en cuenta que la realidad siempre se considerará la del contador que ya esté instalado.

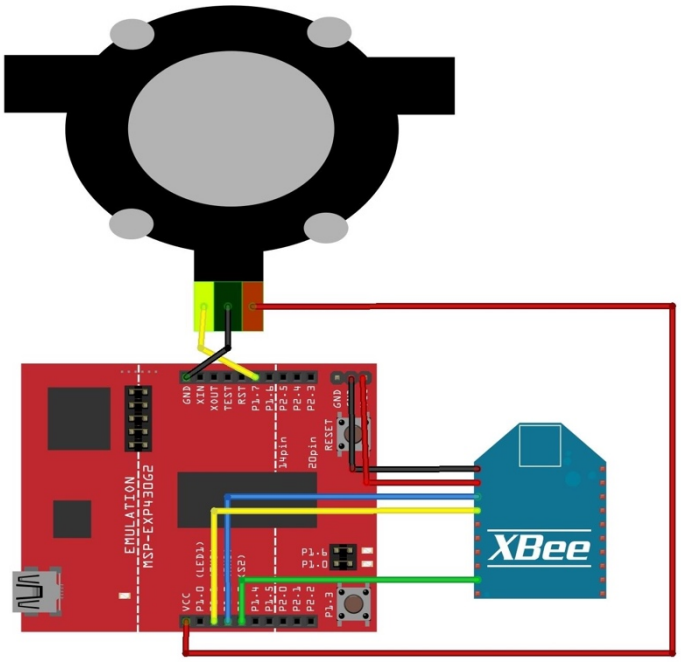


Figura 5-17. Interfaz de Medida de Consumo.

Una vez diseñado el dispositivo se conecta para realizar las pruebas y se le carga un código software para activar una interrupción, para que cada vez que se genere un pulso se muestre el número de pulsos por pantalla.

Los datos se transforman a formato cadena de caracteres para que su transporte e interpretación sea más sencilla en los otros extremos del sistema encargado de mostrar los datos.

A continuación, se muestra una serie de datos de consumo de una muestra real.

Consumo
10.43
13.81
16.81
19.95
22.62
25.71
28.67
31.33
34.33
37.43

Figura 5-18. Muestra Medidas de Consumo.

5.2.4 Estación de Medida de Parámetros de Presión

La estación de medida de parámetros de presión es similar a la descrita para la estación de medida de caudal tipo intrusivo, ya que el elemento de medida de presión debe ser intrusivo.

De esta forma, los elementos de los que consta la estación de presión son los siguientes:

- Placa de control MSP430FR6989.
- Módulo de comunicaciones XBEE PRO S2C.
- Sensor de presión hidráulica para agua de 0-1.2 MPa.

Figura 5-19. Sensor de Presión.

El funcionamiento del sensor de presión se basa en emitir un valor analógico de voltaje según la presión que haya en el agua. El rengo de medida oscila entre 0 y 1.2 Mega Pascales. El sensor se alimenta con una tensión de 5V, y emite un valor de 0.5V a 4.5, correspondiéndose los 0.5V con un valor de 0MPa y los 4.5V con 1.2MPa.

Una vez se tiene la lectura del sensor de presión, se convierte la lectura digital a voltaje a través de la siguiente ecuación: voltaje=Valor analógico*(3.3/4096)

El valor analógico es el valor que recoge el convertidor analógico-digital desarrollado, que se encarga de cuantificar el valor analógico en un rango de 0 a 4096 unidades, donde 0 equivale a 0V y 4096 equivalen a 3.3V.

Se obtiene el voltaje obtenido por la entrada analógica multiplicando el valor analógico por el voltaje de alimentación de la placa, y dividiéndolo por el número de valores escalones analógicos que la entrada puede diferenciar.

En nuestro caso, el sensor recibe valores de 0.5V a 3.3V y se convierte a Mega Pascales aplicando la siguiente ecuación: presionMPa=voltaje-0.5*(1.24.5)

Para terminar, se cambia de escala a Kilo Pascales multiplicando por mil: presionKPa=presionMPa*1000.

Una vez obtenido el valor de presión en Kilo Pascales, se envía al coordinador a través del módulo de comunicaciones al coordinador. Posteriormente, este se encarga de transmitir la información del sensor al núcleo de la plataforma.

Como se ha comentado, el prototipo se asemeja al descrito la Estación de Medida de Parámetros de Consumo, haciendo uso del sensor intrusivo de presión.

5.2.5 Arquitectura Red Dispositivos Finales

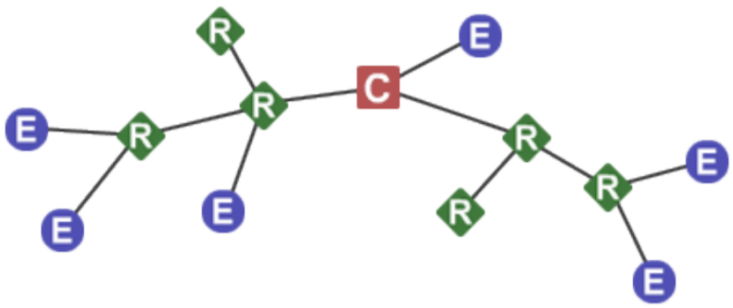
Puesto que uno de los mayores condicionantes es la duración de la batería, pasando por la escalabilidad y la distancia que pueda ser cubierta, la decisión que se ha tomado es seleccionar y utilizar el protocolo ZigBee, ya que está enfocado a bajo consumo de los nodos.

Zigbee es un protocolo de red basado en 802.15.4. Es un estándar inalámbrico de bajo consumo que funciona en la banda de 2.4GHz. Zigbee implementa la capa de red y la subcapa de aplicación sobre los niveles de enlace y físico de IEEE 802.15.4. La capa de red permite crear un mesh, debido a que hay dispositivos con capacidad de rutado, gracias a la funcionalidad de la capa de red. En interiores el alcance es de unos 75m mientras que en exterior con visión directa puede llegar a 1km. El número máximo de dispositivos direccionables es 65535, que pueden distribuirse en topologías tipo estrella, árbol o malla.

Figura 5-20. Protocolo ZigBee [77].

Este protocolo ofrece una ratio consumo/alcance acorde a lo que se necesita en el proyecto, ofrece también una alta escalabilidad y la posibilidad de direccionar muchos nodos, que para zonas de alta densidad de población es adecuado. Surge el problema de que en este protocolo conforme subimos el número de nodos la probabilidad de colisión aumenta, lo que se ha tenido en cuenta a la hora de restringir el número máximos de nodos que tendrá un concentrador.

La topología que se va a emplear es una mezcla de la topología de tipo malla. Los nodos end-devices tienen la capacidad de conectarse directamente con el coordinador si alcanza directamente la cobertura, en caso de que no lleguen existe la posibilidad de colocar un router que le haga de intermediario hasta el coordinador. Hay que tener en cuenta que el número máximo de elementos que el coordinador puede direccionar son 255 entre routers y end-devices. Cada router a su vez puede direccionar hasta 255 end-devices. Por ello, para una mayor escalabilidad de la red, será necesaria la introducción de routers, existiendo también la posibilidad de simplificar la red si el número de nodos es pequeño.



Dispositivo	Función
Coordinador C	Nodo central de la red, se encarga de crearla y gestionarla
Router R	Se encarga de dar cobertura a dispositivos de menor jerarquía
End – Device E	Nodos recolectores de datos, capacidad de "dormirse"

Figura 5-21. Topología en malla.

Los dispositivos de comunicaciones para implementar la tecnología ZigBee en el proyecto, han sido los XBEE PRO S2. Dichos dispositivos son unos de los más usados actualmente bajo el marco de la tecnología Zigbee. Estos dispositivos cuentan con la certificación correspondiente que les habilita a ser usado en entornos industriales.

Figura 5-22. Módulo XBee PRO [78].

Los Xbee son una familia de dispositivos de comunicaciones de la empresa Digi que implementa su propia versión de ZigBee en sus dispositivos. Disponen de dos modos de funcionamiento, el modo transparente y el modo API, siendo el modo transparente para comunicaciones punto a punto y el modo API para crear redes de dispositivos en los que se puede obtener más información de la red. Implementan niveles de seguridad dentro del protocolo. Las características que resaltan del dispositivo son las siguientes:

- Cobertura de 90m en interiores y 3200 en exteriores con visión directa.
- Potencia de transmisión de 18 dBm.
- Es posible comunicarse con el mediante la UART y SPI.
- La tensión de alimentación es de 3.3V.
- Posee un modo de bajo consumo donde se apaga la etapa RF donde se consiguen consumos del orden de los 5 μA.
- Tiene la posibilidad de crear redes de tipo malla.
- Puede funcionar como coordinador de la red, router o end device.
- Bit rate: 250 kbps.
- Usa la banda de 2.5 GHz.
- Potencia de transmisión: 63 mW (+18 dBm).

A la vista de las características de los dispositivos se ha elegido usar el XBEE PRO S2 debido a las siguientes consideraciones:

- Se dispone de una comunidad muy activa.
- Cubre teóricamente una distancia mayor.
- Su montaje es THD lo que implica que tiene valor añadido para prototipos.
- Tiene la posibilidad de usar un modo (API) en el que se implementan tramas de protocolo para el intercambio de mensajes.
- Dispone de niveles de seguridad configurables para la red.
- Disposición de librerías adaptadas en algunas plataformas para el manejo del modo API.

Ha sido necesaria la configuración del dispositivo XBEE mediante el software del fabricante, para introducirle los siguientes parámetros:

- PAN ID: este parámetro debe ser igual para todos los nodos de la red, ya que las redes se identificarán por su PAN ID, se puede usar un número arbitrario.
- Channel verificantion: este parámetro busca si hay un coordinador activo en la red y en caso de haberlo tratar de conectarse a él, estará en este caso.
- Encryption enable: sirve para activar el que haya seguridad en la red, se activa en este caso usando el valor 1.
- Encryption options: configura las opciones de encriptado, en el caso del coordinador se usa el valor 8, este indica que la comprobación de que la clave es correcta se hará en el periodo de conexión.
- Encryption key: es la clave de acceso a la red, esta es la clave que necesitaran los dispositivos de la red para poder acceder a ella.
- Destination address High y Destination address Low: este parámetro sirve para identificar al nodo destino de la comunicación, con el que hay que conectarse, es este caso se introducirá la dirección del coordinador, que será el receptor de los mensajes.

- API enable: este parámetro permite configurar qué modo de comunicación usa el dispositivo, en este caso se usará el modo API enable with escaping.
- Sleep mode: este parámetro permite controlar el modo en el que el dispositivo dormirá, en este caso el modo que interesa es el pinHiberante, que permitirá que el dispositivo será dormido mediante un pin, el cual será controlado desde la placa principal.

Configurando correctamente los parámetros anteriores el dispositivo está listo para ser usado.

Para la configuración de los dispositivos, se utiliza el módulo que el fabricante ofrece llamado XBee Explorer USB, el cual permite configurar y operar el modem Xbee desde PC. Basta con conectar el modem XBee Serie 2 al USB y conectarlo al ordenador tras haber instalado los drivers correspondientes.

Para configurar los módulos, abriremos el software XCTU [79]. Este software lo ofrece la empresa Digi de manera gratuita y está disponible tanto para Windows, Linux y MacOS. En la misma web, se encuentra una extensa variedad de información y tutoriales que facilita la configuración de la topología tipo malla diseñada en el proyecto.

5.2.6 Interfaces Inalámbricas Equipos Terminales

Los dispositivos finales forman parte de la red inalámbrica IEEE802.15.4 creada por el coordinador de la red, con las características impuestas por éste. Cada dispositivo final estará identificado de manera unívoca dentro de la red gracias a la dirección de 16bits otorgada por el coordinador, pero además estará unívocamente identificado en cualquier red gracias a su dirección fija MAC de 64bits.

Al estar directamente conectados a los dispositivos finales que queramos controlar, ya sea sólo para monitorizar o también para incluir algo de actuación sobre él, dispondrán de un bloque extra que se encargará de realizar esta adaptación entre dispositivos.

El envío de datos se hace mediante el módulo de comunicaciones XBEE PRO S2C que se basa en el estándar de comunicaciones ZigBee. El dispositivo posee dos modos de funcionamiento, se usa el modo API que es el más potente y beneficioso para la aplicación.

Para el uso del modo API existe una librería que facilita el manejo y envío de los datos, siendo necesario tener la cadena con los valores a enviar en una tabla de tipo CHAR. Es necesario conocer la dirección MAC del dispositivo destino. En el caso de la aplicación la dirección que debe aparecer como destino es la del nodo coordinador.

```
XBeeAddress64 addr64 = XBeeAddress64(0x0013a200, 0x41557031);
```

Figura 5-23. Configuración dirección XBee coordinador.

La información viaja encriptada por el aire hasta llegar al destino donde será descifrada.

Comprobado que la extracción y posterior creación de trama es correcta se pasa a comprobar que el envío de los datos a través del módulo ZigBee es correcto, para ello es necesario tener un receptor de los datos configurado como coordinador de red. Para ello su usa el equipo concentrador de datos. La prueba de los dos dispositivos es conjunta, ya que uno y otro se prueban entre ellos para descubrir problemas que puedan aparecer.

Cabe destacar que esta comunicación viaja por el aire encriptada mediante una clave generada por el dispositivo que encripta la red y otra que permite el acceso de los dispositivos a ella. No es necesario preocuparse por esta encriptación ya que se genera automáticamente por el módulo de comunicaciones, la configuración de las claves se hace mediante el software del módulo de comunicaciones. Por seguridad las claves de acceso al medio y encriptación de la red no pueden ser leídas mediante el software de configuración, solamente pueden ser escritas, esto es importante ya que de no ser así cualquiera que tuviera acceso al dispositivo coordinador podría tener acceso a las claves.

A través del puerto Serie del dispositivo, se puede contemplar en pantalla si el dispositivo realiza el envío de manera satisfactoria. En caso contrario, en el monitor se mostrará el error que se haya podido originar como error al encontrar coordinador, al leer paquete de respuesta, o errores del propio dispositivo transmisor.

A continuación, una muestra del código que realiza el envío desde el dispositivo.

```
void tx_data(){
  // Cadena de datos a enviar al coordinador
  cadena = "";
  cadena.concat('Q');
  cadena.concat(RTD_data);
  cadena.concat('*');
  cadena.concat(DO_data);
  cadena.concat('*');
  cadena.concat(ph_data);
  cadena.concat('*');
  cadena.concat(ec_data);
  cadena.concat('*');
  cadena.concat(voltaje);

  // Longitud de la cadena a enviar
  longitud = cadena.length();

  // Convertimos cadena a char y la pasamos a una lista de char
  cadena.toCharArray(char2uint8,longitud);
  for(int j=0; j<=longitud; j++)
    payload[j] = uint8_t(char2uint8[j]);

  //Se crea el mensaje para ser enviado por el Xbee
  ZBTxRequest zbTx = ZBTxRequest(addr64, payload, sizeof(payload));
  ZBTxStatusResponse txStatus = ZBTxStatusResponse();
```

Figura 5-24. Método de transmisión de datos mediante XBee (I).

```
//Este bloque es para mostrar lo que se envia
for(int k=0; k<=longitud; k++)
  Serial.print(char(payload[k]));
Serial.println();

//Transmit
xbee.send(zbTx);

// Esperamos respuesta
if (xbee.readPacket(500)) {
  // Obtenemos respuesta
  // Debe ser un valor de stado
  if (xbee.getResponse().getApiId() == ZB_TX_STATUS_RESPONSE) {
    xbee.getResponse().getZBTxStatusResponse(txStatus);

    // obtenemos estado
    if (txStatus.getDeliveryStatus() == SUCCESS) {
      // Envio correcto

    } else {
      // Envio fallido, esta el xbee receptor encedido?
      Serial.println("Envio xbee fallido. Comprueba encendido coordinador");
    }
  }
} else if (xbee.getResponse().isError()) {
  Serial.println("Error reading packet.  Error code: ");
  Serial.println(xbee.getResponse().getErrorCode());
} else {
  // XBee no provee respuesta. No debe pasar
}
}
```

Figura 5-25. Método de transmisión de datos mediante XBee (II).

5.2.7 Equipos Routers

Estos dispositivos se encargan de aumentar la cobertura de la red. Pueden colocarse más de uno. Está formado por un XBEE POR S2C, acompañado de una fuente de alimentación que aporte energía al módulo.

Su funcionalidad es la ampliar la distancia que sea capaces de cubrir los módulos además de dotar a la red de mayor escalabilidad si es necesaria. Si la distancia entre el coordinador y el nodo final es lo suficientemente grande como para que la comunicación directa no sea posible, se introducirán routers que actuaran como pasarelas para los datos hacia el coordinador de la red. En caso de que estemos direccionando más nodos de lo que es posible para el coordinador, la introducción de routers permitirá que el sistema se escale en topología de árbol, ampliando la capacidad de la red de soportar más dispositivos. Pueden unirse a redes existentes ya creadas por los coordinadores, la limitación de routers que se conectar directamente con el coordinador lo pone la tecnología, ascendiendo a 255 nodos que son el máximo que el equipo central puede direccionar. No se les permite entrar en modos de bajo consumo, lo que hace que tengan que estar siempre despiertos. Existe la posibilidad de incorporar una serie de routers que se conecten entre ellos si la distancia con el dispositivo final es grande para así poder ampliar la zona de cobertura.

Dado los niveles de cobertura del protocolo, se ubican routers por cada bloque de pisos o manzana de viviendas para asi evitar al máximo perdidas de informacion de los dispositivos responsables del consumo de

agua.

Figura 5-26. Router ZigBee.

5.2.8 Equipos Concentradores de Datos

Como se ha comentado con anterioridad, este sistema es el encargado de recopilar todos los datos recogidos por los dispositivos inalámbricos.

El cuerpo principal del sistema de adquisición está formado por el llamado coordinador que es el concentrador de datos o pasarela de red. Es el elemento último de la red al que van dirigidas las medidas recogidas por los elementos finales conectador a los contadores de agua y demás elementos de la red.

Es un dispositivo complejo que contiene tanto al coordinador de la red inalámbrica como a un PC embebido y otra serie de elementos que completarán la funcionalidad esperada por el Concentrador de la red.

5.2.8.1 Arquitectura Hardware

La arquitectura HW necesaria para el funcionamiento del Concentrador es la siguiente:

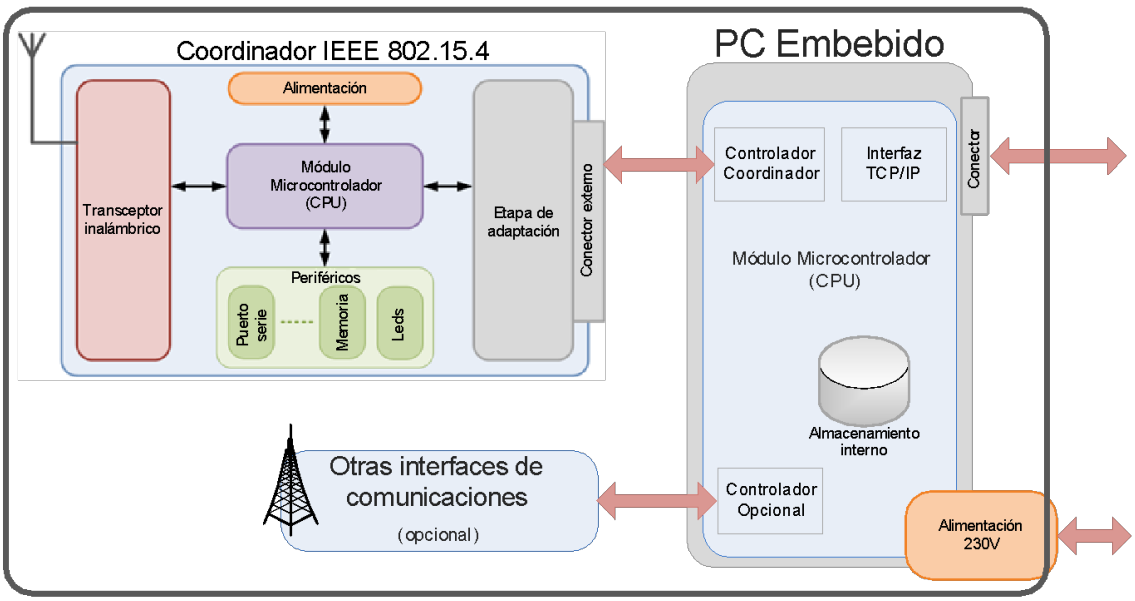


Figura 5-27. Arquitectura Concentrador.

Como se puede observar en la imagen anterior, el coordinador de la red inalámbrica está integrado dentro de la arquitectura general del concentrador de datos.

De manera que los elementos HW que formarán parte del concentrador son los siguientes: coordinador de red, PC Embebido y otras interfaces de comunicaciones.

El coordinador de la red es presenta una estructura interna prácticamente similar a la de los dispositivos finales, ya que es él el que hace de interfaz entre la red inalámbrica y el resto de los sistemas. Sus bloques internos principales son:

- Etapa de alimentación: El dispositivo concentrador completo de alimenta mediante una toma de red eléctrica a 230V, con lo que la etapa de alimentación debe ser tal que pueda convertir este tipo de alimentación en los niveles de tensión e intensidad requeridas por el módulo microcontrolador y por el resto de los periféricos del coordinador.
- Etapa de adaptación: La conexión con la placa principal se hará mediante la UART disponible para ello.
- Transceptor inalámbrico: El transceptor inalámbrico es el bloque encargado de convertir la información recogida en el microcontrolador en energía capaz de ser transmitida de manera inalámbrica. Dicho transceptor cumplirá con las especificaciones del estándar IEEE802.15.4, cuyos puntos principales son:
    - Dos bandas posibles. 2.4GHz o 868MHz, ambas son bandas libres ISM. Con lo que no hay que pagar ningún tipo de licencia por su utilización.
    - Potencia máxima de emisión (con especial cuidado a la selección de la antena).
    - Latencia de media de 4ms por cada paquete (provocado por los mecanismos de acceso al medio).
    - Carga útil entre 80 y 100 bytes por trama.

  La banda de frecuencia escogida para el proyecto será la banda de 2.4 GHz que viene impuesta por el módulo elegido.

  El uso de una antena de la ganancia necesaria para la mejora de la cobertura también está contemplado dentro de este bloque.
- Módulo microcontrolador (CPU): El módulo microcontrolador es el bloque encargado de controlar a todos y cada uno de los elementos que forman parte del dispositivo. Por esto debe tener una capacidad de computación suficiente como para controlar a todos los periféricos sin descuidar la gestión de los datos y su envío a través de la red inalámbrica. Es un microcontrolador de bajo coste y con posibilidad de muy bajo consumo para optimizar el gasto que el mismo pueda tener. Además, dispone de memoria interna suficiente como para alojar a la aplicación completar que realizará las tareas previstas para cada dispositivo en concreto.
- Periféricos: Además de los bloques nombrados, existe otra serie de bloques secundarios necesarios para el funcionamiento global de los dispositivos, de hecho, dentro de periféricos pueden englobarse los circuitos de adaptación mencionados en las etapas de adaptación externa y de alimentación. Estos periféricos se usan para complementar y depurar el funcionamiento del dispositivo.

Como se ha comentado anteriormente el PC embebido es un sistema inteligente con alta capacidad de cálculo capaz de interpretar los datos enviados por el coordinador, almacenarlos y hacer de pasarela de comunicaciones con los sistemas externos de gestión.

Básicamente tendrá una estructura de SBC (Single Borde Computer) con las interfaces necesarias para su conexión con los elementos con los que tenga que interactuar:

- Conexión I2C y SPI.
- Conectores genéricos para uso de puertos series, RS 232, RS485, etc., según necesidad.
- Conector y driver USB.

Aunque en la imagen el bloque de alimentación aparece prácticamente como una parte más del PC embebido, realmente es un elemento más del concentrador. Su función principal es la de convertir la tensión alterna que alimenta al dispositivo completo en una tensión continua dentro de un rango admisible para la electrónica utilizada en el concentrador.

De manera independiente, cada uno de los elementos que componen el concentrador dispondrán de etapas de adaptación específica para el correcto funcionamiento de ellos mismos.

La CPU del PC Embebido es un microcontrolador capaz de realizar complejas operaciones de comunicación y gestión de datos, siendo además capaz de controlar todos los periféricos que intervienen en el funcionamiento del equipo.

Para la obtención de las coordenadas GPS se utiliza un dispositivo externo que esté permanentemente calculando la posición GPS y pueda ofrecer esta posición al PC embebido siempre que éste se lo requiera. De esta manera esta interfaz GPS debe tener la inteligencia justa para el cálculo de la posición GPS y que además permita la consulta y configuración de esta posición a través de una interfaz serie o similares.

Para el envío de datos través de GSM se necesitará que se disponga de un puerto serie para ello, para así tener un sitio mediante el que se puedan enviar los datos directamente al servidor, se hará mediante un módulo externo que permitirá la conexión mediante una tarjeta SIM con una tarifa de datos. En muchas ocasiones, el mismo módulo GPRS, es capaz de realizar la función de localizador GPS de una forma relativamente precisa. Además, este mismo modulo, también puede ayudarnos a sincronizar los relojes de los dispositivos finales con toda la red. Esta funcionalidad que tienen los módulos GPRS nos servirá para que toda la red comporta un reloj síncrono común y podamos obtener las marcas temporales de los datos enviados por los sensores de la manera más precisa posible.

5.2.8.2 Arquitectura Software

En este apartado se describen los principales bloques SW que operan concretamente dentro del PC Embebido y el camino que siguen los datos desde que son recibidos hasta que son enviados al servidor de la plataforma.

Es importante señalar que todo el software que se desarrolla sobre el PC Embebido corre sobre la plataforma Arduino por lo que los dispositivos necesitarán disponer de protocolos de comunicación disponible.

En cuanto a los principales bloques funcionales desarrollados se tienen los siguientes: Controlador de red Man y controlador GPRS.

El bloque del Controlador de red Man es el encargado de recoger las tramas de datos enviadas por el coordinador procedentes de la red inalámbrica. Se encargará de recoger y procesar toda esta información de manera rápida y fiable para evitar pérdidas de información.

El módulo que se encarga de ejercer como coordinador es el XBEE PRO S2C configurado en modo coordinador de la red, siendo el encargado de recibir los mensajes provenientes de los distintos nodos de la red y entregar esa información a la placa Arduino para su posterior envío a la red. La configuración de los parámetros se hace mediante el software del fabricante XCTU.

Los parámetros para configurar este nodo son los siguientes:

- PAN ID: este parámetro debe ser igual para todos los nodos de la red, ya que las redes se identificarán por su PAN ID, se puede usar un número arbitrario.
- Coordinator enable: este parámetro sirve para indicar si el nodo es coordinador o no, en este caso si será coordinador por lo cual usamos el valor 1.
- Encryption enable: sirve para activar el que haya seguridad en la red, se activará en este caso usando el valor 1.
- Encryption options: configura las opciones de encriptado, en el caso del coordinador se usará el valor 2 ya que queremos que funcione como “trust center”, que será donde se compruebe que los nodos disponen de las credenciales para acceder a la red.
- Encryption key: es la clave de acceso a la red, esta será la clave que necesitaran los dispositivos de la red para poder acceder a ella.
- Network encryption key: es la clave que encripta las comunicaciones.
- API enable: este parámetro permite configurar qué modo de comunicación usará el dispositivo, en este caso se usará el modo API enable with escaping.

Configurados todos los parámetros a los valores indicados ya se tiene listo el módulo para la comunicación. Hay que tener en cuenta que esté módulo debe de tener siempre conexión a la red eléctrica por indicación del protocolo usado por el fabricante.

La gestión del dispositivo se hace desde la placa principal, se busca un flujo simple para el funcionamiento, la idea general del funcionamiento del dispositivo es la de estar continuamente escuchando el medio de transmisión para cuando llegue algún paquete leerlo, comprobar si es correcto y en caso de que lo sea levantar una bandera para indicar al flujo principal que se tiene un nuevo dato para ser procesado.

El bloque del Controlador GPRS es el encargado de darle salida a los datos provenientes de los end-devices hacia el servidor. Actúa como pasarela de los datos.

Este es el encargado de, mediante un método GET, enviar la información al servidor. La información va en una cadena que a su vez está incluida en una dirección web. Mediante esa petición los datos llegan a la plataforma AQUASIG para su posterior explotación.

Este módulo no está continuamente enviando datos ya que muchos de ellos se repetirían, teniendo en cuenta también el desaprovechamiento de recursos que esto conllevaría, cuando se reciba un nuevo dato mediante el módulo ZigBee la rutina de funcionamiento levantará una bandera para indicar que hay un nuevo dato, esta bandera activará el bloque de envío de dato al servidor, el cual también incluirá la tarea de bajar la bandera cuando el dato ha sido enviado.

```
void loop() {
  // Recibimos informacion, si la hay
  recibeXbee();
  // Si hay informacion, enviamos a la plataforma
  if(nueva_medida == 1){
    enviaGSM();
    nueva_medida = 0;
  }
```

Figura 5-28. Fragmento Código Concentrador.

En la figura anterior se muestra un fragmento del código de la rutina del coordinador. En ella, se ve como la función "recibeXbee" escucha al módulo de comunicaciones IEE802.15.4 y si esta recibe alguna información de cualquier dispositivo final, esta levanta la bandera para que se puedan enviar estos datos a través del módulo GSM.

5.2.8.3 Integración Coordinador

Finalmente, los elementos que fueron seleccionados para formar parte del coordinador son:

- El Arduino MEGA como pc embebido.
- El módulo XBEE PRO S2C para la comunicación con la red de sensores mallada.
- Módulo AdaFruit FONA 800 para transmitir la información a la plataforma AQUASIG mediante GPRS.

Figura 5-29. Concentrador Final.

El código desarrollado para este coordinador está escrito en Arduino, el lenguaje de programación de la plataforma Arduino.

Se han diseñado funciones específicas para obtener la hora local a través de la red GSM. Por último, se ha

optado por no incluir funciones de ahorro de energía en los dispositivos coordinadores, debido a que estos dispositivos están expuestos a la recepción de información por parte de cualquier dispositivo productor en cualquier instante y deben de estar preparados siempre para el envío de tramas hacia la plataforma, además de que en campo podrán situarse en ubicaciones con conexión directa a la red eléctrica. Es por esto, que los concentradores de datos se situaran de manera individual en cada calle o manzana de viviendas ya que como máximo cada uno podrá direccionar 255 dispositivos en total.

5.3 Agente IoT

El agente IoT está desarrollado usando NodeJS debido a la naturaleza orientada a eventos del sistema. Este, este preparado para la recepción de información desde diferentes protocolos como MQTT, HTTP, CoAP, etc. En nuestro caso, vamos a usar la implementación del agente para su uso sobre HTTP.

Para el correcto funcionamiento del componente, este necesitara de un componente que le aporte persistencia en la información. Podemos usar tanto la memoria interna del dispositivo como una base de datos externa. En nuestro caso, para aumentar la eficiencia de la arquitectura, se desplegará junto al agente una base de datos de naturaleza no relacional de tipo MongoDB.

Para el despliegue del agente, usaremos la tecnología dockers para encapsular dicho componente y crear un entorno de desarrollo aislado sobre este.

```
version: '2'
services:
    iot-agent:
        image: iota
        hostname: iot-agent
        network_mode: bridge
        container_name: fiware-iot-agent
        external_links:
            - fiwareorion_mongo_1:mongo
            - fiwareorion_orion_1:orion
        expose:
            - "4061"
            - "9000"
        ports:
            - "4061:4061"
            - "9000:9000"
        environment:
            - "IOTA_CB_HOST=orion" # name of the context broker to update context
            - "IOTA_CB_PORT=1026" # port the context broker listens on to update context
            - "IOTA_NORTH_PORT=4061"
            - "IOTA_REGISTRY_TYPE=mongodb" #Whether to hold IoT device info in memory or in a database
            - "IOTA_LOG_LEVEL=DEBUG" #The log level of the IoT Agent
            - "IOTA_TIMESTAMP=true"
            - "IOTA_MONGO_HOST=mongo-db" # The host name of ongoDB
            - "IOTA_MONGO_PORT=27017" # The port mongoDB is listening on
            - "IOTA_MONGO_DB=iotagentul" # The name of the database used in mongoDB
            - "IOTA_HTTP_PORT=9000" # The port used for device traffic over HTTP
            - "IOTA_PROVIDER_URL=http://localhost:4061"
```

Figura 5-30.Dockerfile Agente IoT.

En la imagen anterior, se muestra un fragmento del fichero de configuración que recoge las características para desplegar el agente. Características como la dirección de la base de datos utilizada para la persistencia de la información o del proveedor de contenido son necesarias para la interrelación del agente con la plataforma. Los puertos por donde el agente escucha y la url también son configuradas en este fichero.

Una vez confeccionado correctamente el fichero de configuración, bastaría con desplegar el contenedor apoyándose sobre este fichero.

El agente necesitara, una vez esté en funcionamiento, establecer una serie de servicios los cuales el agente va a publicar el contenido, y denotar los dispositivos que van a enviar información al agente, además de sus

características y servicios adheridos al dispositivo.

En primer lugar, definiremos los servicios que el agente va a dar servicio. Estos son tres: calidad, presión y consumo. Para definir estos servicios en el agente IoT, será necesario realizar una petición HTTP POST al agente definiendo varios parámetros:

- Fiware-Service: Nombre del servicio. Este parámetro viajara como cabecera en la petición HTTP. En nuestro caso, todos los servicios que despleguemos en el proyecto tendrán el mismo servicio Fiware: "aquasig".
- Fiware-ServicePath: Indica el nombre del subservicio dentro del marco del proyecto. Este parámetro, al igual que el anterior, viajara como cabecera en el mensaje. En el proyecto, este parámetro contendrá el tipo de dispositivo que envía la información: calidad, presión o consumo.
- Apikey: Sera la identificación que usaran los dispositivos IoT para identificarse con el nombre del servicio determinado. Como ejemplo, el Apikey con los que los dispositivos de calidad se identificaran ante el agente será "calidad-devices".
- Entity type: Sera el tipo de identidad que enviara el agente al proveedor de contenido. En el proyecto, este parámetro se identificará como "WaterStats".
- Resource: Denota la sub-dirección en la cual los dispositivos IoT van a proveerle el contenido al agente. En nuestro caso, este parámetro siempre será el mismo: "/iot/d". Este es el valor por defecto de este parámetro en la plataforma Fiware

A continuación, se muestra un ejemplo del script en bash que se ha realizado para registrar en los agentes IoT los servicios a los que van a dar soporte.

```
#!/bin/bash

curl -X POST -H "Fiware-Service: aquasig" -H "Fiware-ServicePath: /quality"
     -H "Content-Type: application/json" -H "Cache-Control: no-cache" -d '{
     "services": [
       {

         "apikey":   "calidad-devices",
         "entity_type": "WaterStats",
         "resource": "/iot/d"

       }
     ]
 }
 ' 'http://localhost:4061/iot/services'
```

Figura 5-31. Registro entidades calidad.

En segundo lugar, será necesario proveerle al agente el ID de los dispositivos de los que va a recibir información, junto con los atributos que va a recibir de estos. También se le describirá la forma en que recibirá los datos entre otras:

- Device_id: ID del dispositivo el cual va a mandar información al agente.
- Protocol: Protocolo que va a ser usado para el envío de la información. En nuestro caso viajara sobre http.
- Entity_name: Nombre del dispositivo el cual va a ser referenciado al proveedor de contenido.
- Entity_type: Tipo de entidad referenciada a la plataforma. En nuestro caso el tipo de entidad será "WaterStats".
- Timezone: Zona horaria usada por el dispositivo. Esto nos servirá para que el agente mande al proveedor de contenido una marca temporal con el instante en el cual recibe información de los dispositivos.
- Atributes: Lista que describe los atributos del dispositivo registrado. Cada atributo contendrá varios

campos:
    - Object_id: Sera el id con el que el dispositivo se referencie.
    - Name: Nombre referenciado por el Object_id que será mandado a la plataforma.
    - Type: Tipo de variable que recibe el agente.
- Static_attributes: Atributos estáticos del dispositivo. Son atributos que deben ser mandados a la plataforma, pero el agente no espera recibirlos del dispositivo. Cada atributo estático contendrá varios campos:
    - Name: Identificación del atributo hacia la plataforma.
    - Type: Tipo de variable que recibe el agente.
    - Value: Valor estático del atributo.

A continuación, se muestra un ejemplo de registro de dispositivo. En este caso, este script está preparado para registrar dispositivos de presión. En la comunicación entre el dispositivo y el agente, este último recibirá los atributos de presión y fecha de obtención. Además, el agente añadirá a estos atributos, el atributo estático "dispositivo".

```
#!/bin/bash

ID="$1"

curl -X POST -H "Fiware-Service: aquasig" -H "Fiware-ServicePath: /presure"
             -H "Content-Type: application/json" -H "Cache-Control: no-cache" -d '{
  "devices": [
    {
      "device_id":   "'"$ID"'",
      "protocol": "HTTP",
      "entity_name": "'"$ID"'",
      "entity_type": "WaterStats",
      "timezone":    "Europe/Madrid",
      "attributes": [
        {
          "object_id": "p",
          "name":      "presion",
          "type":      "number"
        },
        {
          "object_id": "t",
          "name":  "fecha_obtencion",
          "type":  "number"
        }
      ],
   "static_attributes": [
     {
       "name":  "dispositivo",
       "type":  "string",
       "value":  "'"$ID"'"
     }
   ]
```

Figura 5-32. Registro Dispositivo Presión.

Una vez hayamos registrado los servicios y los dispositivos en el agente correctamente, bastara con que los

dispositivos enviar peticiones HTTP POST con los atributos concatenados siguiendo el patrón key-value. Un ejemplo de la concatenación de los datos de presión puede ser: “p|10|t|120”.

5.4 Orion Context Broker

Se puede arrancar Orion Context Broker fácilmente gracias a Docker. Hay varias maneras de realizar el despliegue del Context Broker con Docker. En este caso, se ha desplegado el Broker usando Docker Compose ya que permite enlazar el contenedor del Context Broker con el de la base de datos MongoDB en pocos minutos y aislarlos los mismos entre sí. Para esto, bastaría con crear un fichero llamada “docker-compose.yaml” con el siguiente contenido:

```
mongo:
    image: mongo:3.4
    command: --nojournal
    network_mode: bridge
    hostname: bridge
    volumes:
      - /home/aquasig/fiware:/fiware
    ports:
      - "27017:27017"
orion:
    image: fiware/orion
    hostname: orion
    network_mode: bridge
    links:
      - mongo
    ports:
      - "1026:1026"
    command: -dbhost mongo
```

Figura 5-33. Docker Compose Orion Context Broker

En él se puede ver como se usan dos instancias y se describe la interconexión una con otra. Además, podemos apreciar a simple vista el mapeo del puerto 1026, el cual usa Orion para recibir peticiones.

Posteriormente usando la línea de comandos, bastaría con arrancar los contenedores: “sudo docker-compose up”. Con esto, en unos minutos, se iniciará el Context bróker y estará listo para ser usado y escuchando peticiones por su puerto predeterminado.

Lo que se ha hecho con este método es descargar las imágenes de Orion Context Broker y de MongoDB del repositorio publico llamado Docker Hub [https://hub.docker.com]. Después se crean los dos contenedores basados en estas imágenes y se realiza la conexión entre ambos.

5.5 Cygnus

El rol del conector en la arquitectura del proyecto AQUASIG, es la de actuar como intermediador entre el proveedor de contenido, el context Broker en nuestro caso, y un sistema de almacenamiento de información.

La configuración de Cygnus se realiza a través del fichero de configuración agent.conf. En este fichero, se detallan las características de las fuentes, de los canales y de los sumideros. En nuestro caso se describe una fuente HTTP, un canal de tipo MongoDB y un sumidero de tipo MongoDB.

```
##############################################################
cygnus-ngsi.sources = http-source
cygnus-ngsi.sinks = mongo-sink
cygnus-ngsi.channels = mongo-channel
##############################################################
```

Figura 5-34. Descripción canal.

A continuación, se detallan las características que tiene la fuente:

```
##############################################################
cygnus-ngsi.sources.http-source.type =
org.apache.flume.source.http.HTTPSource
cygnus-ngsi.sources.http-source.channels = mongo-channel
cygnus-ngsi.sources.http-source.port = 5050
```

Figura 5-35. Características Fuente (I).

```
cygnus-ngsi.sources.http-source.handler =
com.telefonica.iot.cygnus.handlers.NGSIRestHandler
cygnus-ngsi.sources.http-source.handler.notification_target =
/notify
cygnus-ngsi.sources.http-source.handler.default_service =
default
cygnus-ngsi.sources.http-source.handler.default_service_path =
/
cygnus-ngsi.sources.http-source.interceptors =   nmi ts
cygnus-ngsi.sources.http-source.interceptors.ts.type =
timestamp
cygnus-ngsi.sources.http-
source.interceptors.nmi.type=com.telefonica.iot.cygnus.interce
ptors.NGSINameMappingsInterceptor$Builder
cygnus-ngsi.sources.http-
source.interceptors.nmi.name_mappings_conf_file =
/home/aquasig/files/name_mappings.conf
##############################################################
```

Figura 5-36. Características Fuente (II).

Se puede ver que en el fichero se tiene que configurar el puerto por donde se reciben los mensajes, la cabecera que tiene el paquete y posteriormente se configuran unos interceptores de nombres que se encargaran de hacer un mapeo de nombres para escribir en la base de datos mongo con un nombre de entidad determinado, etc. Estas reglas de mapeado se presentan más adelante en el documento.

A continuación, se describen las características que se configuran para el canal. Se establecen características como la capacidad del canal y la capacidad de transacciones.

```
##############################################################
cygnus-ngsi.channels.mongo-channel.type =
com.telefonica.iot.cygnus.channels.CygnusMemoryChannel
cygnus-ngsi.channels.mongo-channel.capacity = 1000
cygnus-ngsi.channels.mongo-channel.transactionCapacity = 100
##############################################################
```

Figura 5-37. Características Canal.

Por último, se configuran las características de los sumideros, en nuestro caso de MongoDB, donde se

configura el modelo de datos de la base de datos, la dirección de la fuente, las credenciales de acceso, etc.

```
################################################################
cygnus-ngsi.sinks.mongo-sink.type =
com.telefonica.iot.cygnus.sinks.NGSIMongoSink
cygnus-ngsi.sinks.mongo-sink.channel = mongo-channel
cygnus-ngsi.sinks.mongo-sink.enable_name_mappings = true
cygnus-ngsi.sinks.mongo-sink.data_model = dm-by-service-path
cygnus-ngsi.sinks.mongo-sink.mongo_hosts =
mongodb.aquasig.isoin.es:27030
cygnus-ngsi.sinks.mongo-sink.mongo_username =user_rw
cygnus-ngsi.sinks.mongo-sink.mongo_password =[redacted]
cygnus-ngsi.sinks.mongo-sink.db_prefix = aqua
cygnus-ngsi.sinks.mongo-sink.collection_prefix = sim_
################################################################
```

Figura 5-38. Características Sumidero.

El mapeo de nombres es una funcionalidad avanzada de Cygnus disponible para todos los sumideros. Esta funcionalidad, permite cambiar ciertos atributos del mensaje de notificación enviado por el bróker pudiendo manipularlo. Los atributos que puede cambiar son el servicio, el path del servicio, el nombre de la entidad, el tipo de entidad, nombres de atributos y tipos de atributos.

El mapeo de nombres se puede configurar a partir de un fichero JSON escribiendo el antiguo valor del atributo y el nuevo de la siguiente manera:

```
################################################################
{
    "serviceMappings": [
        {
            "originalService": "myservice1",
            "newService": "new_myservice1",
            "servicePathMappings": [
                {
                    "originalServicePath": "/myservicepath1",
                    "newServicePath": "/new_myservicepath1",
                    "entityMappings": [
                        {
                            "originalEntityId": "myentityid1",
                            "originalEntityType": "myentitytype1",
                            "newEntityId": "new_myentityid1",
                            "newEntityType": "new_myentitytype1",
                            "attributeMappings": [
                                {
                                "originalAttributeName":
"myattributename1",
                                "originalAttributeType":
"myattributetype1",
                                "newAttributeName":
"new_myattributename1",
                                "newAttributeType":
"new_myattributetype1"
                                },
                                ...
                            ]
                        },
                        ...
                    ]
                },
                ...
            ]
        },
        ...
    ]
}
################################################################
```

Figura 5-39. Configuración Mapeo de nombres.

Cuando una notificación es mandada a Cygnus, un interceptor recoge el cuerpo de los eventos, e itera en los diferentes ficheros de mapeo de nombres de configuración hasta coincidir con alguna para crear una versión mapeada de la notificación original. Posteriormente las dos notificaciones (la mapeada y la original) es mandada al canal, para que posteriormente, la configuración del sumidero decida si se manda la versión mapeada o la original. Esto se puede configurar habilitando el mapeo de nombres en el sumidero.

En la plataforma, se usará este mapeo de nombre para adecuar el nombre de las entidades a las tablas de la base de datos donde se guardarán dichas entidades.

5.6 Consumidores de Información

Los datos obtenidos mediante CYGNUS son almacenados en una base de datos MongoDB, una base de datos de carácter no relacional, caracterizada por su agilidad y escalabilidad.

Este tipo de bases de datos, permiten cambiar los esquemas de datos de manera rápida cuando las aplicaciones evoluciona, aportando además un lenguaje completo de búsquedas y consistencia estricta.

La replicación nativa y la tolerancia a fallos automática ofrece fiabilidad al sistema AQUASIG y flexibilidad operativa.

La concentración de la información en la base de datos, permite confeccionar un análisis de dicha información a partir de técnicas de minado de datos, que facilitan la elaboración de informes de carácter histórico, la identificación de patrones y el cálculo de posibles escenarios futuros.

Los desarrollos que utilizan las bases de datos donde se almacena toda la informacion extraida de los dispositivos productores de información queda fuera del marco del actual trabajo. Por esto, a continuación, se hará una explicación meramente descriptiva para exponer el uso que se le pueden dar a dicha información.

En este caso, toda la información recogida en esta base de datos nos sirve para obtener información de consumo y calidad de agua para que dicha información sea usada tanto para usuarios de viviendas domesticas como de distribuidoras de agua.

En lo que respecta a la parte del cliente, existe un motor web el cual se encarga de mostrarle la información en tiempo real de los parámetros de calidad del agua que le llega al mismo. Además, el cliente podrá comprobar el consumo del mes actual en el que se encuentra, pudiéndolo comparar con los resultados obtenidos mediante gráficas.

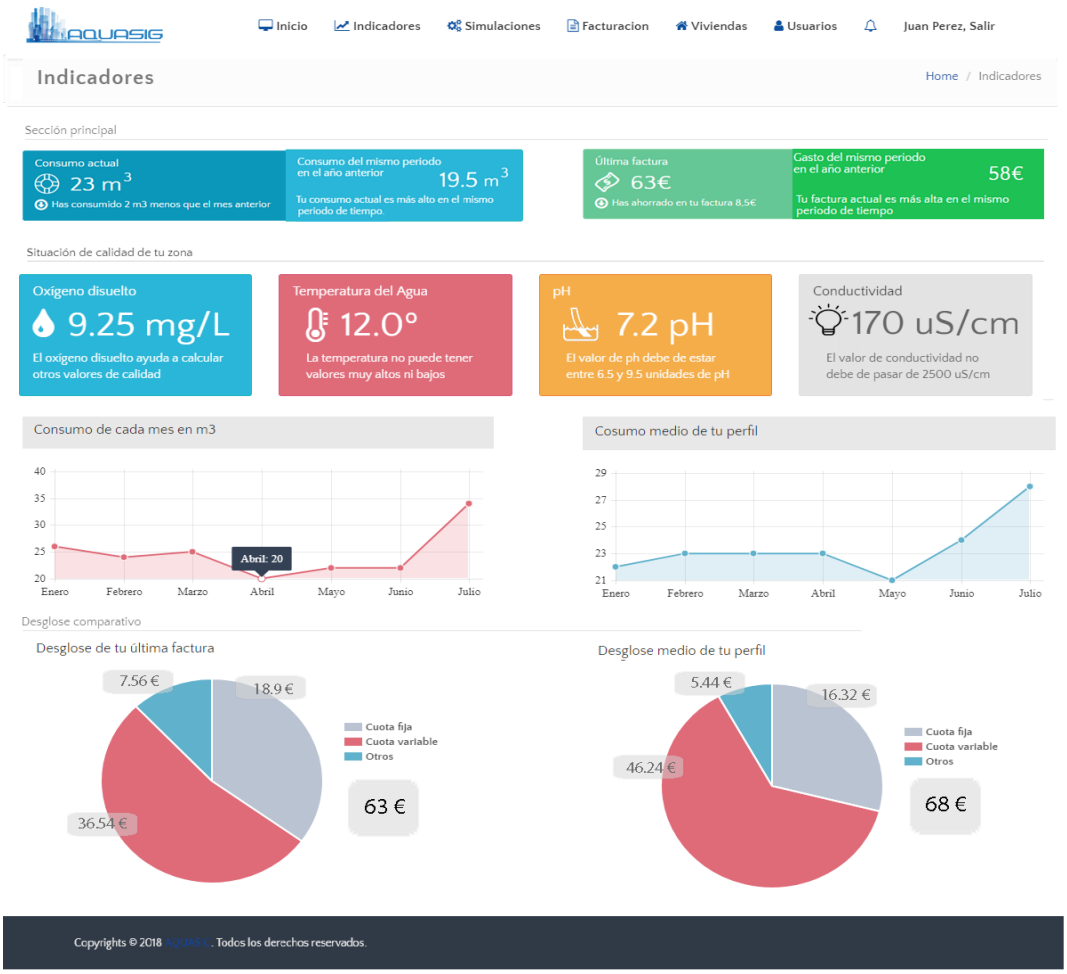


Figura 5-40. Interfaz Web Cliente.

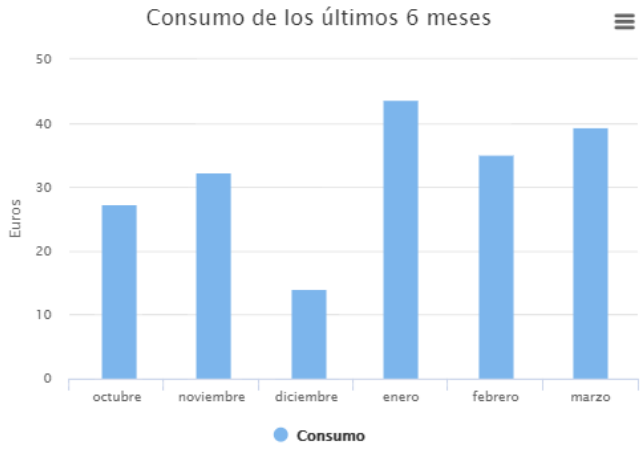


Figura 5-41. Consumo Cliente. Conversión en Euros.

En las figuras anteriores se ve como, además del consumo actual y los parámetros de calidad que el usuario percibe, a partir del consumo puede comprobar el precio de su factura obtenida mediante el precio del agua en la zona y las cuotas variables y fijas que esta tiene. Además, se podrá ver el consumo de meses anteriores para que el cliente pueda identificar patrones en ellos.

En la siguiente imagen, se ve como la plataforma web te permite comparar tu consumo de agua con el de años anteriores para así poder identificar malos usos del agua y poder actuar sobre ellos.

La información de consumo de agua junto con la adición de ciertos consejos de uso de agua, le facilitará al cliente la gestión de este recurso tan escaso y reducir sustancialmente el gasto mensual por este concepto. Ademas, gracias a la información de calidad de agua, el cliente tendrá una mayor transparencia acerca de lo que le esta consumiendo en cada instante.

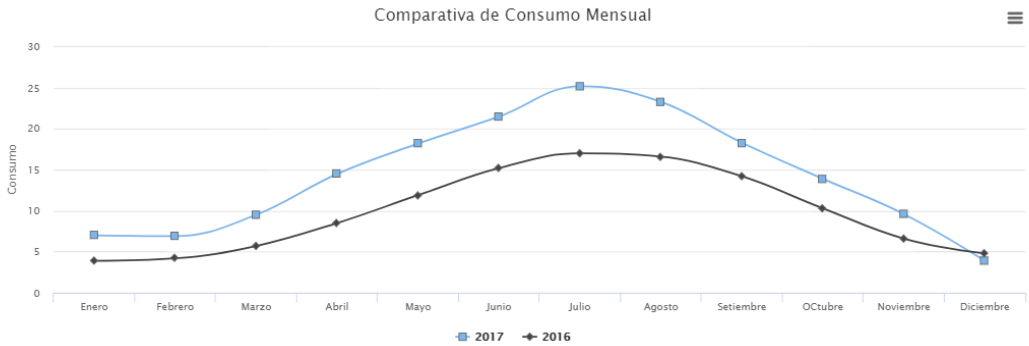


Figura 5-42. Comparativa de consumo de cliente.

En cuanto a la parte web que involucra a la distribuidora, esta cuenta con sistemas propios de alertas los cuales se encargan de comprobar el estado del agua a través de la información obtenida de los dispositivos, y se encarga de enviar notificaciones a los responsables para su actuación. Este sistema de alertas, cuenta con diferentes estados en función del nivel de riesgo que conlleva para notificar y actuar de manera proactiva en función del mismo.

Además del sistema de alertas, la parte de la distribuidora cuenta con una interfaz web la cual sirve de apoyo a los gestores de la misma para comprobar en cada instante el estado del agua en tiempo real. En él, se podrá escoger el dispositivo desplegado el cual se quiere comprobar su estado y se mostraran en él los datos obtenidos según la fecha de obtención de los mismos. En la siguiente figura se puede ver la interfaz web diseñada para la distribuidora.

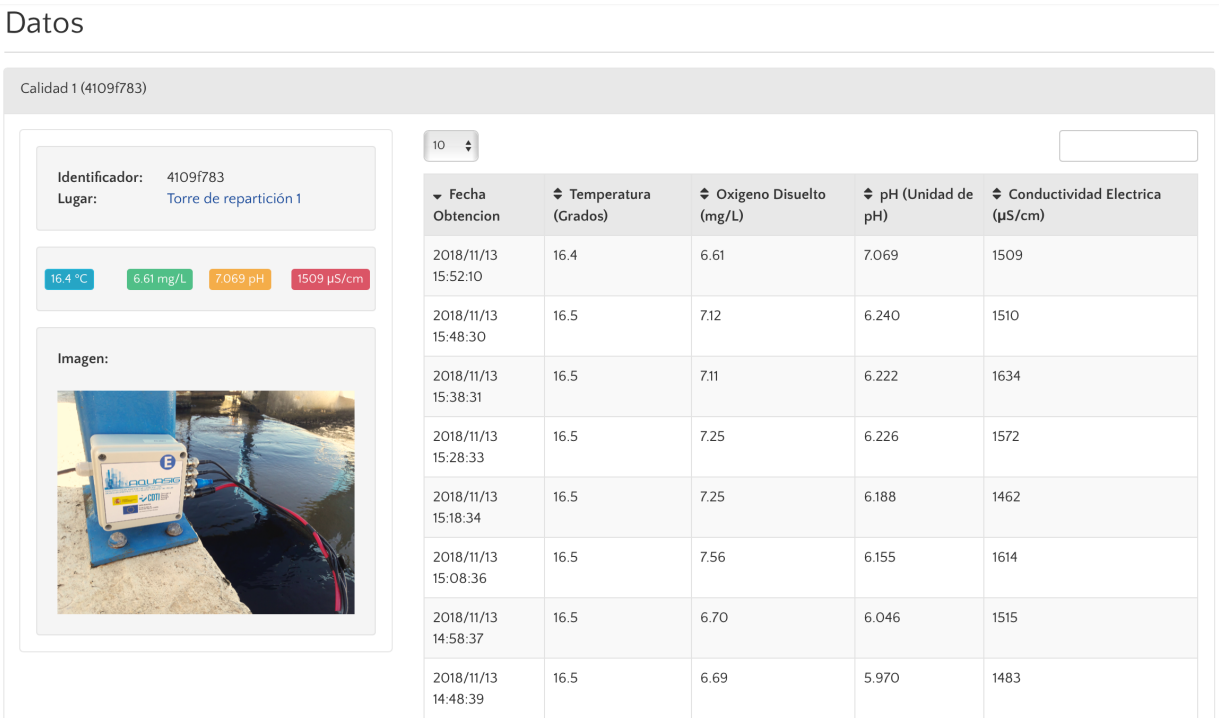


Figura 5-43. Interfaz Web Distribuidora

Ademas, para que los gerentes puedan contemplar los datos de cada parametro de calidad con mayor calidad, se pueden ver los mismos en una serie de graficas donde se podra tener otra perspectivas de los datos y poder identificar pendientes anomalas o posibles valores fuera de lo normal.

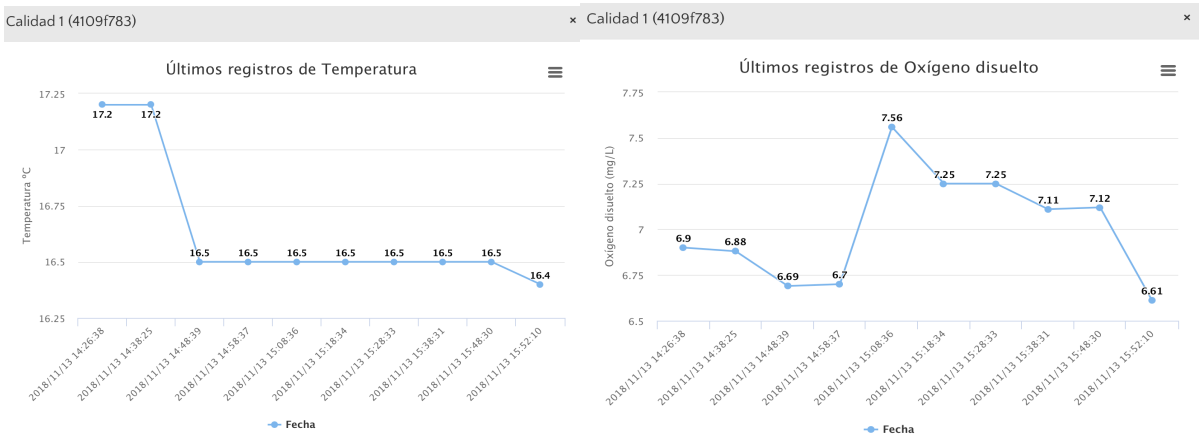


Figura 5-44. Graficas parámetros de calidad (I).

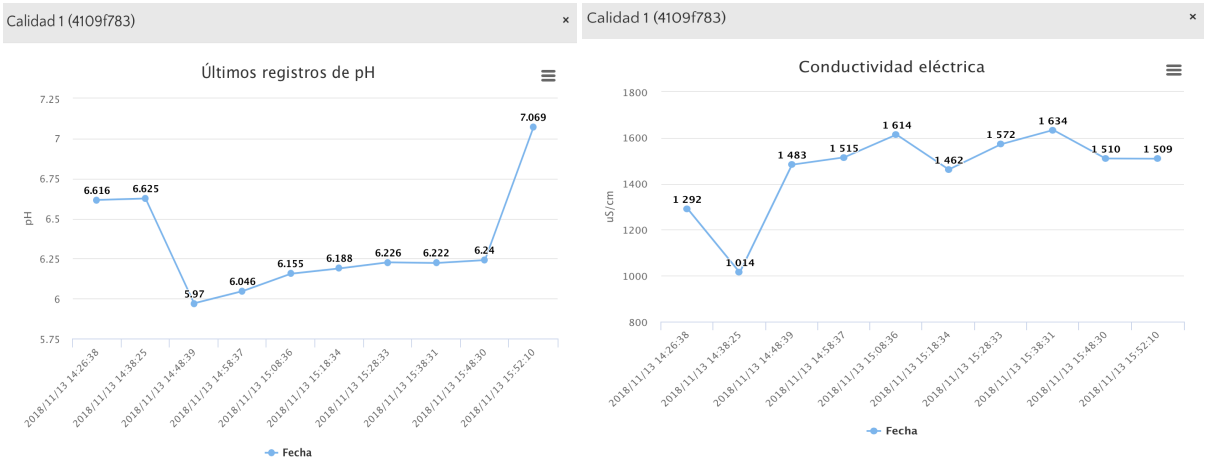


Figura 5-45. Graficas parámetros de calidad (II)

6 CONCLUSIONES

Cuantos más datos tengamos, más posibilidades tenemos de ahogarnos en ellos.

- Nassim Taleb -

Tras haber investigado y recabado información acerca del objeto de nuestro estudio para poder llevar a cabo el desarrollo del presente proyecto, y desarrollado el sistema basándose en los elementos que mejor encajaban en el marco del mismo, el siguiente paso es estudiar el resultado y expresar las conclusiones obtenidas.

Comenzaremos por fijarnos en los objetivos iniciales para sacar unas primeras conclusiones. Por tanto, podemos afirmar que se han cumplido los principales objetivos del proyecto:

- Se ha conseguido desarrollar varios dispositivos que se encargan de la adquisición de las medidas necesarias para el proyecto. Dichas medidas se pueden englobar como medidas de parámetros de calidad de agua, medidas de consumo de agua, medidas de consumo eléctrico y medidas de presión.
- Se han implementado unas placas que se ajustan perfectamente a las necesidades del proyecto, para actuar como cerebro del sistema de adquisición de datos y contando con los puertos suficientes para su posterior integración con el resto de los componentes de la arquitectura.
- Se ha implementado en las interfaces de medida una etapa de comunicaciones que cuenta con un protocolo que encaja con las especificaciones requeridas en el proyecto, además de ser abierto y pudiendo ser usado sobre las bandas ISM.
- Los protocolos de nivel de aplicación usados para la comunicación de los sensores son protocolos estándares en la industria como HTTP, protocolos de tipado de datos como JSON y protocolos de transporte como UDP.
- Se ha desplegado satisfactoriamente los equipos concentradores de datos, los cuales son los elementos finales de la red de área local. Además, se le ha dotado al concentrador con una conexión tipo GSM para el envío de toda información referente a las interfaces de medida hacia la plataforma
- Se ha diseñado un sistema avanzado para la óptima captura, normalización y tratamiento de información en tiempo real para el procesado de datos hacia la plataforma AQUASIG procedentes del sistema de adquisición de la medida desarrollados.
- El software es configurable y reconfigurable a lo largo de todo su ciclo de vida, de manera al modificar sus parámetros se adapta y actualiza su comportamiento para satisfacer las nuevas restricciones.
- La arquitectura diseñada cumple con los requisitos deseados para su desarrollo, para lo que se usan tecnologías Open Source ampliamente reconocidas, innovadoras y que ofrecen altas capacidades de integración, escalabilidad y flexibilidad con los sistemas existentes en las empresas distribuidoras de agua urbana, además de incorporar elementos de confiabilidad, seguridad y persistencia de la información, otorgando robustez al sistema.
- Al desplegar los componentes de la plataforma en el marco de la gestión de datos de agua y de su

posterior procesado de estos a través de el motor web, hemos conseguido aumentar la eficiencia del sistema de distribución, pudiendo detectar anomalías en el uso del agua y en la calidad de la misma que puede llegar al cliente provocando posibles deterioros de la salud humana.

- Además, se pueden obtener patrones de comportamiento, dependencias entre parámetros, tendencias, etc, que pueden ayudar al cliente a actuar sobre ellos para la reducción del consumo.

REFERENCIAS

[1] [En línea]. Available: http://www.internet-of-things-research.eu/abo ut_iot.htm.

[2] «What exactly is the “Internet of Things”?,» [En línea]. Available: http://postscapes.com/what-e xactly-is-the-internet-of-things-infographic.

[3] «ARPANET,» [En línea]. Available: https://www.britannica.com/topic/ARPANET.

[4] «The Internet Toaster,» [En línea]. Available: https://www.livinginternet.com/i/ia_myths_toast.htm.

[5] [En línea]. Available: http://www.ieee802.org/15/pub/TG4.html.

[6] [En línea]. Available: https://www.bluetooth.com/bluetooth-technology/radio-versions.

[7] [En línea]. Available: https://www.iotforall.com/bluetooth-5-iot/:.

[8] «WiFi Alliance,» [En línea]. Available: https://www.wi-fi.org.

[9] [En línea]. Available: https://www.everythingrf.com/community/ism-frequency-bands.

[10] [En línea]. Available: https://www.networkworld.com/article/3196191/lan-wan/wifi-s-evolving-role-in-iot.html.

[11] [En línea]. Available: http://www.3gpp.org.

[12] [En línea]. Available: https://www.qualcomm.com/invention/5g/internet-of-things.

[13] [En línea]. Available: http://www.ieee802.org/3/.

[14] [En línea]. Available: https://searchdatacenter.techtarget.com/es/definicion/Ethernet.

[15] [En línea]. Available: http://www.3gpp.org/images/PDF/R13_IOT_rev3.pdf.

[16] [En línea]. Available: https://www.huawei.com/minisite/iot/img/nb_iot_whitepaper_en.pdf.

[17] [En línea]. Available: http://elb105.com/empezando-a-trabajar-con-nb-iot/.

[18] [En línea]. Available: https://tools.ietf.org/html/rfc791#page-1.

[19] [En línea]. Available: https://www.ietf.org/rfc/rfc2460.txt.

[20] [En línea]. Available: https://tools.ietf.org/html/rfc2663.

[21] [En línea]. Available: https://www.solvetic.com/page/recopilaciones/s/internet/caracteristicas-diferencias-

protocolo-internet-ipv4-ipv6.

[22] [En línea]. Available: https://www.zigbee.org/download/standards-zigbee-specification/.

[23] [En línea]. Available: https://www.arrow.com/es-mx/research-and-events/articles/wireless-connectivity-for-the-internet-of-things-one-size-does-not-fit-all .

[24] [En línea]. Available: https://slideplayer.com/slide/5983758/.

[25] [En línea]. Available: https://tools.ietf.org/html/rfc6282.

[26] [En línea]. Available: http://www.libelium.com/products/waspmote-mote-runner-6lowpan/.

[27] [En línea]. Available: https://tools.ietf.org/html/rfc768.

[28] [En línea]. Available: https://tools.ietf.org/html/rfc793.

[29] [En línea]. Available: https://www.ecured.cu/Protocolo_de_Control_de_Transmisi%C3%B3n.

[30] [En línea]. Available: https://es.ccm.net/contents/281-protocolo-tcp.

[31] [En línea]. Available: http://mqtt.org.

[32] [En línea]. Available: https://www.ibm.com/developerworks/ssa/library/iot-mqtt-why-good-for-iot/index.html .

[33] [En línea]. Available: https://xmpp.org.

[34] [En línea]. Available: https://xmpp.org/extensions/.

[35] [En línea]. Available: https://www.omgwiki.org/dds/].

[36] [En línea]. Available: http://www.rfwireless-world.com/Terminology/DDS-protocol-architecture.html.

[37] [En línea]. Available: http://www.rfwireless-world.com/Terminology/DDS-protocol-architecture.html.

[38] [En línea]. Available: https://www.amqp.org].

[39] [En línea]. Available: https://www.electronicdesign.com/iot/understanding-protocols-behind-internet-things.

[40] [En línea]. Available: https://tools.ietf.org/html/rfc6347.

[41] [En línea]. Available: https://www.vocal.com/networking/datagram-transport-layer-security-dtls/ .

[42] [En línea]. Available: https://tools.ietf.org/html/rfc2616 .

[43] [En línea]. Available: https://concepto.de/http/.

[44] [En línea]. Available: https://tools.ietf.org/html/rfc7252 .

[45] [En línea]. Available: [https://iot.telefonica.com/blog/2016/09/es-fiware-estandar-iot.

[46] [En línea]. Available: http://toools.es/plataforma-fiware/.

[47] [En línea]. Available: https://aws.amazon.com/es/iot/.

[48] [En línea]. Available: https://cloud.google.com/solutions/iot/ .

[49] [En línea]. Available: https://azure.microsoft.com/es-es/overview/iot/ .

[50] [En línea]. Available: https://www.ibm.com/internet-of-things.

[51] [En línea]. Available: https://www.FiWare.org/.

[52] [En línea]. Available: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/NGSI-9/NGSI-10_information_model .

[53] [En línea]. Available: https://fiware-orion.readthedocs.io/en/master/user/federation/index.html.

[54] [En línea]. Available: https://fiware-iotagent-json.readthedocs.io/en/latest/.

[55] [En línea]. Available: https://www.omaspecworks.org/what-is-oma-specworks/iot/lightweight-m2m-lwm2m/.

[56] [En línea]. Available: https://fiware-iotagent-ul.readthedocs.io/en/latest/.

[57] [En línea]. Available: https://fiware-lorawan.readthedocs.io/en/latest/ .

[58] [En línea]. Available: https://www.thethingsnetwork.org/docs/lorawan/] .

[59] [En línea]. Available: http://flume.apache.org.

[60] [En línea]. Available: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html.

[61] [En línea]. Available: https://www.mysql.com/.

[62] [En línea]. Available: http://ckan.org/.

[63] [En línea]. Available: https://www.mongodb.org/.

[64] [En línea]. Available: https://github.com/telefonicaid/IoT-STH.

[65] [En línea]. Available: http://kafka.apache.org/.

[66] [En línea]. Available: https://aws.amazon.com/dynamodb/.

[67] [En línea]. Available: http://www.postgresql.org/.

[68] [En línea]. Available: https://carto.com.

[69] [En línea]. Available: https://flume.apache.org.

[70] [En línea]. Available: https://wirecloud.readthedocs.io/en/stable/ .

[71] [En línea]. Available: https://knowage.readthedocs.io/en/latest/.

[72] [En línea]. Available: https://kurento.readthedocs.io/en/stable/.

[73] [En línea]. Available: https://fiware-cosmos-flink.readthedocs.io/en/latest/ .

[74] [En línea]. Available: https://fogflow.readthedocs.io/en/latest/.

[75] [En línea]. Available: https://www.docker.com.

[76] [En línea]. Available: http://www.ti.com/tool/ENERGIA.

[77] [En línea]. Available: https://www.zigbee.org/zigbee-for-developers/zigbee-3-0/.

[78] [En línea]. Available: https://www.digi.com/products/xbee-rf-solutions/2-4-ghz-modules/xbee-zigbee.

[79] [En línea]. Available: https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu.

GLOSARIO