

Proyecto Fin de Carrera

Ingeniería de Telecomunicación

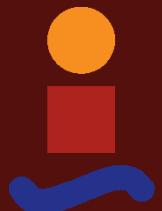
Control de polución en Smart Cities mediante aplicaciones en FIWARE

Autor: Miguel Ángel Caño Rojano

Tutor: José Ramón Cerquides Bueno

Dep. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2015



Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Control de polución en Smart Cities mediante aplicaciones en FIWARE

Autor:
Miguel Ángel Caño Rojano

Tutor:
José Ramón Cerquides Bueno
Profesor Titular

Dep. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2015

Proyecto Fin de Carrera: Control de polución en Smart Cities mediante aplicaciones en FIWARE

Autor: Miguel Ángel Caño Rojano
Tutor: José Ramón Cerquides Bueno

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Never thought I would write these lines, because I have always been a bit skeptical about myself. On one occasion, my younger brother, very idealistic, told someone that his older brother was very intelligent and would be an engineer. After the big setbacks and academic massacres in the first year, I asked him to change his vision of reality, but I did not want to deceive the people I care about. So I continued fighting. For him.

This academic stage is filled with people, experiences and teachings, but what really defines it, from the first embrace to the last, is sacrifice. I know that my grandparents will be wherever they are, that they too have been my pillars. For them, for what I found and for what I lost.

This section would not be complete if I did not dedicate words to the two best engineers I know: my parents. The person I am, result of a life project that began 23 years ago, developed following similar phases to an engineering project; design, analysis and educational construction, and used rudimentary tools (and not very efficient) like effort, delivery, care or passion. My allies, shield and sword, and the two best human beings I have never known.

They say that suffering or sacrifice is more bearable when shared, and also that it is the fault of the companions who are here today. Thank you for trusting me to be your voice during that time.

To Telefónica I+D, for the support and attention provided. To the Malagueña TOPdigital company, for the loan of equipment and its welcome at the beginning of my professional life.

Finally, and no less important, to thank José Ramón Cerquides for his invaluable support, advice and professionalism. With his admirable gift for teaching, from a distance I have returned to feel the same devotion to this project that he showed in his teaching over the years. Without a doubt, one of the best resources available at this school.

Resumen

Las ciudades se han convertido en una pieza fundamental del desarrollo socioeconómico al concentrarse la población y actividad económica en los núcleos urbanos. Se prevee que más del 70% de la población mundial vivirá en las ciudades hacia el año 2050. Este incremento en la concentración de la población traslada a las ciudades los paradigmas de sostenibilidad, eficiencia y progreso de la sociedad, propugnando un cambio de modelo bajo el que se enmarca el concepto de Smart City. Este término aglutina de forma integrada las iniciativas y estrategias orientadas a mejorar la calidad de vida, la sostenibilidad y la gestión eficiente de los servicios, innovando en materiales, recursos y modelos usando la tecnología.

Este proyecto presenta una solución conceptual en el ámbito de la sostenibilidad medioambiental. Mediante un despliegue de una red de sensores y una serie de aplicaciones desarrolladas sobre la plataforma europea FIWARE, las administraciones públicas y organismos interesados podrán obtener una caracterización de la concentración de diferentes tipos de gases que afectan a la polución atmosférica. Esta monitorización podría constituir el punto de partida para tomar decisiones en pos de mejorar la calidad del aire en la ciudad.

Abstract

Cities have become a cornerstone in socioeconomic development due to the increasing concentration of people in urban areas. More than 70 % of people will live in cities in 2050. This increase leads changes in sustainability, efficiency and social progress contextualised under the concept Smart City. This concept gathers initiatives and strategies focused on improving citizens quality of life, sustainability and efficient management of resources, innovating in materials and models with technology as a tool.

This project presents a conceptual solution in the area of environmental sustainability through a deployment of a pollution sensors network. These sensors provide readings from different typologies in order to monitor values of atmospheric gases. Monitoring, management and alerts are performed using FIWARE, an european platform aimed to deliver cloud services.

Índice

<i>Resumen</i>	III
<i>Abstract</i>	V
<i>Introducción</i>	1
1. Fi-Ware	3
1.1. Contexto actual	3
1.1.1. Las perspectivas del mercado	3
1.1.2. Fi-Ware como parte del programa FI-PPP	4
1.1.3. Objetivos del proyecto	4
1.1.4. Áreas técnicas definidas	5
1.1.5. Fi-Ware Testbed y Fi-Lab	5
1.2. Arquitectura	6
1.2.1. Arquitectura general de referencia	6
1.2.2. Especificaciones NGSI en Fi-Ware	6
1.2.3. Arquitectura en <i>Data/Context Management</i>	8
1.2.4. Dependencias e interacción entre GEs	9
1.2.5. Orion Context Broker	10
2. Elección de dispositivos	11
2.1. Restricciones generales	11
2.2. Factores involucrados	12
2.2.1. Protección ante agentes externos	12
2.2.2. Comunicación	13
Protocolos de comunicación inalámbrica	13
Elementos de una red de sensores	14
El protocolo ZigBee	15
El protocolo Bluetooth Low Energy	16
Comparativa final y tecnología escogida	16
2.2.3. Electrónica interna	17
Hardware de fabricante	17
Electrónica open source: Arduino	19
2.3. Elección final: Waspmove	21
2.3.1. Core: especificaciones técnicas	21
2.3.2. Gas Sensor Board	22
2.3.3. Libelium Smart Environment	23
2.4. Elección del gateway	25
3. Desarrollo del software para dispositivos de la red	27
3.1. Primeros pasos con Waspmove Smart Environment	27
3.1.1. IDE	27
3.1.2. Lenguaje de programación	28
3.1.3. Calibración y puesta a punto de los sensores de polución	29

Consideraciones generales	29
Aspectos clave	30
3.2. Nodos sensores	33
3.2.1. Comportamiento básico del sistema	33
3.2.2. Características destacables de la implementación	33
3.3. Gateways	33
3.3.1. Comportamiento básico del sistema	33
3.4. Conexión con Fi-Ware y almacenamiento en BBDD	34
3.4.1. Comportamiento del software	34
3.4.2. Lenguajes posibles para la implementación	34
3.4.3. Interacción con Orion Context Broker	34
Creando y consultando entidades	35
Montando el payload para inserción de dispositivos de la red	37
3.4.4. Inserción de medidas en una base de datos MySQL	38
4. Desarrollo de aplicaciones en Fi-Ware	41
4.1. Apartados de Fi-Lab	41
4.1.1. Cloud	41
Desplegando una instancia propia de Orion Context Broker	41
4.1.2. Mashup	43
Creando aplicaciones masivas: <i>wiring</i>	44
Creando widgets y operadores	44
4.1.3. Store	45
4.1.4. Account	45
4.1.5. Data	46
4.2. Aplicaciones desarrolladas	47
4.2.1. Mapa	47
4.2.2. Histórico de lecturas	48
4.2.3. Inspector de sensores	49
4.2.4. Alertas y notificaciones	49
<i>Conclusiones</i>	53
Apéndice A. Datasheets	55
A.1. Algunos sensores de interés: ozono (O3).	56
A.2. Algunos sensores de interés: dióxido de nitrógeno (NO2).	59
A.3. Wasp mote	62
A.4. XBee modules	65
A.5. Modulo GSM/GPRS Wasp mote: SIM900	66
A.6. Arduino GSM Shield: Quectel M10	68
Apéndice B. Código fuente de interés	71
<i>Índice de Figuras</i>	73
<i>Índice de Tablas</i>	75
<i>Bibliografía</i>	77

Introducción

Cada día son más las administraciones y organismos públicos que se suben al tren *Smart City*. Si bien no está demasiado claro actualmente qué necesita una ciudad para convertirse en *Smart*, la pretensión actual es la de mejorar la calidad de vida de los ciudadanos y obtener un mayor control sobre el impacto que éste tiene sobre su hábitat. Con la tecnología existente, se abre un enorme abanico de posibilidades para gestionar de manera eficiente los recursos de la ciudad. Si bien los conceptos que aparecen resultan novedosos, lo que se persigue no es más que una evolución de la ciudad actual hacia su versión parametrizada, tipificada y gestionada por el organismo dedicado al efecto, que tomará decisiones en cuanto a eficiencia y sostenibilidad.

El presente proyecto trata de abordar una de las posibles verticales en que podría dividirse el término *Smart City*, en este caso en el ámbito de la sostenibilidad medioambiental. Monitorizar la calidad del aire y las concentraciones de un cierto tipo de gas en una zona de la ciudad podría tenerse en cuenta para re-estructurar el tráfico o colocar instrumentos para mejorar la calidad del aire. En general, emprender algún tipo de iniciativa para mejorar la sostenibilidad de la ciudad. Esta solución abarca tanto la elección de los dispositivos adecuados para caracterizar concentraciones y el modelo de red a establecer como el desarrollo de aplicaciones para monitorización sobre la plataforma europea Fi-Ware.

Actualmente no se tiene constancia de proyectos similares por lo que, si bien algunas ciudades ya trabajan con este tipo de dispositivos, resulta innovadora la forma en que se ofrece la información y la administración de los sistemas. La plataforma Fi-Ware aún se encuentra en estado *beta* y, si bien existen desarrollos aislados y no demasiado concluyentes, lo cierto es que la infraestructura es mínima, por lo que los desarrollos realizados durante este proyecto pueden constituir la base de otros venideros. La principal ventaja que aporta Fi-Ware, en detrimento de plataformas cloud como Amazon Web Services o servicios de Google, es el carácter *open-source* de los componentes, el almacenamiento gratuito y la financiación que se está ofreciendo desde Europa, a través de una red de aceleradoras, para proyectos desarrollados sobre Fi-Ware. Existen plataformas puramente dedicadas a *Internet of Things* como Xively, aunque el control y las capacidades que ofrecen son limitadas.

Este documento se ha estructurado siguiendo un orden lógico, tratando de aportar conocimiento sobre lo que se va a discutir de manera clara, concisa y coherente. Si bien existen infinitas soluciones para implementar el modelo que tenemos en mente, esta solución se argumenta siguiendo máximas como la minimización del coste de implementación y la construcción de aplicaciones genéricas que puedan ser parte de otras verticales. La estructura en capítulos queda reflejada de la siguiente forma:

- **Capítulo 1: Fi-Ware.** Se aporta una visión general del proyecto Fi-Ware, abordando el alcance y la situación actual. La arquitectura de la plataforma final aparece expuesta aquí y nos detendremos en aquellos componentes que resulten de mayor utilidad a nuestra pretensión.
- **Capítulo 2: Elección de dispositivos.** Se analizarán las diferentes soluciones del mercado actual, restringiendo la elección a especificaciones propias y factores externos.
- **Capítulo 3: Desarrollo del software para dispositivos de la red.** Se expone el comportamiento que se espera de los dispositivos a nivel software, tratando la forma en que estos quedan conectados con la plataforma.

- **Capítulo 4: Desarrollo de aplicaciones en Fi-Ware.** Si bien a nivel documental es el de menor extensión, es sin duda el grueso del proyecto. Se presenta la tecnología en que se programan las aplicaciones, lo que se pretende conseguir y el resultado final.

Tras esta documentación se aportan algunas conclusiones y líneas futuras de investigación a las que podrían dar pie este proyecto.

1 Fi-Ware

1.1 Contexto actual

En los últimos años, se han podido reconocer ciertas tendencias en el panorama de las TIC que apuntan al comienzo de una nueva evolución de Internet. La primera de ellas se identifica en la continua industrialización de las tecnologías de la información en forma de plataformas de *cloud computing* y de prestación de servicios abiertos. Se espera que la naturaleza, hasta ahora bajo demanda y con un cierto coste de estos modelos, cambie la manera en que se proveen las infraestructuras de IT para ser vendidas como un servicio. Por otro lado, la instauración de nuevas tecnologías de comunicación inalámbrica como LTE, o la desaparición de las redes de cobre en detrimento de las redes FTTH, posibilitará un mayor ancho de banda y capacidad de red, lo cual constituye una excelente vía para crear aplicaciones que hagan uso de estas redes y que hasta ahora podría haber resultado tecnológicamente inviable construir.

En esta línea, podría unirse además la virtualización de redes de operadores clásicos y el auge en que se encuentra el llamado Internet de las Cosas (*Internet of Things*): la visión de conectar dispositivos y sensores a Internet para ofrecer información valiosa acerca del contexto en que se encuentren. El almacenamiento de esta información podría dar pie a un posterior análisis y procesamiento para ofrecer nuevas aplicaciones y servicios de automatización de procesos o control en diferentes ámbitos.

Las tendencias mencionadas constituyen el caldo de cultivo para lo que se conoce como el Internet del Futuro (*Future Internet*).

1.1.1 Las perspectivas del mercado

El Internet del Futuro traerá consigo el potencial necesario para abrir nuevas vías de negocio y aplicaciones emergentes en materia de salud, telecomunicaciones, *e-commerce* o *e-government*. En este sentido, los principales actores en este ecosistema poseen algunas demandas clave para que esta nueva tecnología cumpla sus expectativas:

- El cliente final pretende consumir las aplicaciones de manera sencilla, estando presentes en su vida cotidiana aportando algún tipo de valor. Asimismo, pretende que se mejoren los medios de comunicación, no solo en cuanto a eficiencia o velocidad, sino también a nivel de seguridad y privacidad en la red. A esto subyacen problemas como la gestión de datos cada vez mayores y el acceso en cualquier momento, lugar y dispositivo. Estas nuevas capacidades podrían transformar además las ciudades y la forma en que sus ciudadanos conviven en ellas, convirtiéndose Internet en una herramienta social más.
- Las empresas y organizaciones buscan estar cerca de sus clientes para ofrecerles un mejor servicio. Por ello, obtener información acerca de cómo se comportan e interactúan en la red podría llevar a ofrecer un servicio más personalizado y, por tanto, más eficaz. Esta interacción podría ser útil en alguna de las fases del ciclo de vida de los productos. Los requisitos adicionales que se le exigen al Internet del futuro en materia de servicios de negocios tienen que ver con la escalabilidad, disponibilidad global y el cumplimiento de los requisitos de los clientes. Una plataforma apropiada contribuiría a satisfacer estas demandas.

- Los desarrolladores, integradores y proveedores serán los encargados de crear estas aplicaciones inteligentes que cubrirán las necesidades del usuario. Desde la perspectiva de estos desarrolladores, las aplicaciones y servicios deben: estar basadas en estándares y lenguajes conocidos; disponer de APIs abiertas y frameworks que faciliten un desarrollo potente y flexible, sin atarse a un determinado distribuidor; facilitar medios para administración y distribuir la utilidad a múltiples usuarios.

1.1.2 Fi-Ware como parte del programa FI-PPP

El proyecto Fi-Ware consiste en el diseño y desarrollo de una plataforma (*Core Platform*) para el Internet del Futuro. Está enmarcado en el programa **FI-PPP**¹ (*European Future Internet Public Private Partnership*), cuyo objetivo es mejorar la competitividad de Europa en estas tecnologías y apoyar nuevas aplicaciones y servicios emergentes. Un acercamiento a este programa se encuentra en la imagen, donde se muestra cómo diferentes socios del proyecto colaboran para generar nuevos requerimientos para el proyecto, necesarios para poder desplegar su tecnología u ofrecer su servicio a través de Fi-Ware. Esta plataforma debería tratar de cumplir en la medida de lo posible estos requerimientos.

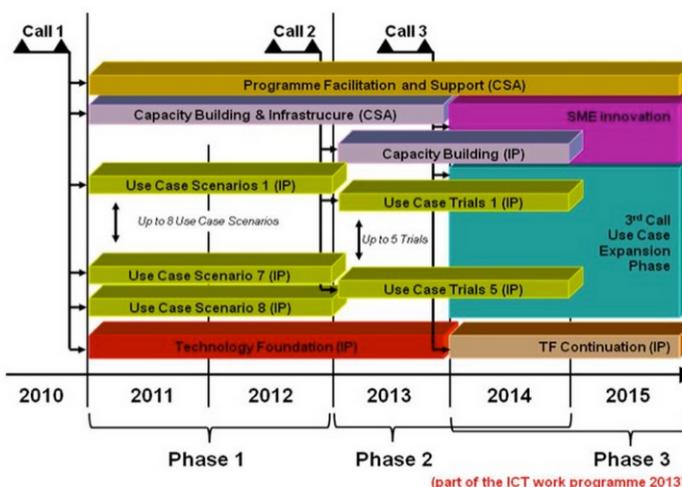


Figura 1.1 El programa FI-PPP.

1.1.3 Objetivos del proyecto

El principal, y por el cual Fi-Ware hoy en día es una realidad, es la creación de la plataforma mencionada para el Internet del Futuro. Mediante esta plataforma se pretende incrementar la competitividad de la economía europea basada en las TIC a través de la introducción de una infraestructura para la creación y entrega de servicios digitales, contando con una alta calidad de servicio y garantías de seguridad. Esto es, proporcionar una base sólida sobre la que construir un ecosistema sostenible para los proveedores de servicios innovadores y los consumidores finales, los cuales participarán activamente en el contenido, la creación y la consumición del servicio.

La plataforma, a la que denominaremos Fi-Ware Platform, o simplemente Fi-Ware, es abierta y libre de royalties y está basada en **Generic Enablers** o GEs que están presentes en múltiples de áreas de uso y ofrecen funciones reutilizables. Un GE es un conjunto de funciones de la plataforma, de propósito general y que se encuentran disponibles a través de APIs. Por tanto, uno de los objetivos principales del proyecto Fi-Ware será identificar y especificar estos GEs, junto con desarrollar implementaciones de ellos. Cualquiera de estas implementaciones comprende un conjunto de componentes y ofrece una serie de funcionalidades que pueden ser utilizadas, combinadas y personalizadas para un área concreta.

¹ Puede encontrarse más información en <http://www.fi-ppp.eu/>

1.1.4 Áreas técnicas definidas

La plataforma Fi-Ware, por tanto, está basada en GEs enmarcados en alguno de las siguientes áreas o capítulos técnicos:

- **Cloud hosting.** Capa que provee almacenamiento, computación y recursos de red, y sobre la que los servicios pueden ser administrados y servidos.
- **Análisis y administración de datos de contexto.** Capacidad para procesar, analizar y acceder al *big data* que podrá constituir información valiosa a consumir por las aplicaciones.
- **Ecosistema de aplicaciones y servicios.** Infraestructura para crear, administrar y consumir nuevos servicios a lo largo de su ciclo de vida.
- **Interfaz para redes y dispositivos (I2ND).** Define un espacio que facilite el proporcionar habilitadores genéricos para conseguir una infraestructura de red abierta y estandarizada.
- **Habilitadores para Internet of Things.** Constitución del puente entre los dispositivos conectados y la plataforma.
- **Seguridad.** Mecanismos que aseguran que la entrega y el uso de los servicios cumplan las especificaciones en materia de seguridad.

Para ilustrar el concepto de GE podemos nombrar algunos que fueron identificados inicialmente como vinculados al capítulo de análisis y administración de datos de contexto: algún tipo de GE que permita la recopilación y almacenamiento de datos masivos que provienen de diferentes fuentes. Este GE puede estar basado, a su vez, en una serie de ellos. La naturaleza abierta de estos GEs permite reemplazar una implementación concreta en Fi-Ware dentro de una instancia particular desplegada para amoldarse a la aplicación concreta.

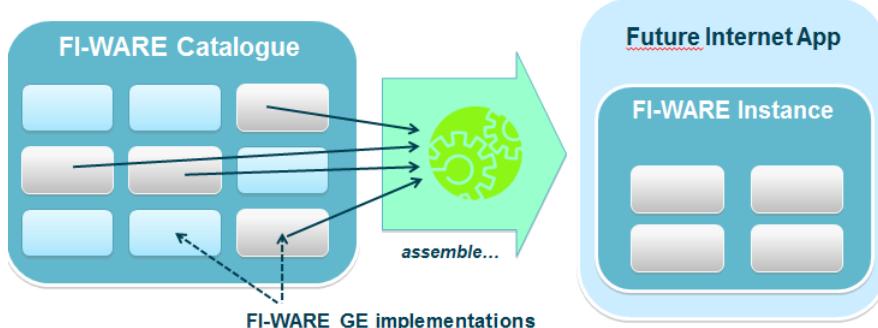


Figura 1.2 Uso de GEs en instancias Fi-Ware.

1.1.5 Fi-Ware Testbed y Fi-Lab

El proyecto Fi-Ware desplegó, inicialmente, una instancia susceptible de ser utilizada por los socios mencionados en el programa FI-PPP. La llamada **Fi-Ware Testbed** se utilizó para probar y ejecutar nuevas aplicaciones de Internet basadas en GEs de Fi-Ware. Un banco de pruebas funcional que se acercara a la idea inicialmente propuesta.

Además de este banco de pruebas, posteriormente se despliega una instancia para creación de aplicaciones de naturaleza abierta para estimular el conocimiento de los proveedores y desarrolladores sobre la plataforma, de manera que estos se sintiesen atraídos a participar y construir una comunidad de usuarios. A este nuevo banco de pruebas se le denomina **Fi-Lab**.

1.2 Arquitectura

1.2.1 Arquitectura general de referencia

En el capítulo anterior introducimos las principales áreas técnicas en las que se enmarcan los GEs de Fi-Ware. La arquitectura de referencia está ligada a cada una de estas áreas de aplicación por lo que, si bien introduciremos una visión de la arquitectura general, no profundizaremos en cada una para no extender innecesariamente el alcance de este documento. El siguiente diagrama muestra la mayoría de los componentes (GEs) y actores implicados en la plataforma, así como las áreas a las que pertenecen.

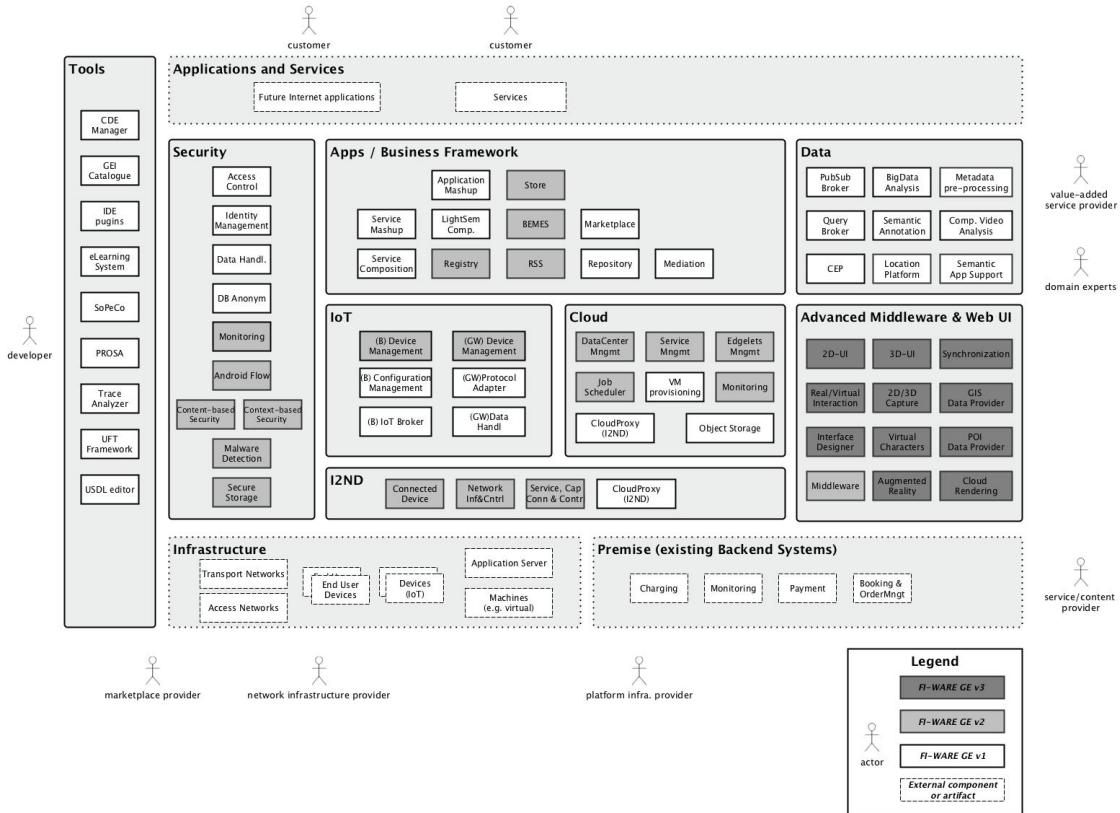


Figura 1.3 Visión general de la arquitectura de Fi-Ware.

En el catálogo actual de Fi-Ware² podemos encontrar implementaciones de los GEs disponibles para instanciar, por ejemplo, en Fi-Lab, además de la documentación necesaria para su uso. Es este el momento adecuado, teniendo en cuenta la naturaleza del presente proyecto, para presentar y ahondar en aquellos GEs que nos serán de mayor utilidad. Centraremos nuestra atención en los que provean funcionalidades en el ámbito de *Internet of Things*. En este sentido, los GEs enmarcados en el capítulo de *Data/Context Management* facilitan el desarrollo de aplicaciones que requieren recopilar y procesar grandes cantidades de información en tiempo real procedentes de los usuarios finales.

1.2.2 Especificaciones NGSI en Fi-Ware

NGSI es un estándar para administración de contexto definido por el OMA (*Open Mobile Alliance*). Este estándar define dos interfaces: para intercambio de información sobre entidades y sus atributos (OMA NGSI-10) y para disponibilidad de la información de estas entidades (OMA NGSI-9). En esta última lo que se intercambia es información acerca del proveedor de la información de la entidad, no de ésta como tal.

² <http://catalogue.fi-ware.org/enablers>

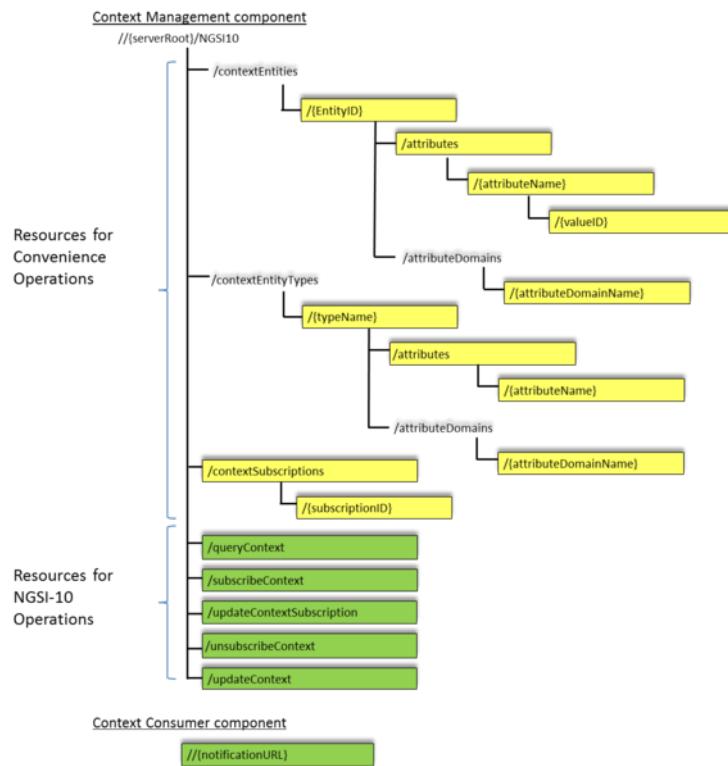


Figura 1.4 Recursos de NGSI.

En Fi-Ware se realizan implementaciones propias del estándar NGSI: APIs RESTful vía HTTP cuyo propósito es intercambiar información de contexto. Las diferencias con el estándar son mínimas, por lo que las trataremos como meras implementaciones de éste. Los tres principales tipos de interacciones con estas APIs son: consultas, suscripciones y actualizaciones del contexto (invocadas por los proveedores de información).

En el ejemplo que se presenta a continuación, tienen lugar dos tipos de interacciones NGSI: `updateContext` y `queryContext`. En este caso, se presenta una aplicación que recoge a tiempo real la velocidad instantánea de un vehículo y la muestra a través de una aplicación móvil. El *Context Broker*, del que hablaremos más adelante, resulta ser el intermediario entre el *Context Producer* (velocímetro) y el *Context Consumer* (móvil).

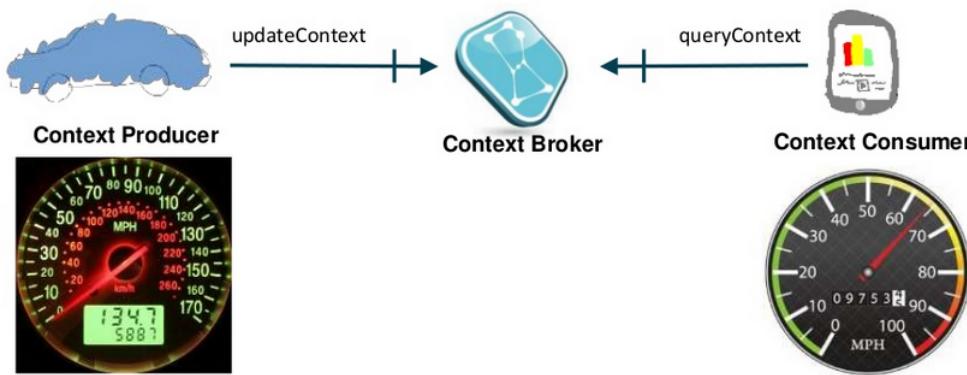


Figura 1.5 Ejemplo de uso: velocímetro a tiempo real en app móvil.

En Fi-Ware, estas operaciones NGSI son invocadas a través de un método POST de HTTP, por lo que los elementos finales (*Context Producer* y *Context Consumer*) deben poseer conexión a Internet para ejecutar estas llamadas, o existir otro sistema que las realice por ellos. Resultará de aplicación esta operativa en el

presente proyecto, por lo que en capítulos posteriores tendremos oportunidad de comprobar cómo se comporta con mayor profundidad.

1.2.3 Arquitectura en *Data/Context Management*

Los GEs pertenecientes a esta parte de la arquitectura de Fi-Ware permiten:

- Generar, notificar, suscribirse o consultar cierta información de contexto proveniente de diferentes fuentes. Un ejemplo de información de contexto lo constituyen las lecturas periódicas de temperatura de un sensor emplazado en un lugar particular.
- Actuar sobre el contexto en base a información recogida que pueda disparar un cierto evento.
- Procesar metadatos que utilicen una semántica estandarizada referidos a una cierta información contextual.
- Tratar grandes cantidades de información de manera agregada utilizando técnicas de Big Data Map & Reduce, con el objetivo de generar nuevo conocimiento.

A continuación se muestran los principales GEs en el capítulo de *Data/Context Management*.

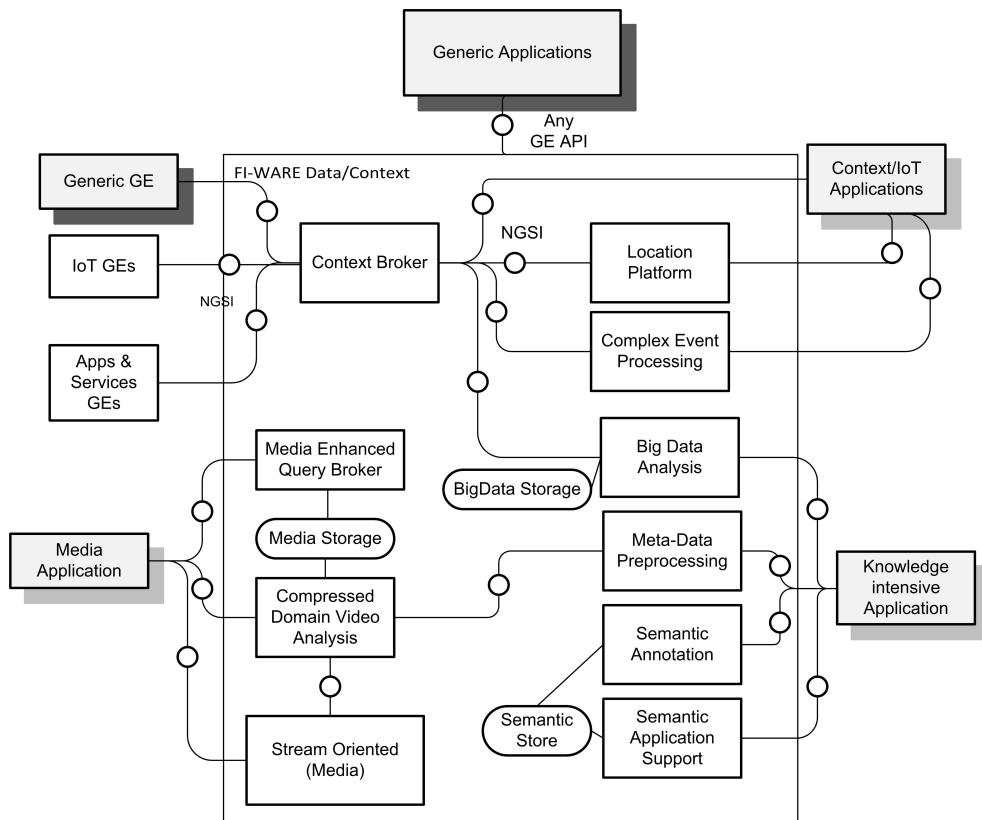


Figura 1.6 Visión general de la arquitectura del *Data/Context Management*.

Un concepto clave en esta arquitectura es la definición de la estructura de los datos del contexto. Esta definición está basada en el estándar NGSI anteriormente mencionado por lo que, si bien se aportan algunas nociones de los conceptos, podría encontrarse mayor información en el propio estándar.

Un **data element** es un paquete de información que es producida o recogida y que puede ser relevante analizar para obtener un nuevo conocimiento. Consiste en una secuencia de una o más tripletas `<name, type, value>` referidas a diferentes atributos de este elemento. Puede existir además *metadata* ligado a alguno de estos atributos. Los *data element* podrían poseer un identificador para hacer más sencilla su inclusión en un almacén (típicamente una base de datos). Este identificador no se considera parte de la estructura.

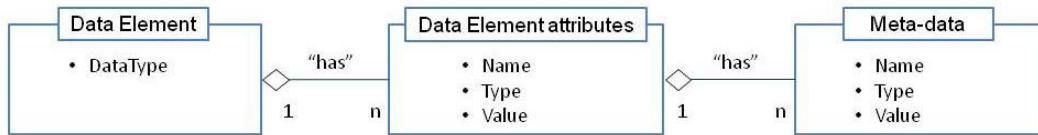


Figura 1.7 Estructura de un *data element*.

Por otro lado, en Fi-Ware se define el **contexto** y los **elementos del contexto** (*context elements*), que vienen a extender el concepto de *data elements* asociandole un EntityId y un EntityType que identifican únicamente la entidad (que en realidad podría referirse, a su vez, a un grupo de entidades).

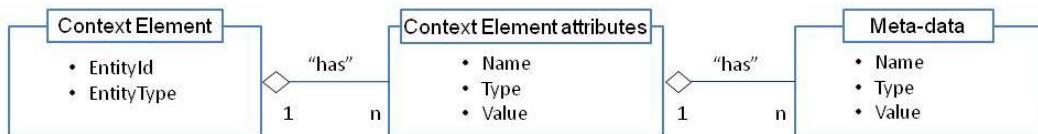


Figura 1.8 Estructura de un *context element*.

Los **eventos** se definen como acontecimientos en un contexto particular. Estos eventos disparan algún tipo de acción particular que deba llevarse a cabo.

Un ejemplo clarificador acerca de estas definiciones lo constituye un cierto dispositivo que, periódicamente, provea lecturas de algún parámetro medible en una habitación como, por ejemplo, temperatura y humedad relativa. En este dominio, el elemento de contexto sería el sensor y el contexto el lugar y las condiciones ambientales en que se encuentra. Podría considerarse como evento un cambio en alguno de los parámetros medibles, desencadenando éste, por ejemplo, una orden para reducir o aumentar la temperatura de un climatizador presente en el habitáculo.

1.2.4 Dependencias e interacción entre GEs

Si bien los GEs siguientes se presentan enmarcados dentro del área de *Data/Context Management*, lo cierto es que son completamente modulares y pueden encontrarse en otras áreas de Fi-Ware. Todos están basados en APIs NGSI y, como puede observarse, el núcleo de esta arquitectura (fuente y sumidero de diferentes solicitudes NGSI de los GEs) es el Context Broker.

El principio fundamental en el que se basa este Context Broker es conseguir desacoplar a los productores de los consumidores de información de contexto. De esta forma, los productores podrán publicar información de contexto a placer y los consumidores hacer uso de ella cuando les interese. Este Context broker debe ser capaz, por tanto, de almacenar la información de contexto de manera que pueda ser consumida en cualquier momento.

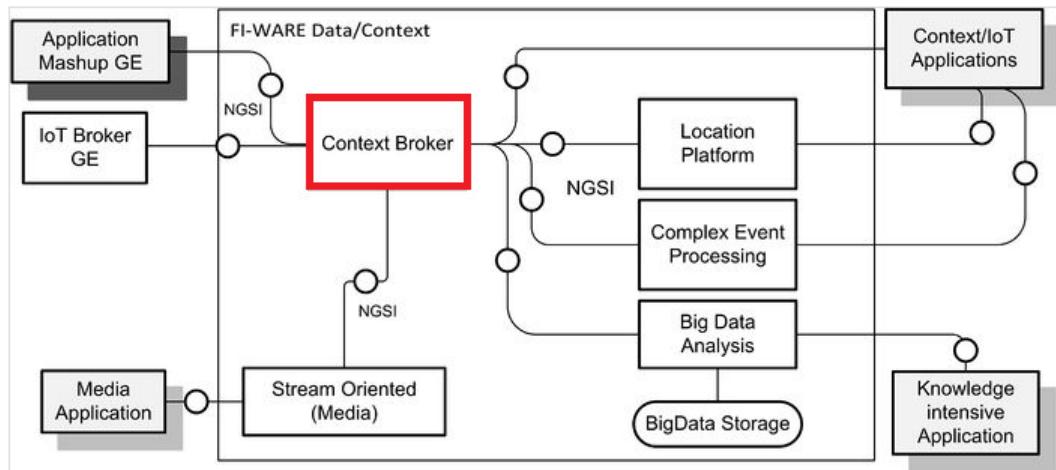


Figura 1.9 Integración NGSI para *Data/Context Management*.

1.2.5 Orion Context Broker

Orion Context Broker, en adelante OCB, es una implementación en Fi-Ware del Context Broker mencionado con persistencia de datos de contexto en una base de datos MongoDB. Viene a cubrir dos roles principales en la declaración de GEs de Fi-Ware: *Publish/Suscribe Context Broker GE* y *Configuration Management GE*, siguiendo y construyendo una implementación propia del estándar OMA NGSI 9/10.

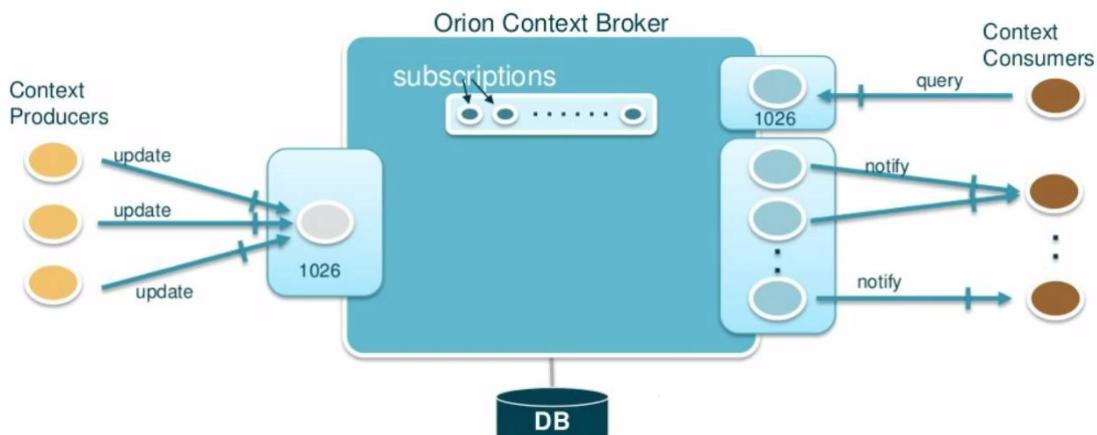


Figura 1.10 Interacciones típicas a través del Context Broker.

Mediante esta implementación del Context Broker se pueden realizar diversas operaciones de contexto, tales como registrar lecturas de humedad relativa en un área de una ciudad, obtener notificaciones cuando esta humedad se ve modificada o guardar dicha información para un análisis posterior. En el caso que nos ocupa, OCB es el componente que nos facilitará ofrecer la información de los sensores a las aplicaciones, constituyendo el intermediario entre el back-end y el front-end que desplegaremos en Fi-Lab.

2 Elección de dispositivos

2.1 Restricciones generales

Para la elección de los dispositivos necesarios se ha de tener muy en cuenta el emplazamiento final en el que se desplegarán. El objetivo principal es instalar sensórica para control de polución en una ciudad por lo que, si bien constituyen un pilar fundamental las características técnicas y de implementación que impongamos, esto podría quedar relegado a un segundo plano en el momento en que la vida útil y autonomía del dispositivo pueda verse reducida, teniendo en cuenta las posibles condiciones meteorológicas a las que estará expuesto el equipo.

Si bien a lo largo de este apartado podrán aparecer nuevas condiciones a exigir para equilibrar la balanza en cuanto a especificaciones satisfechas/precio, definiremos algunas básicas que fijarán un rango de dispositivos, de manera que podamos estrechar el cerco y realicemos una elección mucho más precisa y adecuada.

- Los dispositivos se desplegarán en un emplazamiento exterior, tipo urbano. Esto obliga a que deban estar protegidos ante los diferentes agentes meteorológicos, típicamente lluvia, temperaturas extremas o cualquier otro que pudiese afectar al correcto funcionamiento de la electrónica interna. Constituye esta la principal restricción de diseño.
- Un bajo coste permitiría desplegar un mayor número de dispositivos. Si existe un gran número de nodos en las zonas donde se despliegan podría realizarse una traza mucho más precisa del nivel de polución presente.
- Los sensores deben funcionar sin necesidad de estar conectados a una fuente de alimentación, por lo que el consumo debe ser bajo para alargar la vida útil de las baterías. Como añadido a este caso, podrían colocarse pequeños paneles solares para aportar energía adicional si necesitásemos un gran aporte.
- Conectividad a Internet para monitorización y envío de lecturas periódicamente. De esta forma podremos aprovechar este *data mining* con efecto de analizar y extraer información útil.

Trataremos entonces de aportar las soluciones presentes en el mercado que puedan ajustarse lo máximo posible a las restricciones impuestas.

2.2 Factores involucrados

2.2.1 Protección ante agentes externos

Una de las características que define el ámbito de utilización de un cierto equipo electrónico hoy en día, es su **grado de protección IP**. Este grado de protección IP especifica un sistema de clasificación de los contenedores que protegen la electrónica interna (estándar ANSI/IEC 60529-2004), referenciado por dos letras y dos dígitos (IPxx). IP hace referencia a *International Protection* y los dígitos clasifican el nivel de protección ante entrada de objetos sólidos y ante entrada de agua, definiéndose un rango de 1 a 6 para el primero y 1 a 8 para el segundo. Para obtener esta certificación se deben cumplir las condiciones recogidas en las siguientes tablas:

Tabla 2.1 Grado de protección IP: primer dígito.

Dígito	Tamaño del objeto entrante	Protección ante
0	-	No protegido
1	<50 mm	Un elemento con esta medida no debe llegar a entrar por completo.
2	<12.5 mm	Un elemento con esta medida no debe llegar a entrar por completo.
3	<2.5 mm	Un elemento con esta medida no debe entrar en absoluto.
4	<1 mm	Un elemento con esta medida no debe entrar en absoluto.
5	Protección contra polvo	Aun no pudiendo evitarse, la cantidad permitida no debe interferir con el correcto funcionamiento
6	Protección alta contra polvo	El polvo no debe entrar bajo ninguna circunstancia

Tabla 2.2 Grado de protección IP: segundo dígito.

Dígito	Protección ante	Lugar de prueba	Resultados
1	Goteo de agua	Emplazamiento similar al de despliegue.	No debe entrar el agua cuando se la deja caer, desde 200 mm de altura respecto del equipo, durante 10 minutos (a razón de $3-5 \text{ mm}^3$ por minuto)
2	Goteo de agua	Emplazamiento similar al de despliegue.	Idéntica a la anterior, realizada cuatro veces desde diferentes ángulos
3	Agua en spray	Emplazamiento similar al de despliegue.	No debe entrar el agua nebulizada en un ángulo de hasta 60 a derecha e izquierda de la vertical a un promedio de 11 litros por minuto y a una presión de $80-100 \text{ kN/m}^2$ durante un tiempo no menor a 5 minutos.
4	Chorros de agua	Emplazamiento similar al de despliegue.	No debe entrar el agua arrojada a chorro (desde cualquier ángulo) por medio de una boquilla de 6,3 mm de diámetro, a un promedio de 12,5 litros por minuto y a una presión de 30 kN/m^2 durante un tiempo que no sea menor a 3 minutos y a una distancia no menor de 3 metros.
5	Chorros de agua.	Emplazamiento similar al de despliegue.	No debe entrar el agua arrojada a chorros (desde cualquier ángulo) por medio de una boquilla de 12,5 mm de diámetro, a un promedio de 100 litros por minuto y a una presión de 100 kN/m^2 durante al menos de 3 minutos y a una distancia no menor a 3 metros.
6	Chorros muy potentes de agua.	Emplazamiento similar al de despliegue.	Condiciones similares pero más exigentes que las anteriores.
7	Inmersión completa en agua.	El objeto debe soportar sin filtración la inmersión completa a 1 metro durante 30 minutos.	No debe entrar agua.
8	Inmersión completa y continua en agua.	Inmersión completa en agua: condiciones más severas.	No debe entrar agua.

Según esta especificación, exigir **una restricción IP55 o IP56** sería **suficiente** para nuestro equipo.

2.2.2 Comunicación

Los sensores se dispondrán a lo largo de la ciudad en emplazamientos sin determinar a priori. Las redes de sensores se caracterizan por tener un gran número de nodos, que pueden estar más o menos alejados entre sí, y no requerir, en principio, de una comunicación punto a punto entre nodos finales. Esto nos lleva a no considerar como opción una red de sensores cableada, la cual encarecería el despliegue y dificultaría instalaciones puntuales de nuevos dispositivos. Optaremos por protocolos de comunicación inalámbrica para el intercambio de información entre equipos de la red. Aparecen entonces diferentes opciones para redes de sensores, entre las que destacamos: Bluetooth, ZigBee, WiFi y tecnologías de redes móviles.

Protocolos de comunicación inalámbrica

No debemos perder de vista las restricciones impuestas para los elementos finales (sensores). Quizá una de las más importantes sea la que tiene que ver con el consumo y la vida útil de los dispositivos. Esta restricción condiciona en gran medida la elección de uno u otro protocolo. A continuación podemos encontrar una comparativa de los protocolos mencionados que nos facilitará la tarea de elegir entre las diferentes opciones.

Tabla 2.3 Comparativa de protocolos para comunicaciones inalámbricas.

Protocolo	Ventajas	Inconvenientes
Bluetooth (IEEE 802.15.1)	<ul style="list-style-type: none"> Adecuado para aplicaciones que requieran un bajo consumo. Alta tasa de transmisión comparada con ZigBee (hasta 3Mbps). 	<ul style="list-style-type: none"> Alcance reducido (1-30 metros). Segundo potencia de transmisión.
ZigBee (IEEE 802.15.4)	<ul style="list-style-type: none"> Ideal para aplicaciones de muy bajo consumo (30mA en transmisión y apenas algunos μA en reposo), mucho menor que Bluetooth. Mayor número de nodos por subred (255) que Bluetooth (8). Topologías en malla: mejora de la robustez de la red. 	<ul style="list-style-type: none"> Tasas de transmisión muy bajas (hasta 250kbps).
WiFi (IEEE 802.11b)	<ul style="list-style-type: none"> Muy altas tasas de transmisión (hasta 300Mbps en sus últimas versiones). Gran número de protocolos de cifrado asociados, lo cual se traduce en mayor seguridad y fiabilidad. 	<ul style="list-style-type: none"> Alto consumo: reducción de la vida útil de las baterías.
Red móvil (3G, GPRS, GSM...)	<ul style="list-style-type: none"> Altas tasas de transmisión: velocidad depende de la tecnología de red presente. La estructura interna de la red es transparente al usuario. 	<ul style="list-style-type: none"> Consumo muy elevado motivado por las altas potencias de transmisión.

Los dispositivos finales proveerán lecturas de los sensores y enviarán los paquetes de datos sobre señales de radio utilizando alguno de estos protocolos. Se estima que estos paquetes de datos posean un peso ligero, como máximo de unos pocos cientos de bytes, por lo que no necesitamos una alta tasa de transmisión.

Elementos de una red de sensores

- Sensores.** Presentes en los nodos para tomar información del medio y convertirla en una señal entendible por un sistema de procesamiento.
- Nodos o Motas.** Dispositivo electrónico compuesto por un módulo de comunicaciones, sistema de aporte de energía, sensores y sistemas para procesamiento de información.
- Gateway.** Elemento electrónico que sirve de pasarela entre la red de sensores y una red de datos TCP/IP.
- Servidor de almacenamiento/equipo Cloud.** Elemento electrónico que sirve de pasarela entre la red de sensores y una red de datos TCP/IP.

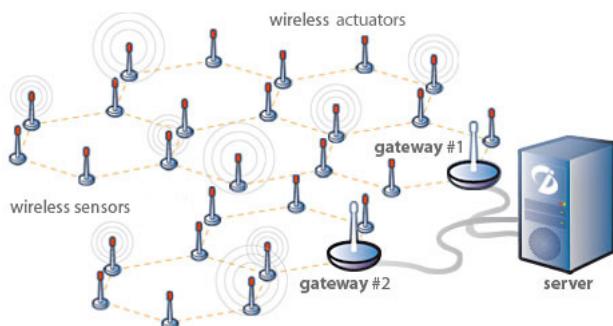


Figura 2.1 Elementos principales en una red de sensores.

El protocolo ZigBee

Zigbee es un protocolo basado en el estándar IEEE 802.15.4 para aplicaciones que requieran un bajo consumo, no muy alta tasa de datos y comunicaciones seguras. Opera en las bandas libres de 868 MHz (Europa), 915 MHz (EEUU) y 2.4 GHz, sin embargo es esta última la libre en todo el mundo, por lo que se suelen emplear, en mayor medida, dispositivos ZigBee que levanten señal en esta radio. Fue desarrollado por la *ZigBee Alliance* hacia el año 2002, una organización sin ánimo de lucro formada por diferentes empresas. Se pretendía desarrollar un protocolo de comunicación que permitiese conseguir un muy bajo consumo, con dispositivos baratos, con un corto alcance (entre 10 y 100m) y que requiriese poca electrónica para construir un nodo. La aplicación típica para este protocolo la encontramos en proyectos de sensores y actuadores para domótica.

Se pueden definir tres tipos distintos de dispositivos ZigBee según su rol en la red:

- Coordinador ZigBee (*PAN Coordinator*). Al menos uno por red. Se encarga de configurar los parámetros de red y de gestión de red. En conjunción con los routers, trazará los caminos que deben seguir los dispositivos finales para interconectarse.
- Router ZigBee (*Full Function Device*). Conecta los diferentes clusters de la red, pudiendo actuar además como gateways hacia otras redes.
- Dispositivo final (*Reduced Function Device*). Funcionalidad completamente reducida para optimizar el consumo. No intervienen en la gestión de la red ni se comunican con otro dispositivo que no sea su nodo inmediatamente superior en escala, en este caso un router.

Es posible, asimismo, diseñar la red en base a topologías típicas como en estrella o árbol, pero la que resulta sin duda más interesante para este protocolo (y aventaja en cuanto a otros) es la topología en malla. Bajo esta topología, si se experimenta una comunicación en alguno de los nodos, existe siempre una alternativa y no se experimentaría una caída del servicio.

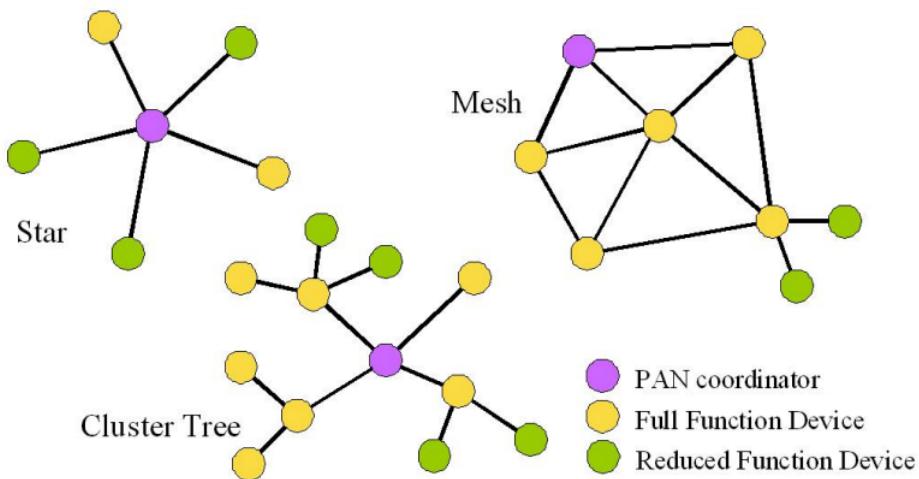


Figura 2.2 Topologías posibles en una red ZigBee.

El protocolo Bluetooth Low Energy

También conocido como BLE o Bluetooth Smart, esta nueva tecnología para redes WPAN viene a mejorar el conocido estándar Bluetooth tratando de conseguir un menor consumo y hacer frente, así, a protocolos ultra bajo consumo como ZigBee. Lo cierto es que en este sentido, lo consigue, siendo capaz de conseguir una vida útil de batería de varios años, al igual que ZigBee. Consigue hasta cuatro veces mayor tasa de transmisión de bits (1Mbps) aunque viendo reducida su cobertura (entre 5 y 10 metros). Requiere más electrónica para la construcción de un nodo, lo que hace que los módulos que implementan esta tecnología posean un coste mayor. Las topologías admitidas para este protocolo son estrella y árbol, por ser un protocolo de comunicación punto a punto. A continuación pueden apreciarse las modificaciones en las capas de una versión de la tecnología a otra.

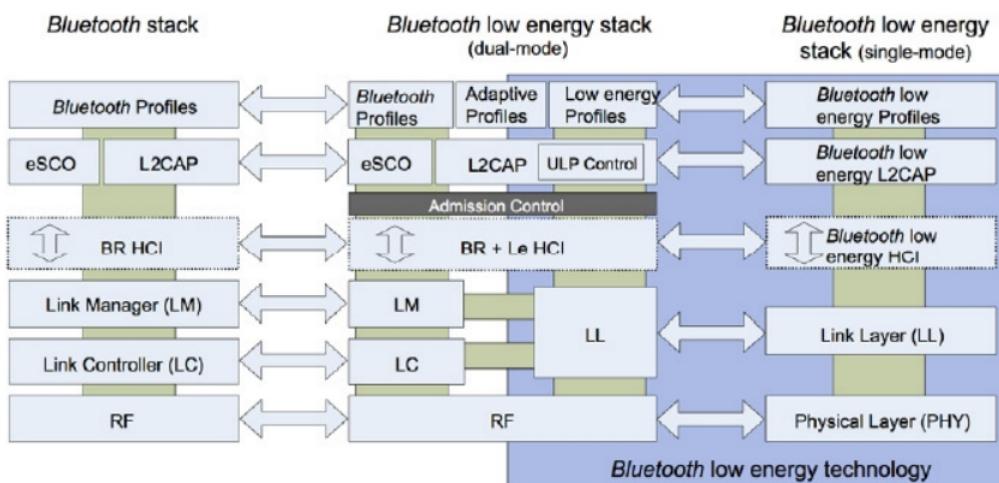


Figura 2.3 Comparativa de la pila de capas en Bluetooth y BLE.

Comparativa final y tecnología escogida

Si bien BLE consigue tasas de transmisión más altas que ZigBee, lo cierto es que para nuestra aplicación no requerimos de un gran bitrate. En cuanto al consumo, sobre el papel y teniendo en cuenta únicamente nodos finales (los que podrán ir al reposo tras su operación), están bastante parejos, pero existe un parámetro más a tener en cuenta: el tiempo que tarda un nodo final en volver del reposo. En ZigBee este tiempo es de aproximadamente 15s, mientras que en BLE es de 3s. Esto implica un menor tiempo de actividad para un

dispositivo ZigBee.

BLE es una muy interesante tecnología que bien podría servir a nuestro cometido, pero el menor coste de los módulos ZigBee, la posibilidad de diseñar la red con topología en malla para evitar nodos inaccesibles y el mayor alcance de cobertura hacen que el protocolo para comunicación inalámbrica de los dispositivos de la red sea **ZigBee**.

2.2.3 Electrónica interna

Existen diferentes alternativas en el mercado para dar solución a lo que pretendemos. Por un lado podemos optar por adquirir dispositivos Plug & Play de fabricante, electrónica open hardware o, por otro lado, diseñar y desarrollar nuestro propio dispositivo. Se plantean por tanto tres opciones perfectamente válidas que deberemos caracterizar para ofrecer una solución acertada, teniendo en cuenta tanto el alcance del proyecto como la satisfacción de las diferentes condiciones impuestas.

Lo que pretendemos es caracterizar concentraciones de gases en zonas urbanas y que estas puedan ser monitorizadas a través de Fi-Ware. La opción de diseñar y construir un dispositivo propio aumentaría el alcance del proyecto de manera considerable y no podríamos detenernos en lo verdaderamente interesante, que es la utilización de esta plataforma para proyectos de *Smart City*, por lo que queda **descartada** esta implementación. A continuación estudiaremos la viabilidad de cada una de las soluciones restantes.

Hardware de fabricante

Esta tecnología es aún emergente y no existen demasiadas soluciones en el mercado, pero empiezan a aparecer diseños e integraciones que son ofrecidos como un producto final. Veamos cuáles son los tres productos más interesantes para nuestro cometido.

TELEFÓNICA THINKING THINGS

Telefónica se sube al carro del *Internet of Things* lanzando una plataforma de hardware modular bajo esta denominación. Se trata de poder construir un dispositivo final en base a diferentes módulos, al estilo de la plataforma Arduino (y con su apoyo). Además ofrecen conectividad global de los dispositivos y monitorización a través de una plataforma propia. Posee un precio aproximado de 90 euros con conectividad global gratis durante seis meses.



Figura 2.4 Telefónica Thinking Things.

Este hardware es demasiado novedoso y actualmente el kit ambiental más parecido a lo que se pretende desplegar es el compuesto por sensores de ruido, luminosidad, temperatura y humedad. Esto es, los parámetros más típicos medibles en un determinado contexto en la ciudad. No aplica por tanto al caso en que nos encontramos por lo que, si bien podría ser una interesante opción en el futuro, tendremos que sopesar otras opciones.

SMART CITIZEN

Smart Citizen es una placa de electrónica basada en Arduino cuyo principal objetivo es poder otorgar medidas de diferentes parámetros medibles en la ciudad, tal y como pretendemos. Su precio aproximado, sin incluir la conectividad es de 175 euros. Si bien mediante esta placa se pueden obtener lecturas de gases como el CO y NO₂ y posee un módulo de comunicación M2M, la problemática reside en el mismo lugar que para *Thinking Things*: actualmente no existen *shields* acoplables para incluir un mayor número de sensores. Esta limitación no hace viable su utilización.

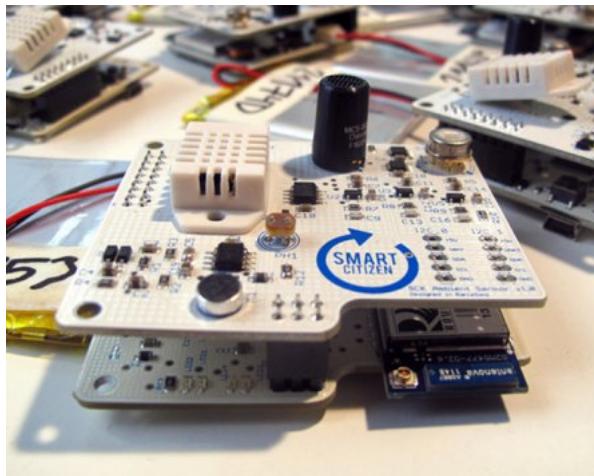


Figura 2.5 Smart Citizen Board.

LIBELIUM WASPMOTE

Si alguna empresa se ha posicionado de manera importante en el sector, esa es la española Libelium. Su modelo de placa Wasp mote es el núcleo de la que probablemente sea la oferta más completa que existe en cuanto a *shields* para sensórica, comunicaciones y otras aplicaciones. Estas placas también están basadas en Arduino, completándose el producto con librerías e IDEs propios para facilitar el manejo de estas *shields*. Este núcleo Wasp mote, junto con un módulo de comunicación, cuesta 155 euros.



Figura 2.6 Libelium Wasp mote.

En nuestro caso, buscamos una *shield* que contenga sensores para diferentes tipos de gases, lo cual es exactamente lo que la *shield Gas Sensor Board* nos ofrece a un precio aproximado de 120 euros.

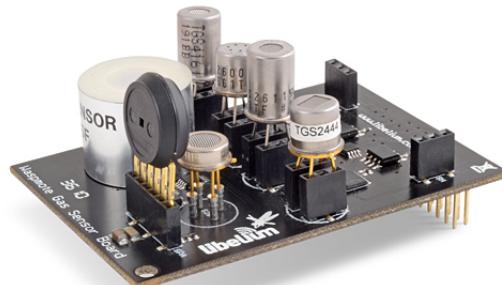


Figura 2.7 Gas Sensor Board para Wasp mote.

Electrónica open source: Arduino

En este momento nos planteamos construir los dispositivos utilizando la más conocida plataforma de electrónica open source: **Arduino**. Las placas presentadas hasta ahora están basadas en esta plataforma, tanto a nivel hardware como software, por lo que no parece descabellado tratar de conseguir el mismo efecto que pretendemos construyendo a partir de esta base para tratar de ahorrar en costes. Existen diferentes *boards* y *shields* acoplables, cada una enfocada a una aplicación diferente. Deberemos tratar de encontrar aquella que se ajuste a la nuestra, en este caso proveer lecturas de polución.

En este momento no existen *shields* Arduino que resulten adecuadas para nuestra aplicación, por lo que podríamos listar una serie de condiciones y elementos exigibles a esta placa para hacernos una idea de lo que necesitamos:

- Suficientes entradas/salidas para conectar diferentes tipos de sensores.
- *Shield* para acoplar un módulo XBee (se requiere al menos una UART).
- Encapsulado externo con certificación de al menos IP55.
- Encapsulado externo con igual certificación para cada sensor a conectar. Dejar los sensores dentro de la caja estanca provocaría lecturas que no se corresponderían con la realidad.
- Electrónica adicional para convertir voltajes/corrientes adecuándose a cada tipo de sensor en la placa.
- Escalabilidad, para ofrecer diferentes soluciones con los sensores que se comercializan hoy en día.

Esto no son más que restricciones generales impuestas ahora sobre este tipo de solución. En la tienda virtual de Arduino (<http://store.arduino.cc/>) se puede encontrar la placa Arduino UNO, que es la placa de gama más baja que podría satisfacer nuestras necesidades a nivel de electrónica y se asemeja más al núcleo Waspmove, aunque éste último presente algunos elementos electrónicos más. Recogemos algunas de sus características técnicas más relevantes:

Tabla 2.4 Arduino UNO: especificaciones técnicas.

Microcontrolador	ATmega328
Voltaje operativo	5V
Voltaje de entrada recomendado	7-12V
Entradas/Salidas digitales	14
Entradas/Salidas digitales PWM	6
Entradas analógicas	6
Memoria Flash	32 KB
Velocidad de reloj	16 MHz
Precio	25 euros

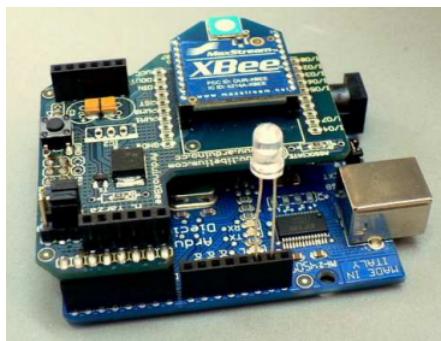


Figura 2.8 Placa Arduino Uno con *shield* para módulo ZigBee.

Una *shield* para comunicación y un módulo ZigBee tendrían un coste aproximado de 43 euros, lo que daría un coste total de unos 73 euros. Habría que añadir componentes de la Waspmove que la placa Arduino UNO

no posee como un acelerómetro, módulo SD o un RTC. La diferencia en precio final es de aproximadamente 30 euros.

A esto, habría que añadir el sobrecoste que origina el encapsulado y las baterías, diseñar cierta electrónica para amoldar la placa a cada tipo de sensor y desarrollar todo el software necesario para que todo quedase en perfecto funcionamiento. Aún así esta opción seguiría resultando más barata.

Realizando una pequeña estimación del trabajo a realizar previo al uso de estas placas, parece que éste extiende el alcance del proyecto en demasiá, al igual que diseñar nuestra propia electrónica desde cero para la aplicación. Teniendo en cuenta que en este proyecto nos encontramos más enfocados en la solución, es decir, en ofrecer una integración en la plataforma de Fi-Ware, y no tanto en el diseño y construcción de este tipo de dispositivos, la **opción de Waspmotte Plug & Sense Smart Environment parece la más adecuada**.



Figura 2.9 Libelium Smart Environment desplegada en una ciudad.

2.3 Elección final: Waspmove

2.3.1 Core: especificaciones técnicas

Lo realmente interesante de esta plataforma hardware es su capacidad para acoplar diferentes shields diseñadas, especialmente, para proyectos Smart City. Hoy en día no existe ninguna otra solución tan modular y escalable dirigida a este tipo de proyectos. Presentamos la última versión del core: Waspmove PRO (v1.2).

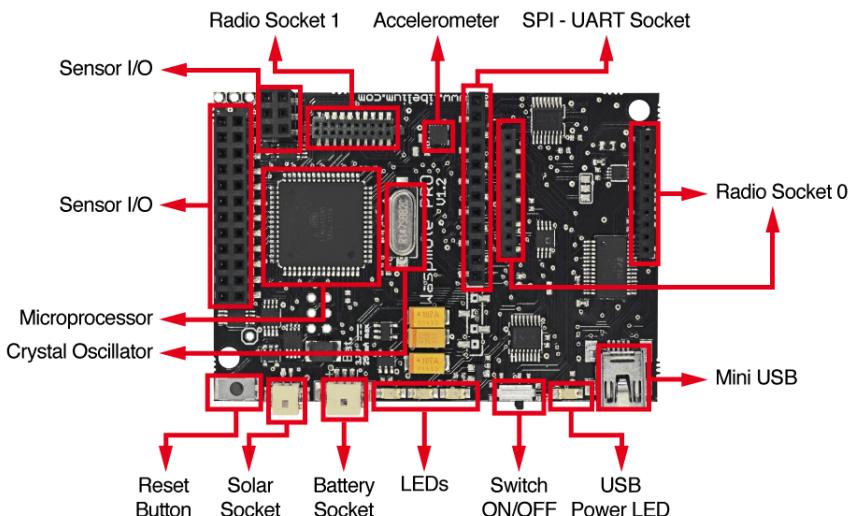


Figura 2.10 Waspmove v1.2: frontal.

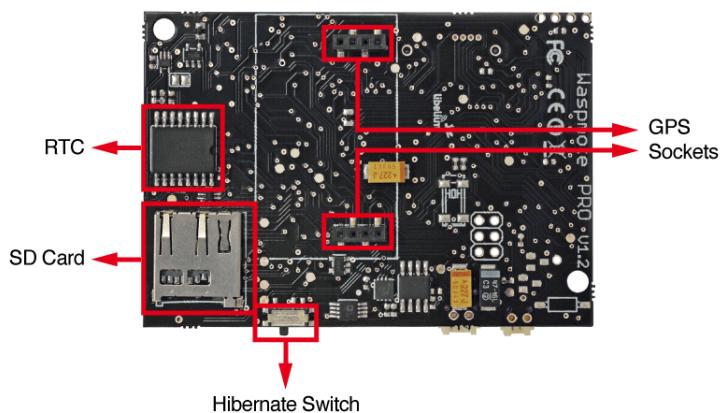


Figura 2.11 Waspmove v1.2: trasera.

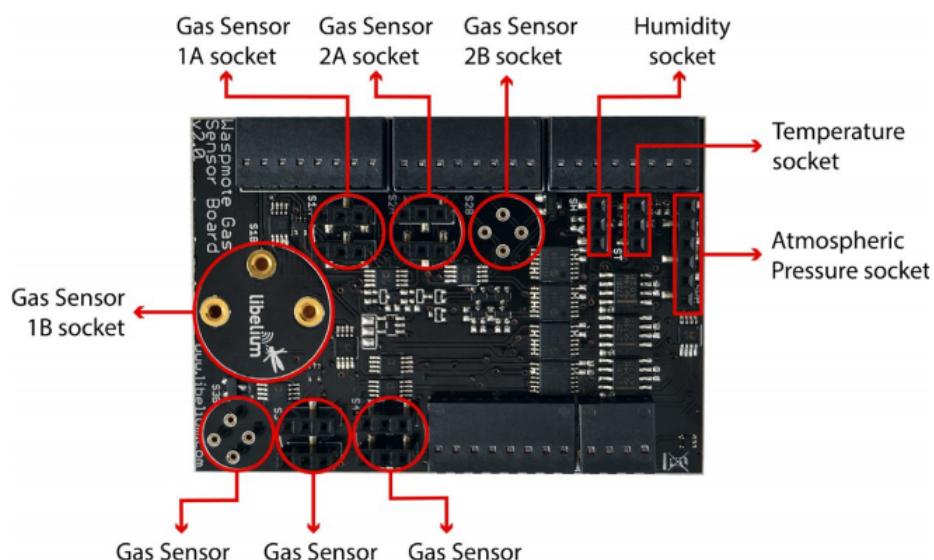
Esta última versión viene a mejorar la anterior con una mayor velocidad de procesamiento, sustitución de jumpers por switches e inclusión de un puerto SPI entre muchas otras. En general, algunos cambios estructurales para mejorar la eficiencia y el uso, aunque incompatibiliza el uso de algunas shields que podrían resultar interesantes. Libelium continúa ofreciendo soporte y unidades de la versión anterior por lo que, elegir una u otra dependiendo de la shield a utilizar, queda a elección del usuario. Se presentan aquí algunas de las características técnicas de esta plataforma.

Tabla 2.5 Wasp mote v1.2: especificaciones técnicas.

Microcontrolador	ATmega1281
Voltaje operativo	5V
Voltaje de entrada recomendado	7-12V
Entradas/Salidas digitales	8
Entradas/Salidas digitales PWM	1
Entradas analógicas	7
Memoria Flash	128 KB
Velocidad de reloj	14 MHz
Módulo radio	Bluetooth/WiFi/ZigBee
Precio	155 euros

2.3.2 Gas Sensor Board

Esta shield se diseñó con el objetivo de monitorizar parámetros ambientales, tales como la temperatura, humedad o presión atmosférica, pero también concentraciones de gases conocidos cuyos valores en entornos urbanos se encuentran regulados y es interesante controlar. Particularmente, esta shield permite la inclusión de hasta 7 sensores de gas a la vez, regulando la alimentación de cada uno y realizando un tratamiento de la señal obtenida para adecuarla a lo que se pretende. Esto se consigue a través de una etapa amplificadora no inversora con una ganancia máxima de 101, regulable por software gracias a un potenciómetro digital conectado al bus I2C.

**Figura 2.12** Shield Gas Sensor Board: detalle.

Los tipos de gases que pueden monitorizarse mediante una conexión directa a la placa son:

- Monóxido de carbono – CO.
- Dióxido de carbono – CO2.
- Oxígeno – O2.
- Metano – CH4.
- Hidrógeno – H2.
- Amoníaco – NH3.
- Isobutano – C4H10.
- Etanol – CH3CH2OH.

- Tolueno – C₆H₅CH₃.
- Sulfuro de hidrógeno – H₂S.
- Dióxido de nitrógeno – NO₂.
- Ozono – O₃.
- Compuestos orgánicos volátiles (VOC's).
- Hidrocarburos.

Existen sockets dedicados a sensores que requieren un tratamiento especial a nivel eléctrico, como es el caso del CO₂ (socket 1A) o O₃ (socket 2B). Existe una API desarrollada especialmente para trabajar con esta shield: WaspSensorGasv20. Más adelante tendremos ocasión de usarla durante el desarrollo del software de las motas.

Resulta muy interesante este diseño modular, con posibilidad de acoplar diferentes tipos de sensores, ya que no se requiere electrónica adicional o rediseñar el sistema según la carga que conectemos a la plataforma.

2.3.3 Libelium Smart Environment

La versión del core Waspmove junto con la shield Gas Sensor Board, y completada con una certificación IP65, es la Waspmove Smart Environment. Esta plataforma provee 6 conectores diferentes internamente conectados a los sockets de la placa Gas Sensor Board, de manera que puedan utilizarse diferentes sensores sin más que ajustar la conexión.



Figura 2.13 Detalle del modelo Smart Environment.

Se debe tener especial precaución con la colocación de los sensores, pues adaptar el sensor sobre un socket que provea una alimentación u un tratamiento interno diferente podría dañarlos. Libelium provee una tabla de compatibilidad para los conectores de esta plataforma.

Sensor Socket	Sensor probes allowed for each sensor socket	
	Parameter	Reference
A	Temperature	9203
	Carbon monoxide - CO	9229
	Methane - CH ₄	9232
	Ammonia – NH ₃	9233
	Liquefied Petroleum Gases: H ₂ , CH ₄ , ethanol, isobutene	9234
	Air pollutants 1: C ₄ H ₁₀ , CH ₃ CH ₂ OH, H ₂ , CO, CH ₄	9235
	Air pollutants 2: C ₆ H ₅ CH ₃ , H ₂ S, CH ₃ CH ₂ OH, NH ₃ , H ₂	9236
B	Alcohol derivates: CH ₃ CH ₂ OH, H ₂ , C ₄ H ₁₀ , CO, CH ₄	9237
	Humidity	9204
	Atmospheric pressure	9250
C	Carbon dioxide - CO ₂	9230
D	Nitrogen dioxide - NO ₂	9238 and 9238-B
E	Ozone - O ₃	9258 and 9258-B
	Hydrocarbons - VOC	9201 and 9201-B
	Oxygen - O ₂	9231
F	Carbon monoxide - CO	9229
	Methane - CH ₄	9232
	Ammonia – NH ₃	9233
	Liquefied Petroleum Gases: H ₂ , CH ₄ , ethanol, isobutene	9234
	Air pollutants 1: C ₄ H ₁₀ , CH ₃ CH ₂ OH, H ₂ , CO, CH ₄	9235
	Air pollutants 2: C ₆ H ₅ CH ₃ , H ₂ S, CH ₃ CH ₂ OH, NH ₃ , H ₂	9236
	Alcohol derivates: CH ₃ CH ₂ OH, H ₂ , C ₄ H ₁₀ , CO, CH ₄	9237

Figura 2.14 Compatibilidad socket-sensores..

Conserva las bondades tanto del core como de la shield: socket para conexión de panel solar, puerto USB para programación y todas las especificaciones técnicas anteriormente comentadas.

2.4 Elección del gateway

Para tener una visión clara de qué se pretende conseguir a nivel de red, se presenta aquí un esquema sencillo del modelo.

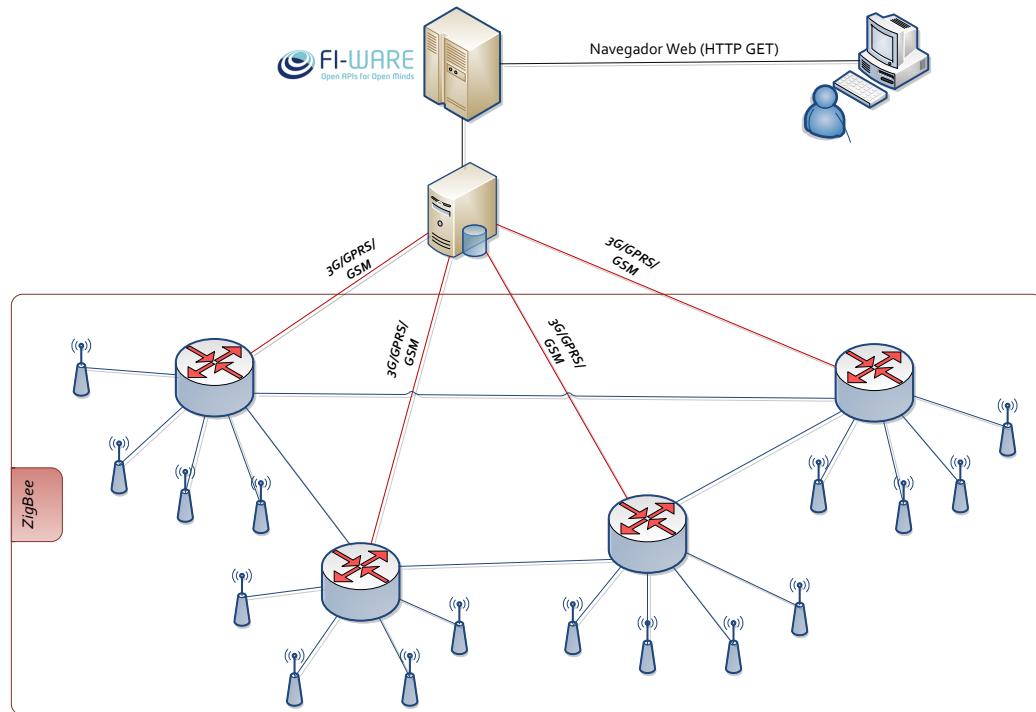


Figura 2.15 Modelo de red.

Los routers XBee actuarán, además, como gateways hacia Internet y encauzados estos hacia el servidor central, donde se encuentra la base de datos y el procesamiento necesario para la conexión a tiempo real con Fi-Ware. Existe, pues, la necesidad de desarrollar equipos electrónicos con capacidad para albergar dos módulos de comunicación. Su función básica será conseguir las tramas de lecturas que llegan hacia ellos por la red ZigBee y lanzarlos hacia el servidor central. Existen diferentes alternativas para construir esto de manera poco costosa en tiempo: utilizar núcleos Arduino o WaspMote. Anteriormente ya pudimos tener un esbozo de la diferencia en coste entre los núcleos WaspMote y Arduino, y algunos de los componentes que encarecían el primer modelo. A continuación se presenta un desglose de los equipos necesarios para construir los routers/gateways según cada plataforma.

Tabla 2.6 Presupuesto del gateway: modelo basado en WaspMote.

Componente	Unidades	Precio (euros/ud)
WaspMote + GPRS	1	158
Xbee Zigbee Pro SMA 5dBi	1	22.9
Expansion Radio Board	1	25
Batería recargable 2300 MAh	1	18
SIM M2M	1	(según tarifa)
TOTAL (sin IVA)		223.90 euros
TOTAL (21% IVA)		270.92 euros

Tabla 2.7 Presupuesto del gateway: modelo basado en Arduino.

Componente	Unidades	Precio (euros/ud)
Arduino Uno	1	20
Xbee Shield Module	1	15
Xbee Zigbee Pro SMA 5dBi	1	22.9
Arduino GSM Shield	1	69
Batería recargable 2300 MAh	1	18
SIM M2M	1	(según tarifa)
	TOTAL (sin IVA)	<i>144.90 euros</i>
	TOTAL (21% IVA)	<i>175.33 euros</i>

El modelo basado en núcleo Wasp mote resulta alrededor de un 35 % más caro. Debemos tener en cuenta que este núcleo incorpora otros componentes adicionales que no monta el núcleo Arduino y que conllevan un sobrecoste aproximado de 60 euros. Para el despliegue que pretendemos realizar, incorporar un acelerómetro, conector para panel solar o módulo para uSD no resulta necesario, por lo que el modelo basado en **Arduino** resulta el más conveniente económicoamente.

3 Desarrollo del software para dispositivos de la red

3.1 Primeros pasos con Wasp mote Smart Environment

3.1.1 IDE

El IDE o *Integrated Development Environment* es un entorno de programación compuesto por diferentes herramientas que ayudan a desarrollar determinadas aplicaciones. Se compone usualmente de un editor de código, un compilador, un depurador y una interfaz gráfica. El software desarrollado para cargar en plataformas Wasp mote, tanto para placas de desarrollo como para las Plug & Sense, debe ser compilado y cargado a través de la IDE de Wasp mote, basada en la IDE de Arduino. El desarrollo de esta IDE no sería posible sin el carácter *open-source* de la plataforma Arduino. Esta IDE incorpora barra de herramientas, editor de código, terminal de mensajes del compilador y monitor serie entre otras muchas.

Cualquier otra IDE que no fuese la de Wasp mote podría volver inestable el dispositivo, por lo que será esta la que utilicemos en el proceso de desarrollo y carga de programas. Libelium proporciona versiones de esta IDE para los sistemas operativos más comunes, así como las APIs para lectura de sensores y utilización de los módulos de comunicación de las diferentes boards.

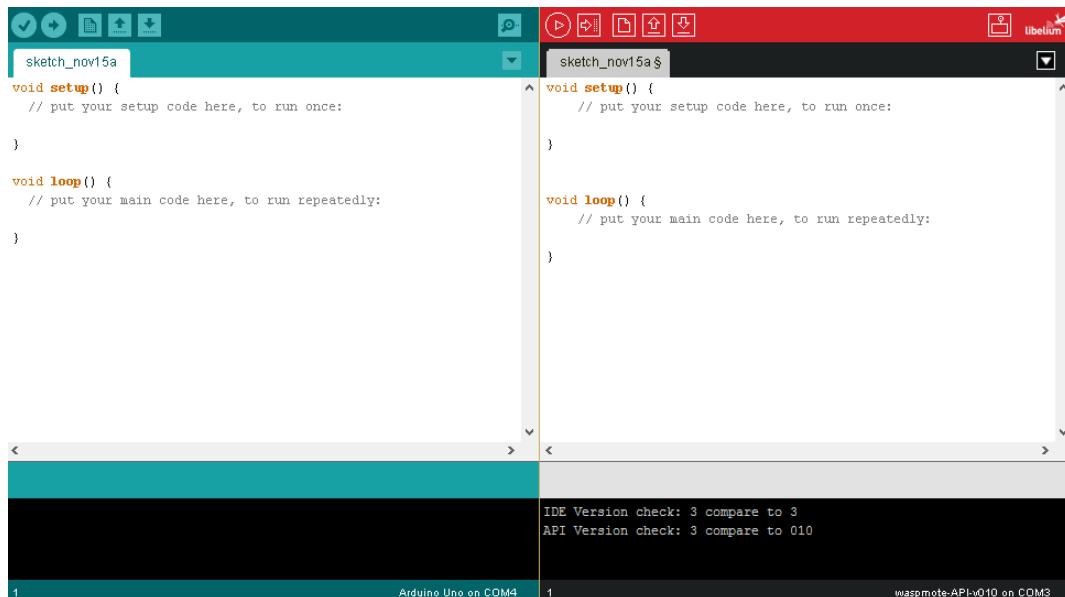


Figura 3.1 Arduino IDE / Wasp mote IDE.

Los programas serán cargados en los nodos a través del puerto USB que incorporan, como puede apreciarse a continuación.



Figura 3.2 Carga de programas en Waspmoté.

3.1.2 Lenguaje de programación

La programación de las Waspmoté es idéntica a la de las placas Arduino, por lo que hereda el lenguaje y todas las características asociadas. El software escrito para Arduino se denomina *sketch*, está basado en C y C++ y librerías estándar de este lenguaje (AVR Libc). Los *sketches* son convertidos a lenguaje máquina a través de un compilador tipo *gcc* propio de los microprocesadores Atmel (AVR) que incorporan las placas. Este código ya convertido, combinado con las librerías básicas de Arduino, contiene el fichero binario que será cargado en la memoria de programa del chip en la placa. Los *sketches* se dividen en tres partes: estructura, constantes/variables y funciones.

La estructura principal de un *sketch* en Arduino se compone de dos funciones principales: *setup()* y *loop()*. *Setup()* es la función llamada cuando comienza la ejecución del *sketch* y contiene todas las inicializaciones de variables, inicio de módulos de la placa mediante librería y seteo de pines como entrada/salida o encendido/apagado. Esta función sólo se ejecuta una vez, después de cada reseteo o encendido de la placa.

Loop() es, como su nombre indica, el bucle que se mantiene en ejecución y provee un control activo de la placa, provocando que esta responda ante cambios externos según se programe.

Las estructuras de control, operadores lógicos, sintaxis, formatos de variables y funciones básicas son idénticas a las de C/C++, por lo que para mayor información de aquí en adelante acudiremos a la página de referencia para programación de Arduino¹. Aquí se ofrece un ejemplo de programación típica en Arduino, enviando el estado de un botón situado en el pin 3 y monitorizado a través del puerto serie.

```
const int pin_Boton = 3;

void setup()
{
    Serial.begin(9600);
    pinMode(pin_Boton, INPUT);
}

\\Si se pulsa el boton,
\\se enviara estado por puerto serie
```

¹ <http://arduino.cc/en/pmwiki.php?n=Reference/HomePage>

```

void loop()
{
    if (digitalRead(pin_Boton) == HIGH)
        Serial.write('H');
    else
        Serial.write('L');

    delay(1000);
}

```

3.1.3 Calibración y puesta a punto de los sensores de polución

Consideraciones generales

En el interior de la Waspmove Plug & Sense Smart Environment se encuentra una *board* del estilo a la que se puede ver a continuación.

Tener una caracterización visual de ella facilitará el entendimiento de este apartado. Debe tenerse muy en cuenta que pueden existir diferencias significativas entre sensores de un mismo modelo, pues a nivel macroscópico podrían resultar iguales pero no así a nivel microscópico. Con el fin de construir una placa que pueda adaptarse por software al sensor que se coloque en ella, esta *board* incorpora resistencias de carga y etapas de amplificación regulables. Mediante la librería disponible para la *gas board* podremos configurar estos parámetros electrónicos. Tampoco es la misma alimentación para cada socket: 5V para cualquiera salvo para el socket 3B (1.8V) y para el socket 2B (2.5V).

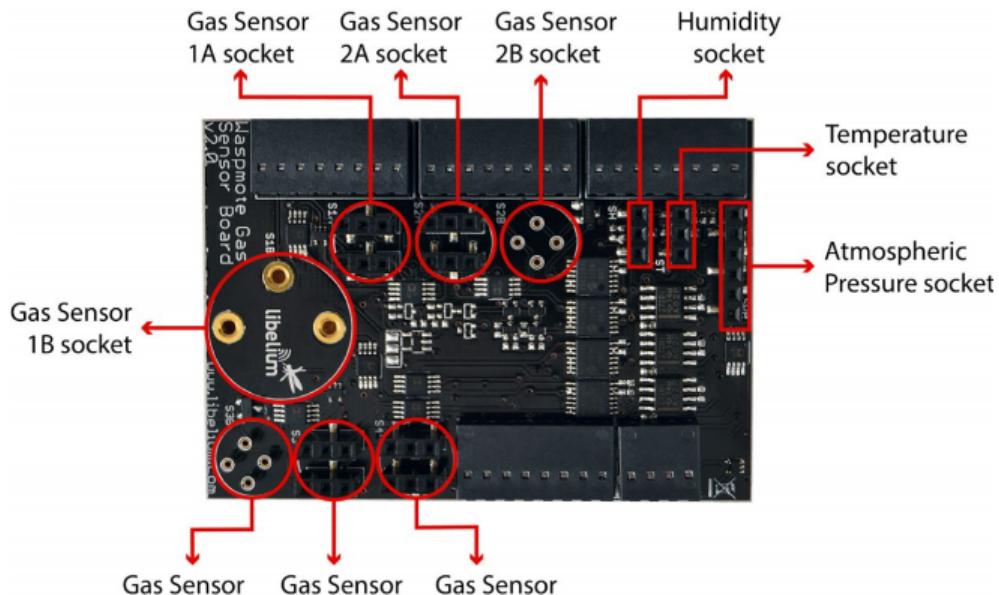


Figura 3.3 *Gas board* presente en el interior del dispositivo.

La sensibilidad del sensor dependerá además de las condiciones climáticas del entorno al que se exponga o si ha pasado por un largo periodo de inactividad. El cálculo de la resistencia del sensor, desde la que puede inferirse la concentración del gas que se pretende medir, puede ser realizado a través de la ecuación:

$$R_s = \frac{V_c * R_L}{V_{out}} - R_L$$

Siendo R_s la resistencia del sensor, V_c la tensión de alimentación, V_{out} la tensión de salida medida y R_L la resistencia de carga del socket.

Aspectos clave

Los principales factores a tener en cuenta deberán ser, por tanto, cómo configuramos las etapas de amplificación y resistencias de carga, cuál es el proceso de calibración a seguir y cómo convertimos el valor de voltaje o resistencia del sensor a concentración de gas.

Calibración

Una calibración precisa exige contar con un laboratorio en que reproducir diferentes condiciones en que se puedan dar ciertas concentraciones del gas a medir. Los diferentes valores medidos se traducirían en una curva de calibración mediante una aproximación matemática (logarítmica, por ejemplo). Se debe tener en cuenta que las diferencias microscópicas para cada sensor dan lugar a infinitas curvas de calibración. Para una aplicación que no exija una precisión extrema como la que estamos considerando, se puede acudir a la curva de calibración que ofrece el fabricante para cada tipo de sensor, que no es más que una media de comportamientos de sensores similares a nivel macro.

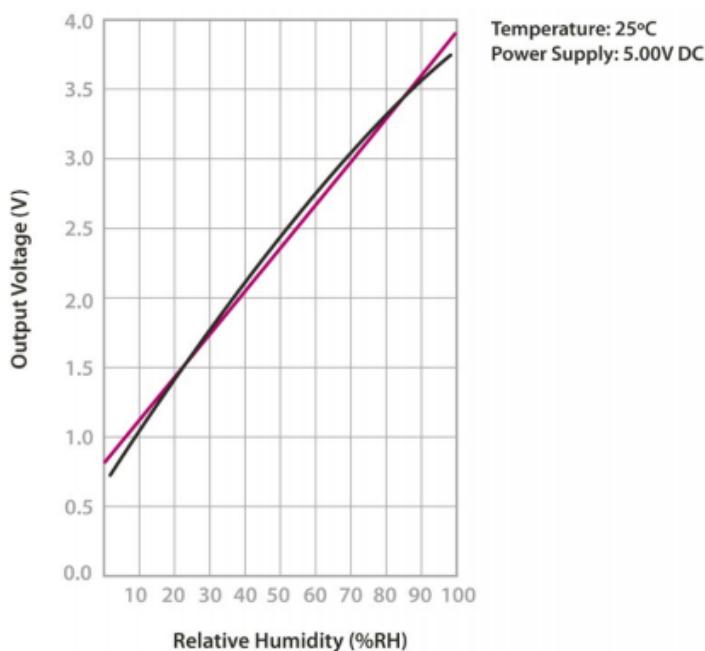


Figura 3.4 Ejemplo de curva de calibración: sensor de humedad Sencera 808HSV5.

Ajuste de ganancia y resistencia de carga

La ganancia y resistencia configurada para cada etapa dependerá de dos aspectos principales: el sensor conectado y las condiciones en que se vaya a situar el dispositivo para la aplicación. Esta configuración dependerá de lo fino que se quiera hilar para las lecturas. En una aplicación como ésta en que se pueda requerir producciones en masa que hagan inviable calibrar un gran número de sensores y actuar en consecuencia, seguiremos las reglas generales propuestas por el fabricante.

Para la etapa de amplificación la ganancia se fijará a 1, salvo para aplicaciones que requieran llevar el rango de medida del sensor al límite. Existen dos excepciones a esta regla: para los sensores de CO_2 y O_2 . Estos sensores requieren que el voltaje de salida sea amplificado para ajustarse al convertidor ADC de la WaspMote. En el caso del O_2 , deberá amplificarse con un factor de 100 y de entre 7 y 10 para el CO_2 .

Para el caso de la resistencia de carga no es tan simple por lo comentado anteriormente. Se requerirá un análisis basado en prueba-error para conseguir el valor correcto de resistencia, comenzando con el valor inicial más bajo según el *datasheet* del sensor y ajustándolo hasta conseguir el valor medio del rango del ADC (1.6V).

Asimismo, se recomienda calentar los sensores realizando lecturas continuas durante al menos 12 horas y no configurar valores de resistencia de carga inferiores a la mínima, ya que esto podría dañar los sensores.

Unidades de medición

Es importante caracterizar de manera precisa las unidades en que se proporcionan las lecturas para que ofrezcan información relevante. Existen numerosos estudios realizados por la OMS y otras entidades certificadas que aportan el límite de exposición de vapores químicos en el aire (TLV) mediante dos unidades principales: partes por millón y microgramos por metro cúbico. La medida en partes por millón solo puede considerarse si la sustancia existe como gas o vapor a temperatura y presión normales (25 °C y 1 atm). En estas condiciones, se relacionan de la forma:

$$TLV_{mg/m^3} = \frac{PM(g) * TLV_{ppm}}{24.45}$$

Siendo PM el peso molecular de una sustancia determinada en gramos y 24.45 el volumen en litros de un mol de gas cuando la temperatura es 25°C y la presión 1 atm. Si consideramos diferentes condiciones del entorno, la conversión de mg/m^3 a ppm no es tan inmediata:

$$V = \frac{RT}{P}$$

$$TLV_{mg/m^3} = \frac{P}{RT} * PM * TLV_{ppm}$$

Con R la constante ideal de los gases ideales ($0.082 \frac{atm * L}{K * mol}$), T la temperatura en grados Kelvin ($273 + T^{\circ}C$) y P la presión en mm Hg.

Conversión a unidades típicas

El valor de concentración de los gases en ppb o ppm (dependiendo del sensor) puede obtenerse mediante las curvas de calibración del fabricante, ya que no se realiza en este caso una calibración exhaustiva para cada sensor. Es necesario, aún así calcular la resistencia inicial del sensor, cuyo rango se nos facilita en el *datasheet*. Habrá que tener en cuenta, además, que factores ambientales como la temperatura o la humedad también pueden afectar a las lecturas. Se pueden encontrar dos tipos de gráficas en los *datasheets* de los sensores según el eje Y represente un ratio de resistencias o una diferencia de tensiones (para CO_2). Fijando un valor de tensión de referencia y calculando la diferencia de este con el valor de tensión otorgado por el sensor podremos tener una caracterización de la concentración del gas en un ambiente determinado.

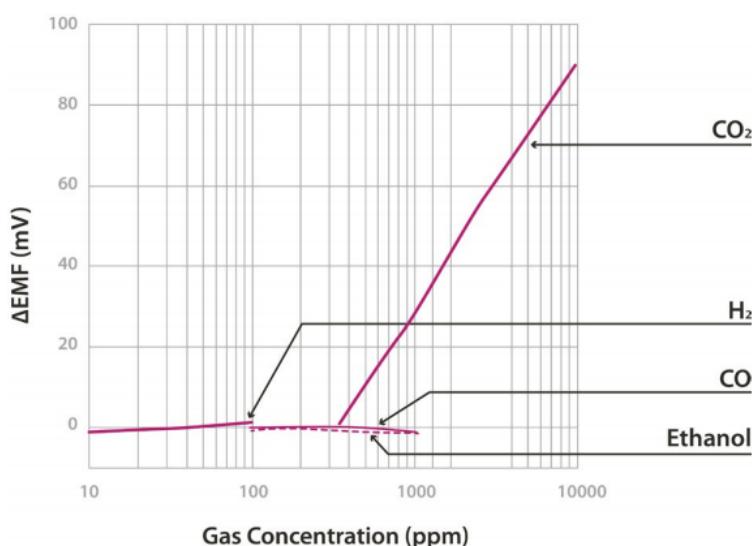


Figura 3.5 Ejemplo de curva de calibración: ΔEMF .

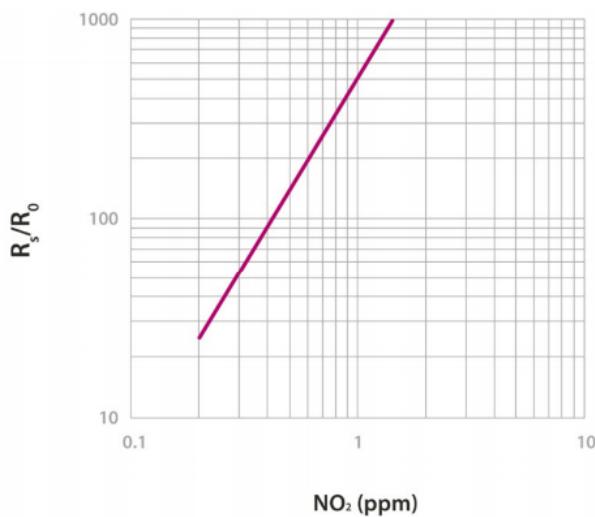


Figura 3.6 Ejemplo de curva de calibración: R_s/R_o .

3.2 Nodos sensores

3.2.1 Comportamiento básico del sistema

El software desarrollado para los dispositivos finales deberá ser robusto y eficiente en cuanto al consumo, además de cumplir con la funcionalidad que se pretende que tengan. Para ello, definimos algunas pautas:

- El estado normal del dispositivo será el de reposo. Para ello deberemos dormir al dispositivo tras las lecturas y despertarlo con la periodicidad que deseemos. En este caso apagaremos todos los módulos no necesarios durante el reposo y volveríamos a arrancarlos en cada lectura.
- Una lectura adecuada de los sensores implica el calentamiento y la calibración de estos para otorgar medidas tan precisas como sea posible. Aparece aquí un compromiso tiempo de calentamiento - consumo final del dispositivo que tendremos que tener en cuenta.
- Configuración del módulo de comunicación (XBee) para enviar las tramas contenedoras de lecturas a su gateway/router más próximo.
- Sería interesante llevar control de posibles errores en formación de paquetes, verificar conexión a la red ZigBee, lecturas de sensores o posibles transmisiones erróneas. Para monitorizar esto, inicializamos el puerto serie y habilitamos una variable booleana que defina si nos encontramos o no en modo depuración.
- Añadiremos alguna funcionalidad adicional para ganar robustez ante problemas que puedan surgir tras muchos ciclos de actividad del microprocesador (desbordamiento de buffers...).

3.2.2 Características destacables de la implementación

- Los sensores de gas necesitan entre 1 y 10 minutos para estabilizar la lectura, según el fabricante. Si bien debemos esperar un tiempo suficiente para el calentamiento de los sensores (el cual marcará aquel cuyo valor sea más alto), tomaremos varias medidas y ponderaremos para obtener un valor medio. La vida útil de las baterías viene marcada en gran medida por el tiempo que pasa encendido el dispositivo, por lo que para aquellos dispositivos que incorporen sensores con tiempos de calentamiento altos habrá que llegar a un compromiso en cuanto a la frecuencia de las lecturas, de manera que se puedan recargar las baterías adecuadamente y que esto no afecte al correcto funcionamiento.
- Para cada tipo de gas se requiere linealizar la curva de calibración en torno a un punto de trabajo conocido. Por ejemplo, para CO_2 se linealiza en torno a 335 ppm, el valor medio sobre el que oscilan estas medidas en un ambiente urbano.

3.3 Gateways

3.3.1 Comportamiento básico del sistema

- Este dispositivo deberá estar continuamente encendido esperando paquetes ZigBee procedentes de los nodos finales. Al incorporar un módulo 3G/GPRS (red móvil) tendrá picos de consumo en transmisión, y una batería típica no ofrece suficiente carga para alimentar ambos módulos, por lo que necesita estar alimentado continuamente. Pediremos que el procesado interno sea lo más sencillo posible.
- Configuración del módulo de comunicación (XBee) para recibir las tramas contenedoras de lecturas.
- Configuración del módulo 3G/GPRS para envío de lecturas al proxy.
- Control del proceso a través del puerto serie.

3.4 Conexión con Fi-Ware y almacenamiento en BBDD

3.4.1 Comportamiento del software

- Recogida de las lecturas procedentes de los gateways.
- Almacenamiento en base de datos.
- Conexión con el punto de entrada a Fi-Ware, en este caso Orion Context Broker, para reflejar esta medida en tiempo real.
- Añadir datos relevantes a la lectura, como la fecha y hora en que se mide, para que aparezca reflejado en Fi-Ware.

3.4.2 Lenguajes posibles para la implementación

Desde el *gateway* deberemos acceder a una máquina (PC) que actúe como servidor web, ejecute las operaciones exigidas y se comunique con Fi-Ware. Este equipo de la red es comúnmente conocido en informática como **proxy**. Aparecen diversos lenguajes (todos del lado servidor) en que se puede implementar este comportamiento, entre los que destacamos:

- **ASP.NET o Active Server Pages.** Es la tecnología desarrollada por Microsoft para crear páginas web dinámicas. El lenguaje es similar a Visual Basic Script con algunas ventajas específicas en entornos web. Está limitado a funcionar sólo en Microsoft Windows a través de un servidor tipo IIS.
- **PHP o Hypertext Pre-processor.** Es un lenguaje de programación para desarrollo web dinámico. El código es interpretado por un servidor web con módulo intérprete PHP (Apache). Es multiplataforma y permite conexión a diferentes tipos de bases de datos, tanto SQL (MySQL) como NoSQL (MongoDB). Es libre y está considerado uno de los lenguajes más potentes y con mayor comunidad de usuarios.
- **JSP o Java Server Pages.** Basado en Java, viene a ofrecer un comportamiento similar a los anteriores. Para desplegar webs escritas en JSP se requiere un servidor web compatible, como *Apache Tomcat* o *Jetty*.

Por la versatilidad que ofrece, la abundante documentación existente y aprovechando que ya conocemos algo de PHP, será este el lenguaje que utilizaremos para implementar el middleware.

3.4.3 Interacción con Orion Context Broker

Como ya comentábamos en capítulos anteriores, el Orion Context Broker es una implementación de un servidor NGSI 9/10 para administración de información de contexto. Se despliega sobre una máquina Linux, particularmente con distribución Debian, y se ejecuta como un servicio "demonio". Queda accesible a través de su API REST en el puerto 1026 de manera que podemos interactuar con él a través de llamadas HTTP conociendo la IP de la máquina en la que se ejecuta y dirigiéndolas al puerto comentado.

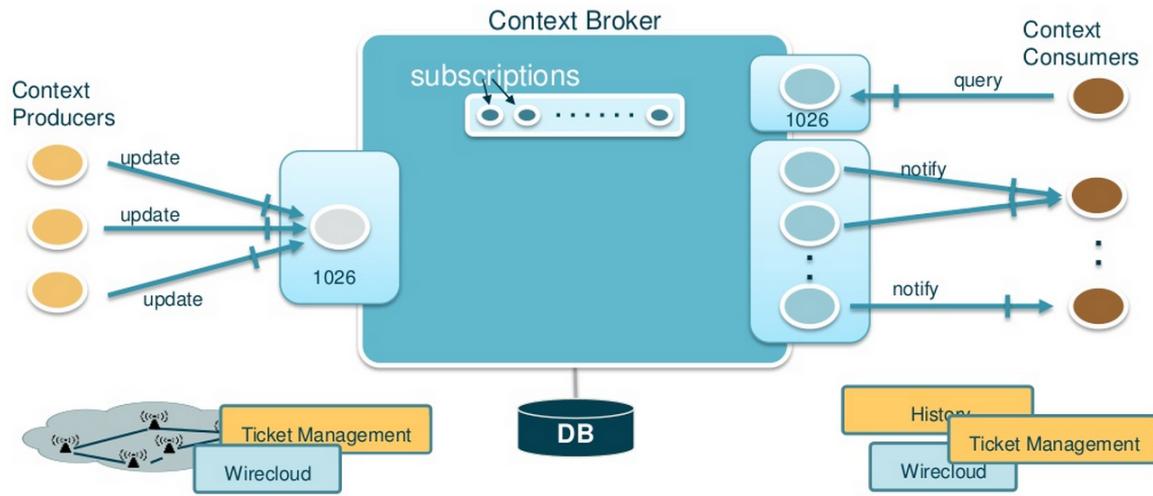


Figura 3.7 Orion Context Broker: interacciones..

Si bien REST (*Representational State Transfer*) originalmente se refería a un conjunto de principios de arquitectura software para sistemas en la red, actualmente se utiliza para describir cualquier tipo de interfaz web que utilice XML y HTTP, viiniendo a mejorar arquitecturas similares para *web services* como SOAP. Hoy en día es muy usual encontrar APIs REST para desarrolladores en las webs de contenido de las grandes marcas (Facebook, Amazon, Twitter...). Las operaciones que nos permite esta API, en el caso particular del Orion Context Broker son las NGSI 9/10 definidas en el estándar que posibilitan la creación, actualización y suscripción a cambios en entidades del contexto:

- updateContext
- queryContext
- subscribeContext
- updateContextSubscription
- unsubscribeContext

El *payload* de la llamada HTTP puede estar escrito en XML (Extensible Markup Language), un lenguaje de marcas para intercambio de información, o JSON (JavaScript Object Notation), un formato más ligero que el anterior con el mismo cometido.

Creando y consultando entidades

Orion Context Broker se encontrará en permanente estado de escucha, e inicialmente la base de datos mongoDB se encontrará vacía. Para que OCB maneje la información que llega acerca de un cierto dispositivo de la red, deberemos instanciar la entidad correspondiente asignando parámetros iniciales, que más tarde serán actualizados por medio de suscripción a esta entidad. Vamos a crear una entidad *Habitación1* de tipo *Habitación* en la que se dispondrán dos sensores: *temperatura* y *humedad*. Inicialmente estos parámetros tendrán el valor que creyésemos oportuno, hasta que se reciban actualizaciones por parte de los dispositivos.

```
<?xml version="1.0" encoding="UTF-8"?>
<updateContextRequest>
  <contextElementList>
    <contextElement>
      <entityId type="Habitacion" isPattern="false">
        <id>Habitacion1</id>
      </entityId>
      <contextAttributeList>
        <contextAttribute>
          <name>temperatura</name>
          <type>centigrados</type>
          <contextValue>23</contextValue>
        </contextAttribute>
      </contextAttributeList>
    </contextElement>
  </contextElementList>
</updateContextRequest>
```

```

</contextAttribute>
<contextAttribute>
  <name>presion</name>
  <type>mmHg</type>
  <contextValue>720</contextValue>
</contextAttribute>
</contextAttributeList>
</contextElement>
</contextElementList>
<updateAction>APPEND</updateAction>
</updateContextRequest>

```

La directiva final *< updateAction >* puede hacer referencia a dos métodos: *APPEND* y *UPDATE*. La creación de un nuevo dispositivo se realiza mediante *APPEND* mientras que las posteriores actualizaciones se construirán mediante el método *UPDATE*. En formato JSON resulta algo menos verboso en ocasiones, por lo que presentamos un ejemplo de esto mismo:

```

{
  "contextElements": [
    {
      "type": "Habitacion",
      "isPattern": "false",
      "id": "Habitacion1",
      "attributes": [
        {
          "name": "temperatura",
          "type": "centigrados",
          "value": "20"
        },
        {
          "name": "presion",
          "type": "mmHg",
          "value": "720"
        }
      ]
    },
    "updateAction": "APPEND"
  }
}

```

En este momento, Orion Context Broker creará la entidad en su base de datos interna, asignará los valores a los parámetros y devolverá una respuesta afirmativa si el *parsing* ha resultado correcto.

```

<?xml version="1.0"?>
<updateContextResponse>
<contextResponseList>
<contextElementResponse>
<contextElement>
<entityId type="Habitacion" isPattern="false">
  <id>Habitacion1</id>
</entityId>
<contextAttributeList>
<contextAttribute>
  <name>temperatura</name>
  <type>centigrados</type>
  <contextValue/>
</contextAttribute>
<contextAttribute>
  <name>presion</name>
  <type>mmHg</type>
  <contextValue/>
</contextAttribute>
</contextAttributeList>
</contextElement>
<statusCode>
  <code>200</code>
  <reasonPhrase>OK</reasonPhrase>
</statusCode>
</contextElementResponse>

```

```
</contextResponseList>
</updateContextResponse>
```

Montando el payload para inserción de dispositivos de la red

En nuestro caso, deberemos crear entidades de tipo *SensorGas* (o un nombre similar) y asignar un *ID* a cada una para poder referenciarlas y actualizar la información. Proporcionaremos los parámetros:

Tabla 3.1 Parámetros a los que suscribir el Context Broker..

Parámetro	Tipo	Información
CO2	Partes por millón	Medición del nivel de CO2 presente en una zona.
NO2	Partes por millón	Medición del nivel de NO2 presente en una zona.
O3	Partes por millón	Medición del nivel de O3 presente en una zona.
Latitud	WGS84	Primera coordenada geográfica (estática).
Longitud	WGS84	Segunda coordenada geográfica (estática).
Dirección	Dirección postal	Calle en la que se encuentra el dispositivo.
Fecha y hora	ISO-8601	Visual de última lectura en Fi-Ware.

Parámetros como la longitud y la latitud pueden fijarse inicialmente y no modificarse a no ser que se asigne otra localización al dispositivo. Por otro lado, existen diferentes APIs vinculadas con mapas geográficos para proveer una dirección postal a partir de coordenadas geográficas. Teniendo en cuenta que utilizaremos Google Maps para el posicionamiento haremos uso de la API asociada a éste. Fecha y hora serán obtenidas del servidor intermedio por no poseer RTC los dispositivos de la red. Mediante la función *time()* de PHP puede conseguirse esto.

Lo que implementaremos en este punto, por tanto, será el parseo de todos los datos vinculados a un dispositivo sobre un *payload* del tipo al que hemos visto. Todo esto se procesará de manera automática cuando llegue una lectura de los sensores de un determinado dispositivo al servidor. La llamada HTTP hacia el Orion Context Broker para crear una entidad del tipo *SensorGas* tendría una forma similar a la siguiente.

```
<?xml version="1.0" encoding="UTF-8"?>
<updateContextRequest>
  <contextElementList>
    <contextElement>
      <entityId type="SensorGas" isPattern="false">
        <id>387243712</id>
      </entityId>
      <contextAttributeList>
        <contextAttribute>
          <name>BAT</name>
          <type>%</type>
          <contextValue>70</contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>CO2</name>
          <type>ppm</type>
          <contextValue>335</contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>NO2</name>
          <type>ppm</type>
          <contextValue>0.025</contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>O3</name>
          <type>ppm</type>
          <contextValue>0.025</contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>latitude</name>
          <type>coords</type>
        </contextAttribute>
      </contextAttributeList>
    </contextElement>
  </contextElementList>
</updateContextRequest>
```

```

<contextValue>37.394742</contextValue>
</contextAttribute>
<contextAttribute>
<name>longitude</name>
<type>coords</type>
<contextValue>-5.983526</contextValue>
</contextAttribute>
<contextAttribute>
<name>date</name>
<type>ISO-8601</type>
<contextValue>20/11/2014 12:24:37</contextValue>
</contextAttribute>
<contextAttribute>
<name>address</name>
<type>street</type>
<contextValue>Calle Maria Auxiliadora, 25 41003 Sevilla</contextValue>
</contextAttribute>
</contextAttributeList>
</contextElement>
</contextElementList>
<updateAction>APPEND</updateAction>
</updateContextRequest>

```

3.4.4 Inserción de medidas en una base de datos MySQL

MySQL es un sistema de gestión de bases de datos relacional, de código abierto y muy extendido, que permite desplegar de manera sencilla tablas de datos estructuradas en registros y campos. A través de phpMyAdmin, una herramienta software libre escrita en PHP, podemos gestionar una base de datos de este tipo de manera gráfica, creando bases de datos, tablas o ejecutando órdenes SQL como si de una consola de comandos se tratase.

Tabla	Acción	Registros	Tipo	Cotejamiento	Tamaño	Residuo a depurar
bluetoothData		0	MyISAM	utf8_unicode_ci	1.0 KB	-
currentSensors		0	MyISAM	latin1_swedish_ci	2.7 KB	752 Bytes
encryptionData		0	MyISAM	latin1_swedish_ci	1.0 KB	-
gpsData		258,223	MyISAM	utf8_unicode_ci	9.4 MB	-
meshlium		1	MyISAM	latin1_swedish_ci	2.1 KB	48 Bytes
pfc		238	MyISAM	latin1_swedish_ci	11.7 KB	-
sensorParser		3,066	MyISAM	latin1_swedish_ci	129.3 KB	-
sensors		85	MyISAM	latin1_swedish_ci	6.5 KB	-
tokens		2	MyISAM	latin1_swedish_ci	2.3 KB	188 Bytes
users		1	MyISAM	latin1_swedish_ci	2.2 KB	160 Bytes
waspmote		1	MyISAM	latin1_swedish_ci	2.6 KB	552 Bytes
wifiScan		0	MyISAM	latin1_swedish_ci	1.0 KB	-
zigbeeData		498	MyISAM	utf8_unicode_ci	32.9 KB	-
13 tabla(s)	Número de filas	262,115	MyISAM	latin1_swedish_ci	9.6 MB	1.7 KB

↓ Marcar todos/as / Desmarcar todos / Marcar las tablas con residuo a depurar Para los elementos que están marcados:

Vista de impresión Diccionario de datos

Crear nueva tabla en la base de datos MeshliumDB

Nombre: Número de campos:

Figura 3.8 Interfaz web phpMyAdmin.

Asimismo, el lenguaje PHP define una serie de APIs para gestión de bases de datos MySQL. Utilizaremos una de estas APIs para conectar a una base de datos e insertar las lecturas de los sensores en una tabla. Posteriormente podremos utilizar este almacén de datos para mostrar un histórico de las lecturas desde Fi-Ware. A continuación se muestra un pequeño ejemplo acerca de cómo se abriría una conexión con una base de datos, recabando toda la información de una tabla o insertando nuevas filas de datos.

```

//Parametros de conexion a BD:
$DATABASE = 'MeshliumDB';
$TABLE = 'pfc';
$IP = 'IP_BBDD';
$USER = 'root';
$PASS = 'pass';

// Conectar con el servidor de base de datos
$conexion = mysql_connect ($IP, $USER, $PASS)
    or die ("No se puede conectar con el servidor");

// Seleccionar base de datos
mysql_select_db ($DATABASE)
    or die ("No se puede seleccionar la base de datos");

// Realizar una consulta MySQL
$query = 'SELECT * FROM pfc';
$result = mysql_query($query) or die('Consulta fallida: ' . mysql_error());

// Insertando una medida de los sensores
$instruccion = "insert into pfc (ID, BAT, CO2, NO2, O3) VALUES ('".$id_secret."','".$$sensorValue[0]."','".$$sensorValue[2]."',";
    ".$$sensorValue[3].",'".$$sensorValue[4]."')";
$consulta = mysql_query ($instruccion, $conexion) or die ("Fallo de insercion en BBDD");

```

Una vez hayamos implementado este *web service* ya tendremos lista la recepción de tramas procedentes de los sensores. Las lecturas se desgranarán, insertando cada una en duplas parámetro-valor en la base de datos y posteriormente serán enviadas hacia Orion Context Broker para que queden disponibles en Fi-Ware.

4 Desarrollo de aplicaciones en Fi-Ware

4.1 Apartados de Fi-Lab

4.1.1 Cloud

Aquí se ofrecen *GEs* útiles para desplegar una infraestructura de almacenamiento en la nube (*cloud hosting*) sobre la que desarrollar y administrar las aplicaciones y servicios sobre Fi-Ware. Nuestra instancia de Orion Context Broker correrá sobre una máquina que despleguemos corriendo el sistema operativo que deseemos según las imágenes que se ofrecen de estos. Existen diversas opciones, cada una con un propósito diferente. En la siguiente imagen se aprecian algunas de ellas, así como el panel de navegación con los distintos *GEs*.

The screenshot shows the Fi-Ware Cloud interface with the 'Images' tab selected. On the left, there is a navigation sidebar with sections for Project Name (m2msense), Blueprint, Region (Spain), Compute, and Storage. The 'Compute' section has 'Instances', 'Images' (which is currently selected and highlighted in blue), 'Flavors', 'Security', and 'Snapshots'. The 'Storage' section has 'Containers'. The main content area displays a table titled 'Images' with the following data:

Name	Type	Status	Visibility	Container Format...	Disk Format	Actions
CentOS-7-x64	baseimages	active	public	OVF	QCOW2	<button>Launch</button>
CentOS-6.3-x86_64	baseimages	active	public	OVF	QCOW2	<button>Launch</button>
CentOS-6.5-x64	baseimages	active	public	OVF	QCOW2	<button>Launch</button>
Ubuntu Server 14.04.1 (x64)	baseimages	active	public	OVF	QCOW2	<button>Launch</button>
Ubuntu12.04-server-x86_64	baseimages	active	public	OVF	QCOW2	<button>Launch</button>
wstore-img	fiware:apps	active	public	OVF	QCOW2	<button>Launch</button>
wirecloud-img	fiware:apps	active	public	OVF	QCOW2	<button>Launch</button>
repository-image-R3.2	fiware:apps	active	public	AMI	AMI	<button>Launch</button>
marketplace-ri-R2.3	fiware:apps	active	public	AMI	AMI	<button>Launch</button>

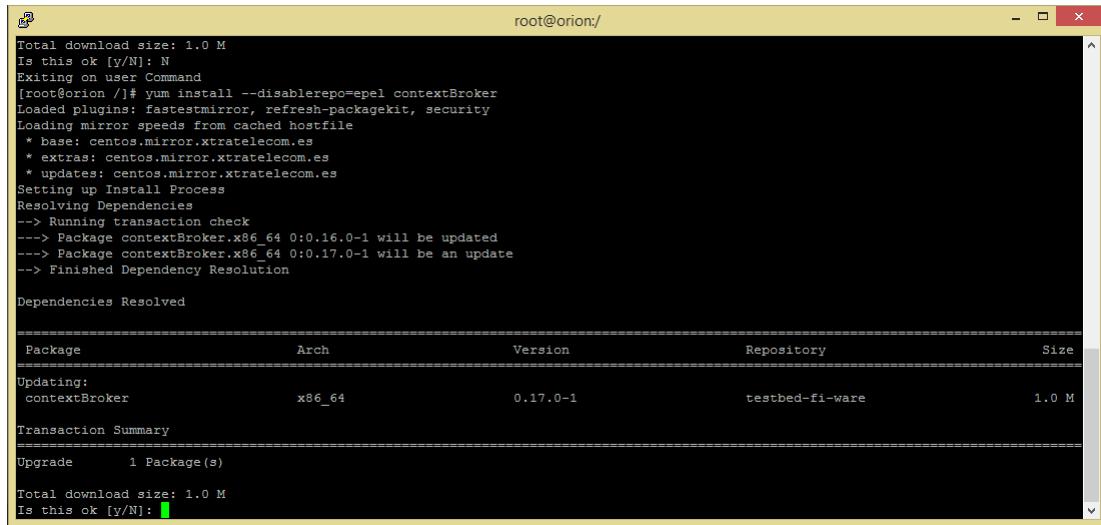
At the bottom right of the table, it says 'Displaying 32 items'.

Figura 4.1 Cloud: el apartado para *cloud hosting* en Fi-Ware.

En este momento no precisamos conocer en profundidad este apartado, por lo que simplemente utilizaremos aquello que nos sea útil para nuestro propósito final.

Desplegando una instancia propia de Orion Context Broker

Para la utilización de Orion Context Broker en un determinado proyecto, es posible emplear la instancia Orion de Fi-Lab (accesible desde <http://orion.lab.fi-ware.org:10026/>) o desplegar una propia. En este caso, como pretendemos que el proyecto sea privado y nadie pueda tener acceso a nuestros datos, desplegaremos una propia. Para ello seguiremos los pasos descritos en la wiki de Fi-Ware, desplegando una máquina con sistema operativo CentOS e instalando todos los paquetes necesarios a través de un acceso SSH.



```

Total download size: 1.0 M
Is this ok [y/N]: N
Exiting on user Command
[root@orion ~]# yum install --disablerepo=epel contextBroker
Loaded plugins: fastestmirror, refresh-packagekit, security
Loading mirror speeds from cached hostfile
 * base: centos.mirror.xtratelecom.es
 * extras: centos.mirror.xtratelecom.es
 * updates: centos.mirror.xtratelecom.es
Setting up Install Process
Resolving Dependencies
--> Running transaction check
--> Package contextBroker.x86_64 0:0.16.0-1 will be updated
--> Package contextBroker.x86_64 0:0.17.0-1 will be an update
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch      Version       Repository      Size
=====
Updating:
contextBroker    x86_64   0.17.0-1    testbed-fi-ware  1.0 M

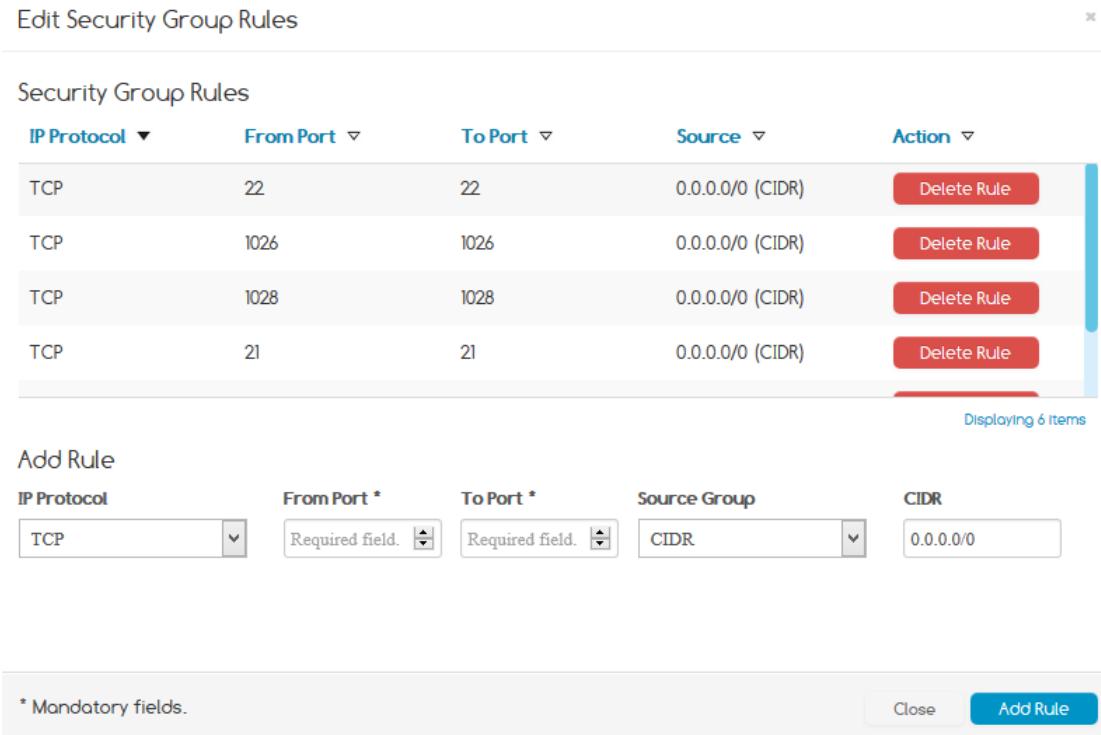
Transaction Summary
=====
Upgrade      1 Package(s)

Total download size: 1.0 M
Is this ok [y/N]: 

```

Figura 4.2 Instalando Orion Context Broker.

Necesitaremos conocer Linux y algunos comandos de administración de MongoDB. Además, para añadir seguridad a la máquina, solo abriremos los puertos necesarios y requeriremos una clave encriptada para acceso mediante SSH.



IP Protocol	From Port	To Port	Source	Action
TCP	22	22	0.0.0.0 /0 (CIDR)	<button>Delete Rule</button>
TCP	1026	1026	0.0.0.0 /0 (CIDR)	<button>Delete Rule</button>
TCP	1028	1028	0.0.0.0 /0 (CIDR)	<button>Delete Rule</button>
TCP	21	21	0.0.0.0 /0 (CIDR)	<button>Delete Rule</button>

Displaying 6 items

Add Rule

IP Protocol	From Port *	To Port *	Source Group	CIDR
TCP	Required field.	Required field.	CIDR	0.0.0.0 /0

* Mandatory fields.

Figura 4.3 Abriendo puertos y creando un grupo de seguridad.

Nuestra instancia de Orion quedaría así activa y accesible a través de la IP que se nos asigna.

Instances

	Instance Name	IP Address	Size	Keypair	Status	Task	Power State...	Actions
Orion	10.0.4.112 130.206.82.151		M2M		ACTIVE	None	RUNNING	

Displaying 1 item

Info: Connected to project m2msense [ID 000000000000000000000000000000004263]

Figura 4.4 Orion Context Broker desplegado y accesible.

4.1.2 Mashup

Wirecloud es una plataforma open-source para creación de aplicaciones masivas que forma parte de Fi-Ware siendo la implementación de referencia del **Application Mashup GE**. Está basada en interfaces conocidas como *widgets* que permiten conectar diferentes procesos y servicios web de fuentes diferentes. La plataforma está orientada a que usuarios sin conocimientos de programación sean capaces de componer interfaces web de consulta de manera sencilla. Este apartado es el realmente interesante, ya que nos permitirá desplegar las aplicaciones que se nos ocurra desarrollar. El endpoint del Mashup sobre Fi-Ware queda accesible a través de <https://mashup.lab.fi-ware.org>.

Architecture

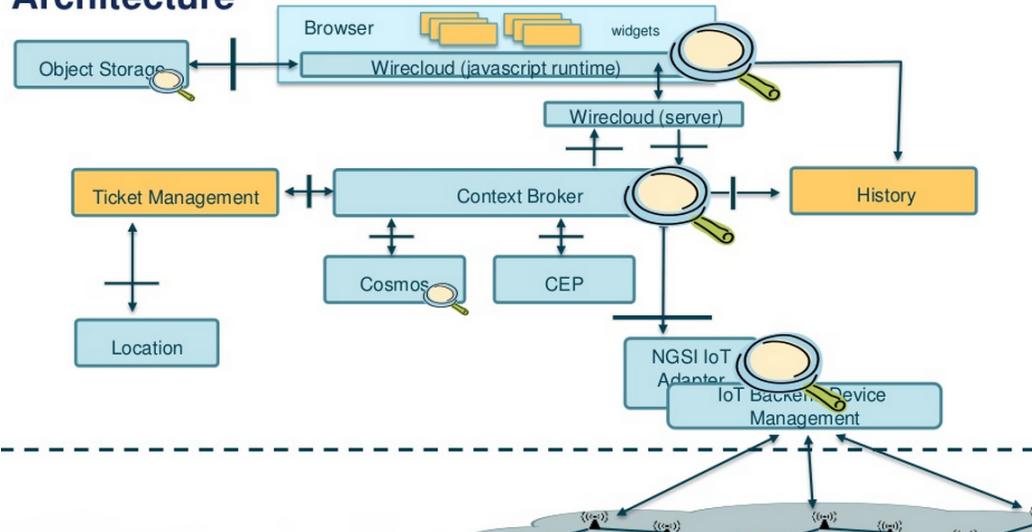


Figura 4.5 Wirecloud: capa más alta de la arquitectura Fi-Ware.

Wirecloud ofrece un catálogo de widgets y operadores que pueden ser compartidos por la comunidad (número reducido hoy en día). De esta forma, desarrolladores y empresas pueden comercializar aplicaciones o difundirlas para un uso público.

Las tecnologías que se han utilizado para su implementación son meramente tecnologías web: python

en lado servidor (back) y HTML/CSS/Javascript en el lado cliente (front). Es software libre y puede ser redistribuida y modificada bajo los términos de licencia *GNU Affero General Public License*.

Creando aplicaciones masivas: *wiring*

Un widget define tanto un back-end (motor) como un front-end (visual), no así un operador, que solo implementa el lado back-end. Esto es, un widget típico podría ser un mapa y un operador un pequeño proceso que consulte una base de datos y genere un objeto que contenga la información de la lectura. Este objeto debería ser consumido por otro widget u operador cableando (*wiring*) ambos en el Mashup.

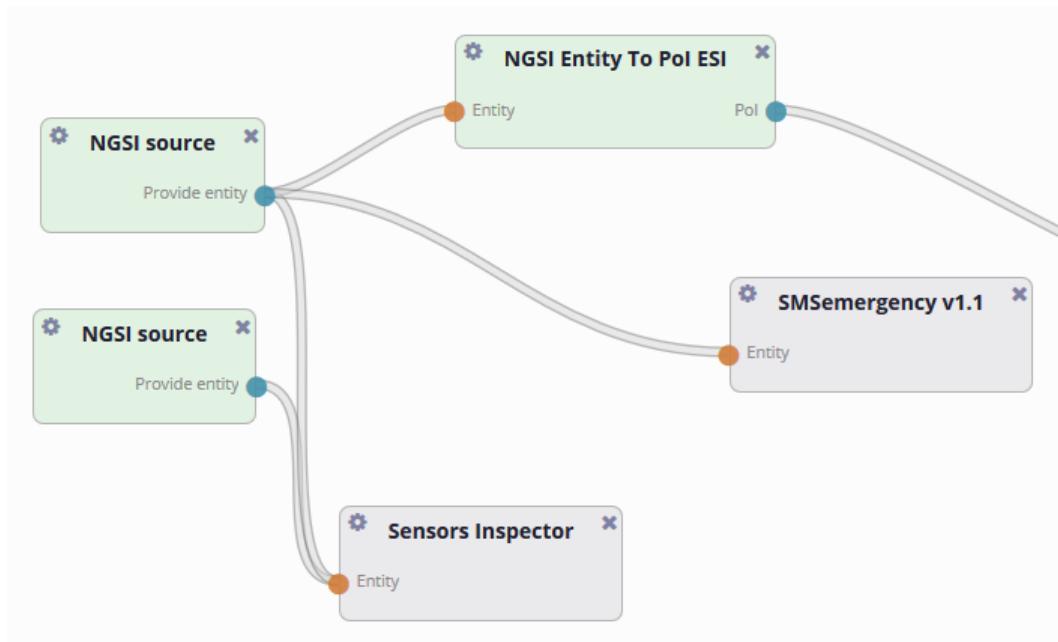


Figura 4.6 Interconexión de widgets en el *wiring*.

En la imagen anterior, el operador *NGSI Source* define un proceso de backend por el que se ejecuta una conexión NGSI con el Orion Context Broker mediante el que recaba la información de las entidades que se deseen. Esto no es más que una serie de consultas reiterativas a la mongoDB de la máquina que desplegamos en el Cloud. A través del cable, la salida de este operador va hacia diferentes widgets que puedan alimentarse de esta información.

Un ejemplo sencillo de aplicación masiva podría ser un mapa que muestre información a tiempo real de los sensores desplegados en la ciudad, así como su posición exacta. En la solución que ofrecemos más adelante, esta aplicación requiere un operador que conecte con la mongoDB de Orion, recabe las diferentes entidades de sensores y las mande hacia otro operador. El siguiente operador prepararía la información para un widget cuya visual principal fuese un mapa de Google Maps sobre el que se muestra el emplazamiento de los sensores y la información asociada.

Creando widgets y operadores

Si bien nos harán falta algunos widgets y operadores ya existentes (sobre todo los que definen procesos de backend), lo realmente interesante será crear nuevas aplicaciones que puedan ser utilizadas por la comunidad o que sirvan de base para futuros proyectos.

Los widgets y operadores se ejecutarán en la plataforma, a la cual se accede mediante un navegador web. Los widgets, al poder ofrecer la visual que necesitemos, podrán ser programados como si de una web se tratase, utilizando HTML/CSS. La lógica de control se la otorgaremos mediante Javascript. Por otra parte, los operadores sólo definen procesos lógicos, por lo que se deberán implementar íntegros en este último lenguaje. Wirecloud posee una API Javascript además para tratar los eventos de entrada/salida de datos a través del *wiring* y otros procesos, por lo que también nos será de utilidad en este caso.

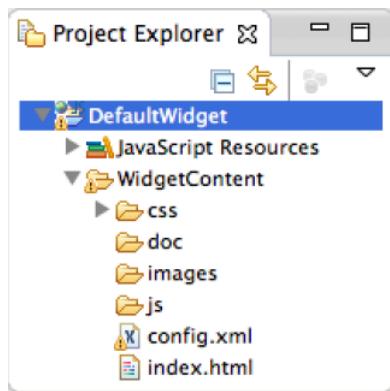


Figura 4.7 Estructura básica (gráfica) de un widget.

Tabla 4.1 Estructura básica de un widget (.wgt).

config.xml	Información básica del widget en lenguaje XML: entradas/salidas, preferencias...
index.html	Vista principal - Referencias a ficheros JS y CSS.
/js	Contenedor de ficheros Javascript.
/css	Contenedor de ficheros CSS.
/images	Directorio para imágenes.
/doc	Documentación para su uso.

Esta estructura básica de ficheros se comprimen en un fichero .zip, se renombra como .wgt y es este fichero el que se incluye en los recursos propios del usuario en Fi-Ware si está bien construido.

4.1.3 Store

No requiere mayor reseña, pues se trata del catálogo de widgets y operadores que quedan accesibles mediante pago o de manera gratuita al usuario final. Directamente se pueden desplegar sobre un *workspace* en el Mashup y modificarse o utilizarse como se deseé.

Figura 4.8 Marketplace en Fi-Lab.

4.1.4 Account

Aquí se definen algunos datos identificativos del usuario o concesión de permisos para aplicaciones que necesiten utilizar Fi-Lab. Un ejemplo de esto sería permitir el desarrollo de widgets en un editor de código (como Eclipse) de forma que resulte directo hacer pruebas sobre la plataforma a través de éste.

4.1.5 Data

Una de las partes más interesantes de Fi-Ware hacia el futuro. Un portal basado en CKAN (Comprehensive Knowledge Archive Network) que define una plataforma de Open Data, en la que los desarrolladores y usuarios pueden almacenar datos de interés sobre su ciudad que pudiesen ser utilizados por otros usuarios para experimentación o algún desarrollo de aplicación concreta.

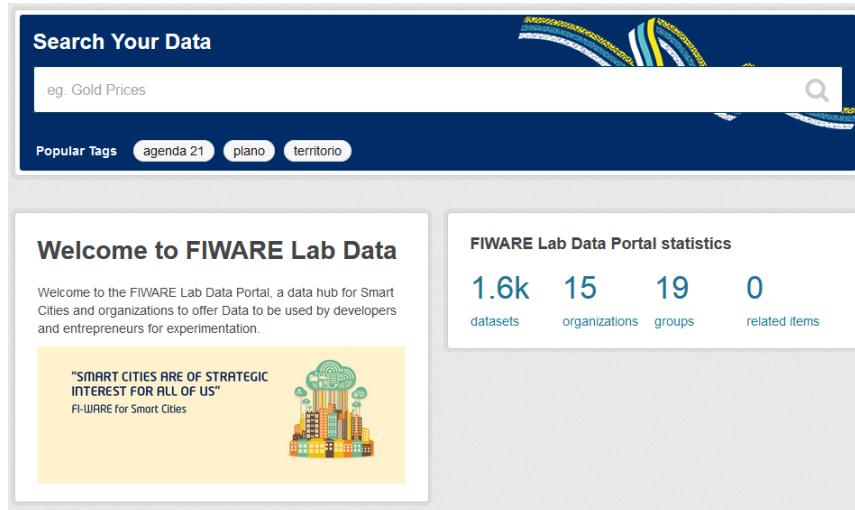


Figura 4.9 Marketplace en Fi-Lab.

4.2 Aplicaciones desarrolladas

Salvo la aplicación del mapa, basada en una implementación sobre Google Maps presente en el catálogo de Fi-Ware, todas las demás han sido concebidas y desarrolladas desde cero. Aún así, esta aplicación ha sido mejorada para ofrecer una visualización mucho más adecuada acerca de los sensores, mejorando además el comportamiento en cuanto a eficiencia y rendimiento del widget.

4.2.1 Mapa

Aprovechamos la implementación del widget *Map Viewer* presente en el Marketplace para adaptarla a nuestras necesidades. Inicialmente, este widget base no es más que un mapa típico de *Google Maps* sobre el que aparecerán las entidades geoposicionadas. La visualización de un sensor desplegado es idéntica a la que obtendríamos posicionando un edificio o elemento del mapa interactivo de Google.

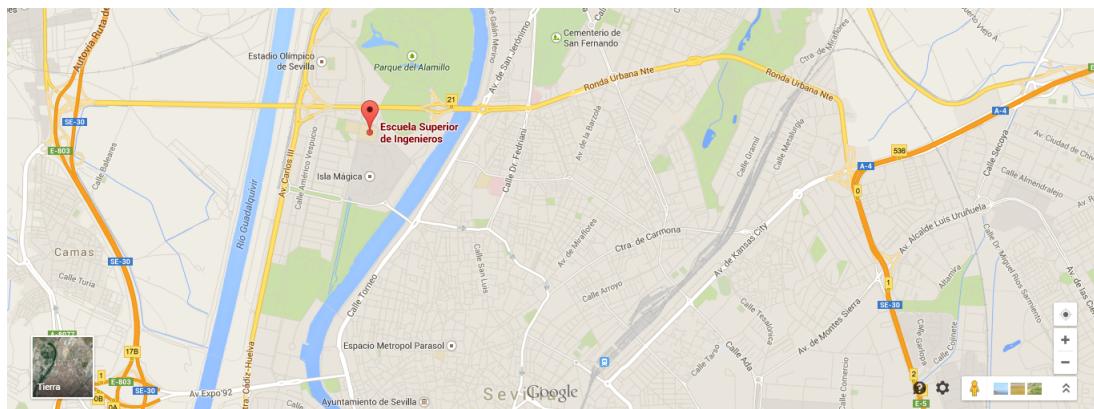


Figura 4.10 Visualización de puntos de interés en Google Maps.

Presentaremos el marcador de manera diferente, dando un toque personal y adecuado al proyecto. Esto no es más que referenciarle una nueva imagen. Al pulsar sobre el dispositivo, deberá abrirse una ventana emergente sobre la que aparezcan los datos relativos a éste.

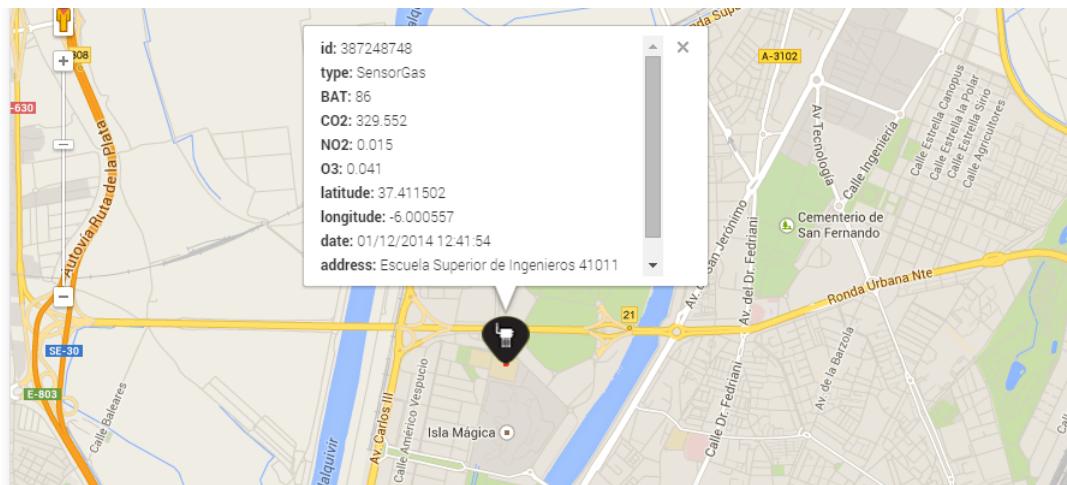


Figura 4.11 Visualización inicial de sensores en Google Maps.

Lo que resultaría interesante conseguir sobre esta ventana emergente (capa) es una visualización personalizada sobre el tipo de entidad en concreto. Además, podríamos indicar algo más de información relevante, tal como mostrar un ícono diferente en caso de una notificación importante (p. ej. un nivel bajo de batería). El resultado final es el que a continuación se muestra.

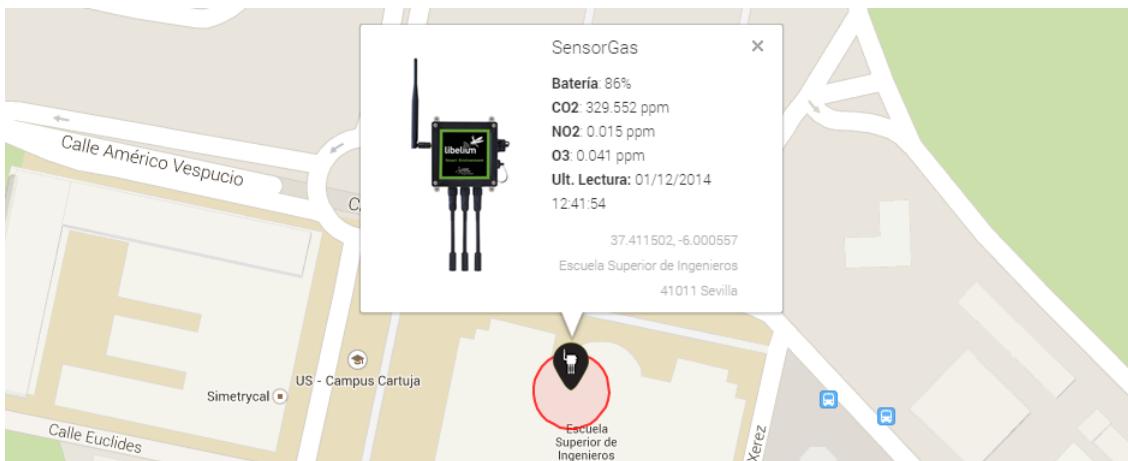


Figura 4.12 Visualización final de sensores en Google Maps.

4.2.2 Histórico de lecturas

Teniendo en cuenta que hemos desarrollado el software necesario para registrar los datos generados en una base de datos y tenemos posibilidad de acceder a ellos, implementar un motor de back-end sobre una aplicación de Fi-Ware para recabar las lecturas y presentarlas como datos históricos haría que nuestra aplicación tuviese un mayor impacto. Esto es lo que se describe a continuación. A través de un motor PHP, obtenemos las lecturas y las presentamos sobre gráficas Javascript en Fi-Ware. Construir esta aplicación de manera genérica, y no concebida para un uso particular, permite que pueda ser utilizada para diferentes fines y no sólo para una aplicación de sensórica, como es nuestro caso. Así pues, esta es la filosofía que se ha seguido para el desarrollo. Las gráficas desarrolladas son *responsive* y ofrecen información de las lecturas parametrizada en: *fecha, valor*. Se han utilizado modelos similares a las gráficas open-source Chart.js (<http://www.chartjs.org/>).

El primer tipo de gráfica que se nos puede ocurrir representar es una en la que queden reflejadas las últimas lecturas de cada uno de los sensores. Se debe tener en cuenta que estas lecturas no se han tomado en un ambiente urbano, como es el emplazamiento final en que se espera se encuentren, sino controlado (habitación, laboratorio...) por lo que las medidas no difieren unas de otras en demasiado.



Figura 4.13 Últimas 15 lecturas registradas para diferentes sensores.

El otro tipo de gráfica que se ha desarrollado es aquella que provee los valores de las lecturas máximas, medias y mínimas por día.

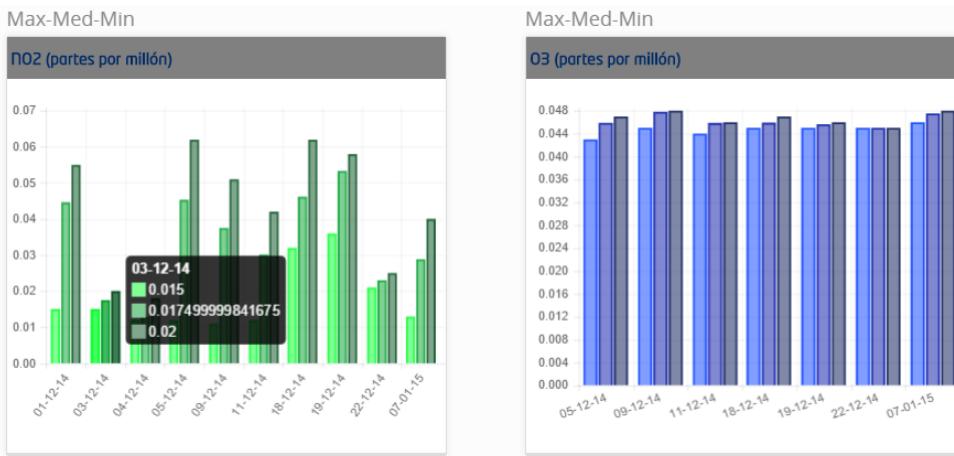


Figura 4.14 Gráfico de valores mínimos, medios y máximos.

Tener al alcance de la mano esta información puede ayudar a tomar decisiones en pos de, por ejemplo, re conducir el tráfico en una cierta zona por riesgo a sobre pasar valores máximos para un cierto contaminante. El análisis de este *big data* y las actuaciones posteriores son las que realmente definirán la valía de este tipo de soluciones.

4.2.3 Inspector de sensores

Resulta interesante poder obtener las últimas lecturas de los diferentes sensores a modo de comparativa entre zonas, instantes de tiempo, o simplemente para echar un rápido vistazo a la información que en un momento dado pueda requerirse. El widget que se presenta recaba la información de las lecturas a través de una interacción directa con Orion Context Broker y genera un estadillo en el que va rellenándose la información. Añadiendo algo de estilo podemos crear una aplicación sencilla pero muy útil a nivel de administración. No solo tenemos por qué obtener información de los sensores de polución sino que, de poseer nuestro proyecto otras verticales o tipo de sensores, también podrían aparecer reflejadas aquí.

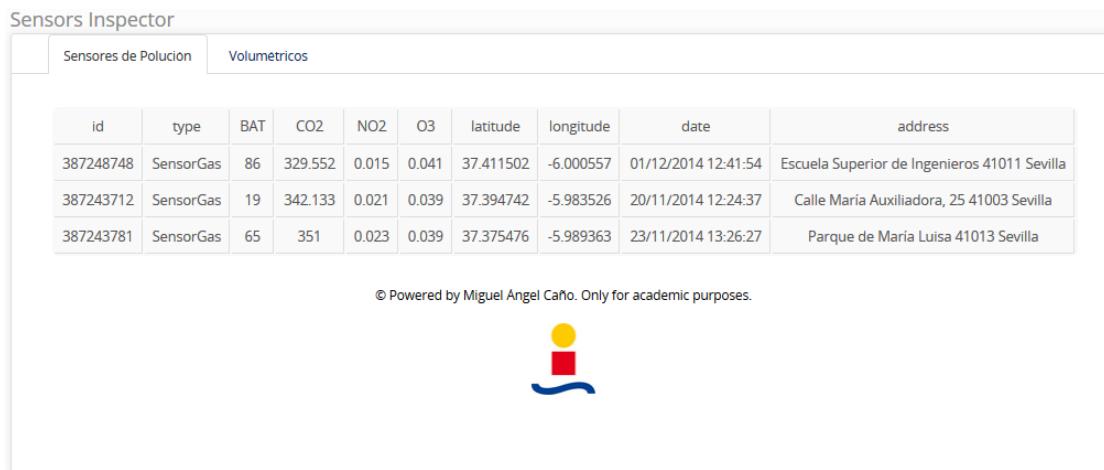


Figura 4.15 Inspector de sensores: diferentes verticales.

4.2.4 Alertas y notificaciones

Generar alarmas para alertar acerca de un determinado evento en la ciudad podría resultar de gran utilidad en proyectos de esta índole. Un ejemplo de esto podría ser generar alertas para avisar de un incendio en un contenedor donde se haya desplegado un sensor de incendio. Esto podría promover una mayor celeridad en la actuación de los servicios de bomberos. Este no es nuestro caso, pero desarrollando una aplicación genérica podríamos dar soporte a diferentes verticales. En el caso que nos ocupa nos puede ser útil para generar alertas

por zona, cuando se supere un nivel preestablecido para algún parámetro. A través de los ajustes del widget, proveeremos la regla o condición para generación de alertas, y el teléfono móvil o email al que irán a parar las notificaciones.

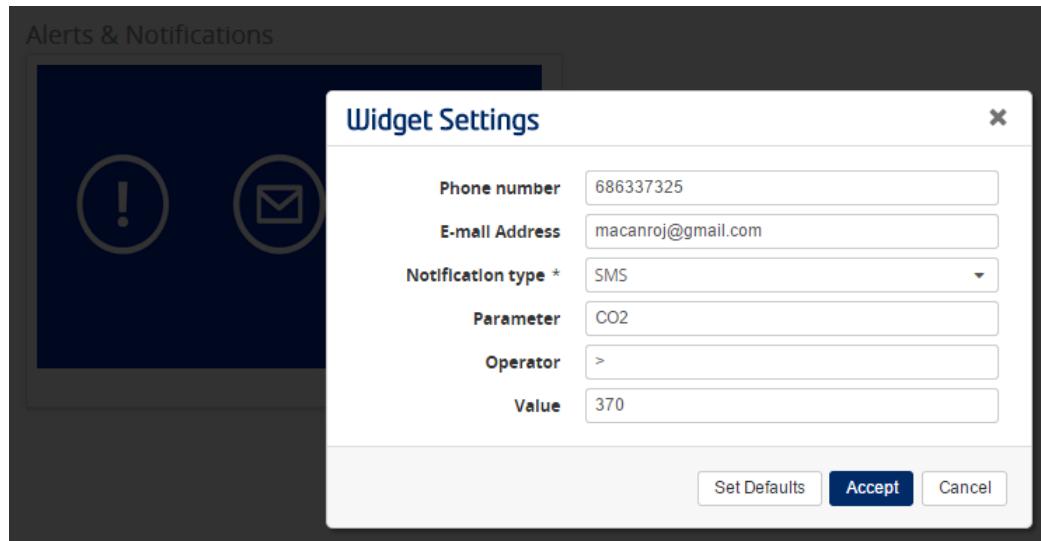


Figura 4.16 Alertas y notificaciones: ajustes.

Reiniciando la aplicación para que los cambios se apliquen, podemos ver cómo la alerta se genera vía SMS en el teléfono indicado, o email en la dirección ajustada.

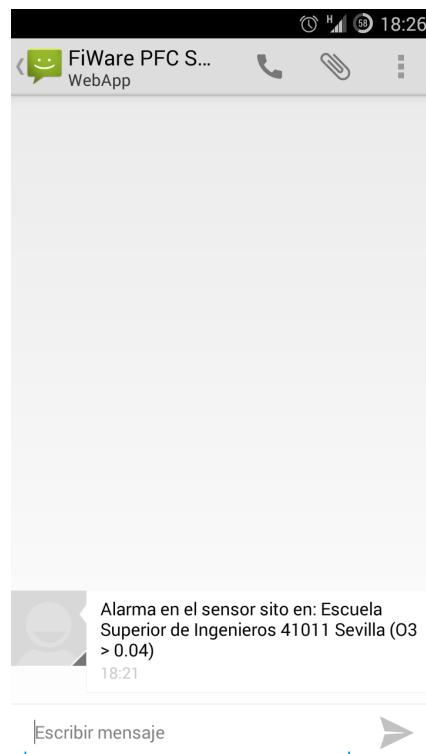


Figura 4.17 Alertas SMS: recepción de la alerta.

Esta aplicación se ha implementado meramente en Javascript/AJAX y PHP, utilizando la API PHP de Esendex, una empresa dedicada a ofrecer servicios de SMS para el sector profesional. Puede probarse

gratuitamente con una cierta limitación de mensajes. Para el envío de emails se utiliza una librería PHP dedicada al efecto.

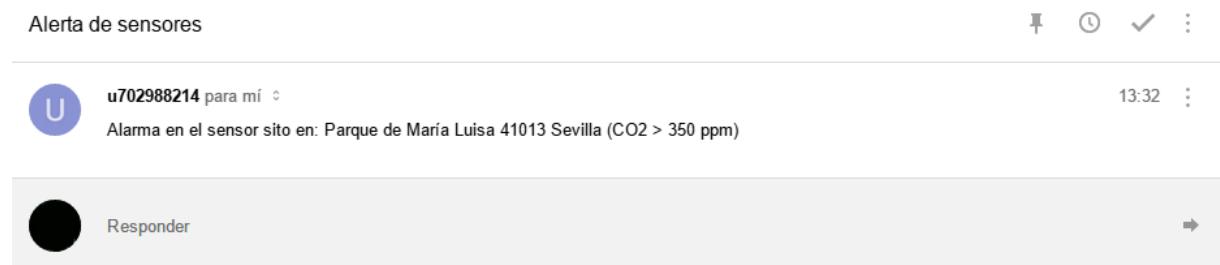


Figura 4.18 Alertas email: recepción de la alerta.

Conclusiones y líneas futuras de trabajo

La pretensión inicial de este desarrollo era asentar una base para la sostenibilidad medioambiental en la ciudad, en forma de una solución integral basada en el despliegue de redes de sensores y el desarrollo del software adecuado para monitorización a través de FIWARE. Si bien se expone sobre la ciudad de Sevilla, el proyecto resulta perfectamente replicable a otras. Bajo una visión algo más comercial y desmarcada de la línea académica, este proyecto podría perfectamente tomarse como una solución llave en mano para una ciudad que pretenda monitorizar la presencia de ciertos gases en su entorno.

Sin duda alguna la fase en que se realizó una mayor inversión en tiempo fue durante el desarrollo en Fi-Ware. Si bien la documentación era extensa, la curva de aprendizaje se tornó más prolongada de lo que inicialmente parecía. Resultó de gran ayuda el soporte por parte de Telefónica I+D, aunque las respuestas podían demorarse días e, incluso, semanas. La plataforma se encontraba en una temprana fase *beta* y a menudo las máquinas desplegadas y los GEs quedaban en un estado inconsistente, dejando inutilizable el sistema. Por otro lado, algunos componentes interesantes como IDAS (Intelligence Data Advanced Solution), entre otros, no pudieron utilizarse por encontrarse aún en desarrollo. Este elemento es un conector que actúa en conjunción con Orion Context Broker añadiendo capacidad de actuación ante eventos definidos para los elementos del contexto. La aplicación desarrollada para envío de SMS/Email, si bien resulta funcional aprovechando las suscripciones actuales de Orion, si deseamos añadir usabilidad y control por parte del usuario final, no nos queda otra opción que relegarla a la posición de línea futura de trabajo.

Llegados a este punto, podemos definir algunas de las nuevas posibilidades que se abren, bien como evolución natural de este proyecto, o como extensiones del mismo. Actualmente han aparecido nuevos protocolos de comunicación optimizados para despliegues en entornos urbanos, como es el caso de LoRa, un protocolo basado en ZigBee que consigue un mayor rango de distancia entre nodos. Un estudio analítico a nivel radio de los diferentes protocolos, apoyado de pruebas en campo y conclusiones correctamente extraídas podría ser útil para tomar decisiones en cuanto al protocolo de comunicación a emplear en este tipo de proyectos.

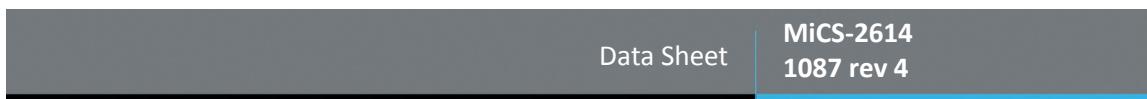
Si bien la solución ofrecida a nivel de equipos es la óptima hoy en día para este cometido, se propone diseñar y construir electrónica dedicada a proveer lecturas de concentraciones de diferentes gases, tratando de minimizar costes y aportando una solución modular y adecuada a cada tipo.

Como evolución natural de este proyecto, se podría proponer analizar el *big data* generado por los sensores tratando de establecer líneas de tendencia o niveles esperados de polución para cada zona de la ciudad en que exista despliegue de sensores. Este análisis podría ir acompañado del desarrollo de un *heatmap* (mapa de calor) sobre el que quedasen reflejadas las zonas de mayor concentración para cada tipo de gas. Tiempo después de que este proyecto se desarrollase aparecieron nuevos GEs de Fi-Ware, como los ya mencionados, por lo que sus capacidades aumentaron y, consecuentemente, el número de proyectos que puedan hacer uso de ella.

Apéndice A

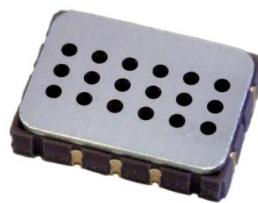
Datasheets

A.1 Algunos sensores de interés: ozono (O₃).



The MiCS-2614 is a compact MOS sensor.

The MiCS-2614 is a robust MEMS sensor for ozone detection; suitable also for gas leak detection and outdoor air quality monitoring.

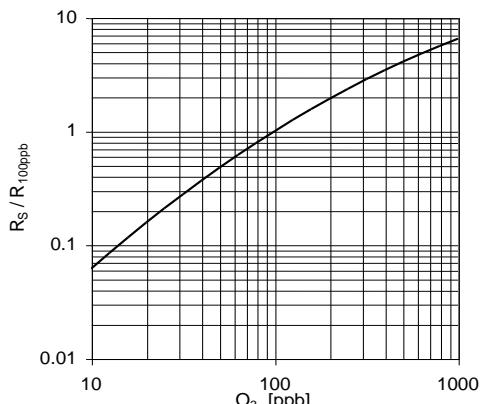


Features

- Smallest footprint for compact designs (5 x 7 x 1.55 mm)
- Robust MEMS sensor for harsh environments
- High-volume manufacturing for low-cost applications
- Short lead-times

Detectable gases

- Ozone O₃ 10 – 1000 ppb



Continuous power ON, 25°C, 50% RH

For more information please contact:

info.em@sgxsensortech.com

SGX Sensortech, Courtils 1
CH-2035 Corcelles-Cormondrèche
Switzerland

www.sgxtech.com

Performance sensor

Characteristic RED sensor	Symbol	Typ	Min	Max	Unit
Sensing resistance in air (see note 1)	R_0	11	3	60	kΩ
Typical NH ₃ detection range	FS		10	1000	ppb
Sensitivity factor (see note 2)	S_R	2	1.5	4	-

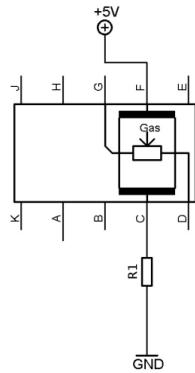
Notes:

1. Sensing resistance in air R_0 is measured under controlled ambient conditions, i.e. synthetic air at 23 ± 5°C and 50 ± 10% RH. Sampling test.
2. Sensitivity factor is defined as R_s at 100 ppb of O₃ divided by R_s at 50 ppb of O₃. Test conditions are 25 ± 2°C and 50 ± 5% RH. Indicative values only. Sampling test.

IMPORTANT PRECAUTIONS:

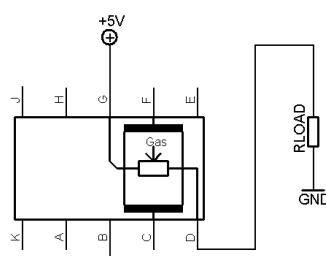
Read the following instructions carefully before using the MiCS-2614 described here to avoid erroneous readings and to prevent the device from permanent damage.

- The sensor must be reflow soldered in a neutral atmosphere, without soldering flux vapours.
- The sensor must not be exposed to high concentrations of organic solvents, silicone vapours or cigarette-smoke in order to avoid poisoning the sensitive layer.
- Heater voltage above the specified maximum rating will destroy the sensor due to overheating.
- This sensor is to be placed in a filtered package that protects it against water and dust projections.
- SGX sensortech strongly recommends using ESD protection equipment to handle the sensor.



MiCS-2614 with recommended supply circuit (top view)

R1 is 82 Ω . This resistor is necessary to obtain the right temperature on the heater while using a single 5V power supply. The resulting voltage is typically $V_H = 2.35V$.



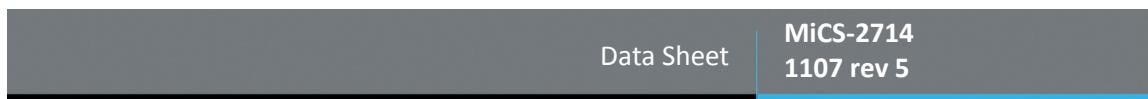
MiCS-2614 with measurement circuit (top view)

The voltage measured on the load resistor is directly linked to the resistance of the sensor respectively. RLOAD must be 820 Ω at the lowest in order not to damage the sensitive layer.

Parameter	Symbol	Typ	Min	Max	Unit
Heating power	P_H	80	66	95	mW
Heating voltage	V_H	2.35	-	-	V
Heating current	I_H	34	-	-	mA
Heating resistance at nominal power	R_H	68	58	78	Ω

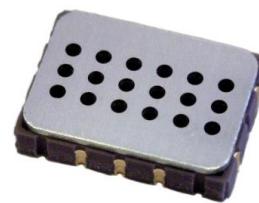
Rating	Symbol	Value / Range	Unit
Maximum heater power dissipation	P_H	95	mW
Maximum sensitive layer power dissipation	P_s	1	mW
Voltage supplyHeating current	V_{supply}	4.9 – 5.1	V
Relative humidity range	RH	5 – 95	%RH
Ambient operating temperature	Tamb	-40 – 70	°C
Storage temperature range	Tsto	-40 – 50	°C
Storage humidity range	RHsto	5 - 95	%RH

A.2 Algunos sensores de interés: dióxido de nitrógeno (NO₂).



The MiCS-2714 is a compact MOS sensor.

The MiCS-2714 is a robust MEMS sensor for nitrogen dioxide and leakage detection.

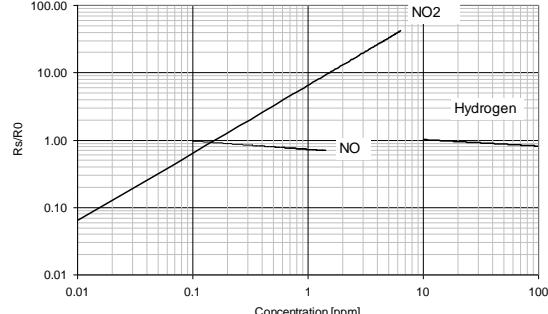


Features

- Smallest footprint for compact designs (5 x 7 x 1.55 mm)
- Robust MEMS sensor for harsh environments
- High-volume manufacturing for low-cost applications
- Short lead-times

Detectable gases

- | | | |
|--------------------|-----------------|--------------|
| • Nitrogen dioxide | NO ₂ | 0.05 – 10ppm |
| • Hydrogen | H ₂ | 1 – 1000ppm |



Continuous power ON, 25°C, 50% RH

For more information please contact:

info.em@sgxsensortech.com

SGX Sensortech, Courtis 1
CH-2035 Corcelles-Cormondrèche
Switzerland

www.sgxtech.com

Performance sensor

Characteristic OX sensor	Symbol	Typ	Min	Max	Unit
Sensing resistance in air (see note 1)	R_0	-	0.8	20	kΩ
Typical CO detection range	FS		0.05	10	ppm
Sensitivity factor (see note 2)	S_{60}	-	2	-	-

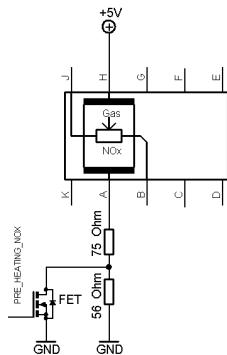
Notes:

1. Sensing resistance in air R_0 is measured under controlled ambient conditions, i.e. synthetic air at 23 ± 5°C and 50 ± 10% RH. Sampling test.
2. Sensitivity factor is defined as R_s at 0.25 ppm NO₂, divided by R_s in air. Test conditions are 23 ± 5°C and ≤5% RH. Indicative values only. Sampling test.

IMPORTANT PRECAUTIONS:

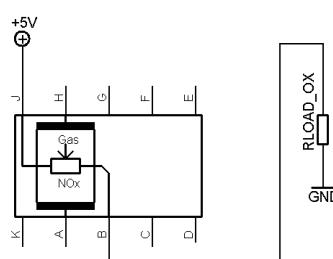
Read the following instructions carefully before using the MiCS-2714 described here to avoid erroneous readings and to prevent the device from permanent damage.

- The sensor must be reflow soldered in a neutral atmosphere, without soldering flux vapours.
- The sensor must not be exposed to high concentrations of organic solvents, silicone vapours or cigarette-smoke in order to avoid poisoning the sensitive layer.
- Heater voltage above the specified maximum rating will destroy the sensor due to overheating.
- This sensor is to be placed in a filtered package that protects it against water and dust projections.
- SGX sensortech strongly recommends using ESD protection equipment to handle the sensor.



MiCS-2714 with recommended supply circuit (top view)

R is 131 Ω . This resistor is necessary to obtain the right temperature on the heater while using a single 5 V power supply. The resulting voltages is typically V_H = 1.7 V.



MiCS-2714 with measurement circuit (top view)

The voltage measured on the load resistor is directly linked to the resistance of the sensor. R_{LOAD} must be 820 W at the lowest in order not to damage the sensitive layer.

Parameter	Symbol	Typ	Min	Max	Unit
Heating power	P _H	43	30	50	mW
Heating voltage	V _H	1.7	-	-	V
Heating current	I _H	26	-	-	mA
Heating resistance at nominal power	R _H	66	59	73	Ω

Rating	Symbol	Value / Range	Unit
Maximum heater power dissipation	P _H	50	mW
Maximum sensitive layer power dissipation	P _s	8	mW
Voltage supplyHeating current	V _{supply}	4.9 – 5.1	V
Relative humidity range	RH	5 – 95	%RH
Ambient operating temperature	T _{amb}	-30 – 85	°C
Storage temperature range	T _{sto}	-40 – 120	°C
Storage humidity range	RH _{sto}	5 - 95	%RH

A.3 WaspMote



WaspMote - Datasheet

WaspMote

General data:

Microcontroller:	ATmega1281
Frequency:	14.7456 MHz
SRAM:	8KB
EEPROM:	4KB
FLASH:	128KB
SD Card:	2GB
Weight:	20gr
Dimensions:	73.5 x 51 x 13 mm
Temperature Range:	[-10°C, +65°C]
Clock:	RTC (32KHz)



Consumption:

ON:	15mA
Sleep:	55µA
Deep Sleep:	55µA
Hibernate:	0.07µA

Operation without recharging: 1 year *

* Time obtained using the Hibernate mode as the energy saving mode

Inputs/Outputs:

7 Analog (I), 8 Digital (I/O), 1 PWM, 2 UART, 1 I2C, 1USB, 1SPI

Electrical data:

Battery voltage:	3.3 V - 4.2V
USB charging:	5 V - 100mA
Solar panel charging:	6 - 12 V - 280mA

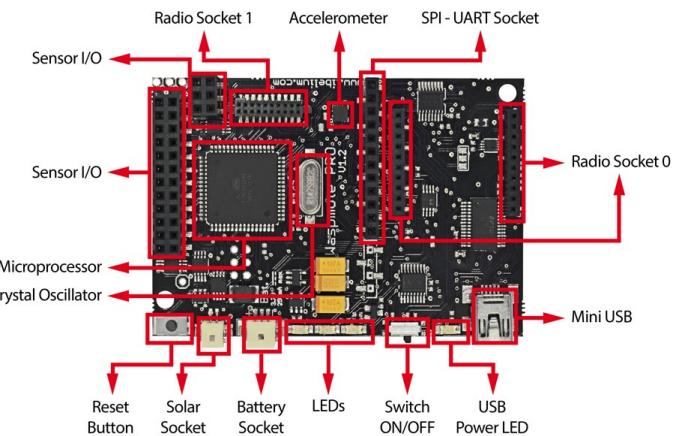


Figure: WaspMote Board Top

Built-in sensors on the board:

Temperature (+/-): -40°C , +85°C. Accuracy: 0.25°C
Accelerometer: ±2g/±4g/±8g
 Low power: 0.5 Hz/1 Hz/2 Hz/5 Hz/10 Hz
 Normal mode: 50 Hz/100 Hz/400 Hz/1000 Hz

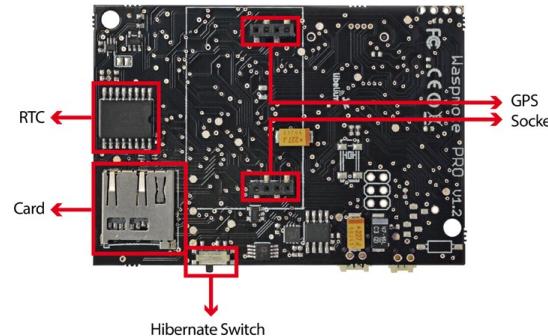


Figure: WaspMote Board Bottom

802.15.4/ZigBee

Model	Protocol	Frequency	txPower	Sensitivity	Range *
XBee-802.15.4-Pro	802.15.4	2.4GHz	100mW	-100dBm	7000m
XBee-ZB-Pro	ZigBee-Pro	2.4GHz	50mW	-102dBm	7000m
XBee-868	RF	868MHz	315mW	-112dBm	12km
XBee-900	RF	900MHz	50mW	-100dBm	10km

* Line of sight and Fresnel zone clearance with 5dBi dipole antenna



Figure: XBee

Antennas: 2.4GHz: 2dBi / 5dBi
 868/900MHz: 0dBi / 4.5dBi
Connector: RP-SMA
Encryption: AES 128b
Control Signal: RSSI
Standards: XBee-802.15.4 - 802.15.4 Compliant / XBee-ZB - ZigBee-Pro v2007 Compliant
Topologies: star, tree, mesh

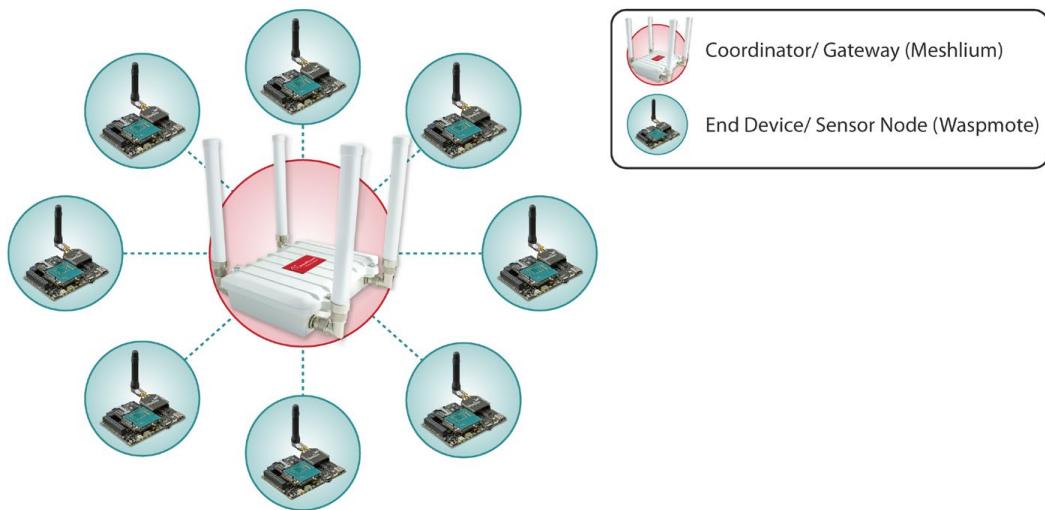


Figure: Star

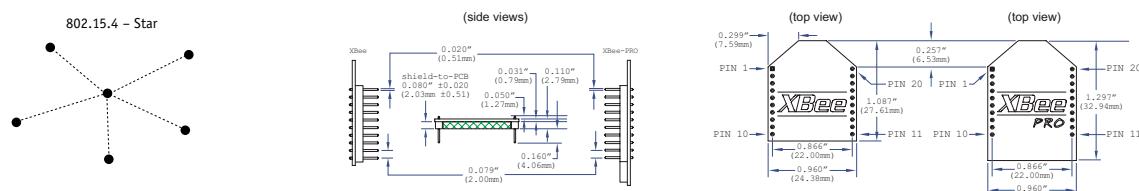
Certifications

- CE (Europe)
- FCC (USA)
- IC (Canada)



A.4 XBee modules

Platform	XBee® 802.15.4 (Series 1)	XBee-PRO® 802.15.4 (Series 1)	XBee-PRO® XSC
Performance			
RF Data Rate	250 kbps	250 kbps	10 kbps / 9.6 kbps
Indoor/Urban Range	100 ft (30 m)	300 ft (100 m)	Up to 1200 ft (370 m)
Outdoor/RF Line-of-Sight Range	300 ft (100 m)	1 mi (1.6 km)	Up to 6 mi (9.6 km)
Transmit Power	1 mW (+0 dBm)	60 mW (+18 dBm)*	100 mW (+20 dBm)
Receiver Sensitivity (1% PER)	-92 dBm	-100 dBm	-106 dBm
Features			
Serial Data Interface	3.3V CMOS UART	3.3V CMOS UART	3.3V CMOS UART (5V Tolerant)
Configuration Method	API or AT Commands, local or over-the-air	API or AT Commands, local or over-the-air	AT Commands
Frequency Band	2.4 GHz	2.4 GHz	902 MHz to 928 MHz
Interference Immunity	DSSS (Direct Sequence Spread Spectrum)	DSSS (Direct Sequence Spread Spectrum)	FHSS (Frequency Hopping Spread Spectrum)
Serial Data Rate	1200 bps - 250 kbps	1200 bps - 250 kbps	1200 bps - 57.6 kbps
ADC Inputs	(6) 10-bit ADC inputs	(6) 10-bit ADC inputs	None
Digital I/O	8	8	None
Antenna Options	Chip, Wire Whip, U.FL, & RPSSMA	Chip, Wire Whip, U.FL, & RPSSMA	Wire Whip, U.FL, RPSSMA
Networking & Security			
Encryption	128-bit AES	128-bit AES	No
Reliable Packet Delivery	Retries/Acknowledgments	Retries/Acknowledgments	Retries/Acknowledgements
IDs and Channels	PAN ID, 64-bit IEEE MAC, 16 Channels	PAN ID, 64-bit IEEE MAC, 12 Channels	PAN ID, 32-bit Address, 7 Channels
Power Requirements			
Supply Voltage	2.8 - 3.4VDC	2.8 - 3.4VDC	3.0 - 3.6VDC
Transmit Current	45 mA @ 3.3VDC	215 mA @ 3.3VDC	265 mA typical
Receive Current	50 mA @ 3.3VDC	55 mA @ 3.3VDC	65 mA typical
Power-Down Current	<10 uA @ 25° C	<10 uA @ 25° C	45 uA pin Sleep
Regulatory Approvals			
FCC (USA)	OUR-XBEE	OUR-XBEEPRO	MCQ-XBEEXSC
IC (Canada)	4214A-XBEE	4214A-XBEEPRO	1846A-XBEEXSC
ETSI (Europe)	Yes	Yes* Max TX 10 mW	No
C-TICK Australia	Yes	Yes	No
Telec (Japan)	Yes	Yes*	No



Visit www.digi.com for part numbers.

DIGI SERVICE AND SUPPORT - You can purchase with confidence knowing that Digi is here to support you with expert technical support and a strong five-year warranty. www.digi.com/support



91001412
B1/911

Digi International Worldwide HQ
877-912-3444
952-912-3444

France
+33-1-55-61-98-98

Japan
www.digi.fr

Japan
+81-3-5428-0261
www.digi.co.jp

India
+91-80-4287-9887
www.digi-intl.co.in

Singapore
+65-6213-5380
www.digi.sg

China
+86-21-5150-6898
www.digi.cn

BUY ONLINE • www.digi.com

info@digi.com



© 2006-2011 Digi International Inc.
All rights reserved. Digi, Digi International, the Digi logo, the When Reliability Matters logo, XBee and XBee-PRO are trademarks or registered trademarks of Digi International Inc. in the United States and other countries worldwide. All other trademarks are the property of their respective owners.

A.5 Modulo GSM/GPRS Waspmove: SIM900

SIM900
The GSM/GPRS Module for M2M applications

SIM900 GSM/GPRS Module



The SIM900 is a complete Quad-band GSM/GPRS solution in a SMT module which can be embedded in the customer applications.

Featuring an industry-standard interface, the SIM900 delivers GSM/GPRS 850/900/1800/1900MHz performance for voice, SMS, Data, and Fax in a small form factor and with low power consumption. With a tiny configuration of 24mm x 24mm x 3 mm, SIM900 can fit almost all the space requirements in your M2M application, especially for slim and compact demand of design.

- SIM900 is designed with a very powerful single-chip processor integrating AMR926EJ-S core
- Quad - band GSM/GPRS module with a size of 24mmx24mmx3mm
- SMT type suit for customer application
- An embedded Powerful TCP/IP protocol stack
- Based upon mature and field-proven platform, backed up by our support service, from definition to design and production

General features

- Quad-Band 850/ 900/ 1800/ 1900 MHz
- GPRS multi-slot class 10/8
- GPRS mobile station class B
- Compliant to GSM phase 2/2+
 - Class 4 (2 W @850/ 900 MHz)
 - Class 1 (1 W @ 1800/1900MHz)
- Dimensions: 24* 24 * 3 mm
- Weight: 3.4g
- Control via AT commands (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands)
- SIM application toolkit
- Supply voltage range 3.4 ... 4.5 V
- Low power consumption
- Operation temperature:
-30 °C to +80 °C

Specifications for fax

- Group 3, class 1

Specifications for data

- GPRS class 10: max. 85.6 kbps (downlink)
- PBCCH support
- Coding schemes CS 1, 2, 3, 4
- CSD up to 14.4 kbps
- USSD
- Non transparent mode
- PPP-stack

Specifications for SMS via GSM Pin Assignment

/ GPRS

- Point-to-point MO and MT
- SMS cell broadcast
- Text and PDU mode

Drivers

- MUX Driver

Specifications for voice

- Tricodec
 - Half rate (HR)
 - Full rate (FR)
 - Enhanced Full rate (EFR)

More about SIM900 module, Please contact: Tel:+86 21 32523300

Fax:+86 21 32523301

Email:simcom@sim.com

- Hands-free operation
(Echo suppression)
- AMR
Half Rate(HR)
Full Rate(FR)

Interfaces

- Interface to external SIM 3V/ 1.8V
- analog audio interface
- RTC backup
- SPI interface
- Serial interface
- Antenna pad
- I2C
- GPIO
- PWM
- ADC

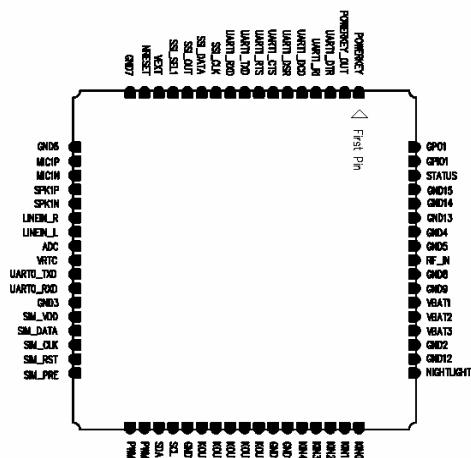
Compatibility

- AT cellular command interface

Approvals (in planning)

- CE
- FCC
- ROHS
- PTCRB
- GCF
- AT&T
- IC
- TA

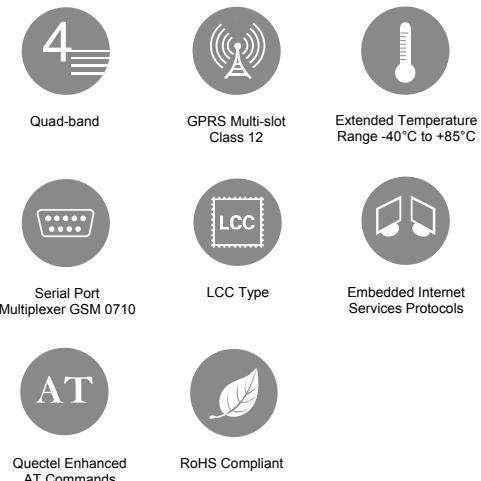
Pin Assignment



A.6 Arduino GSM Shield: Quectel M10

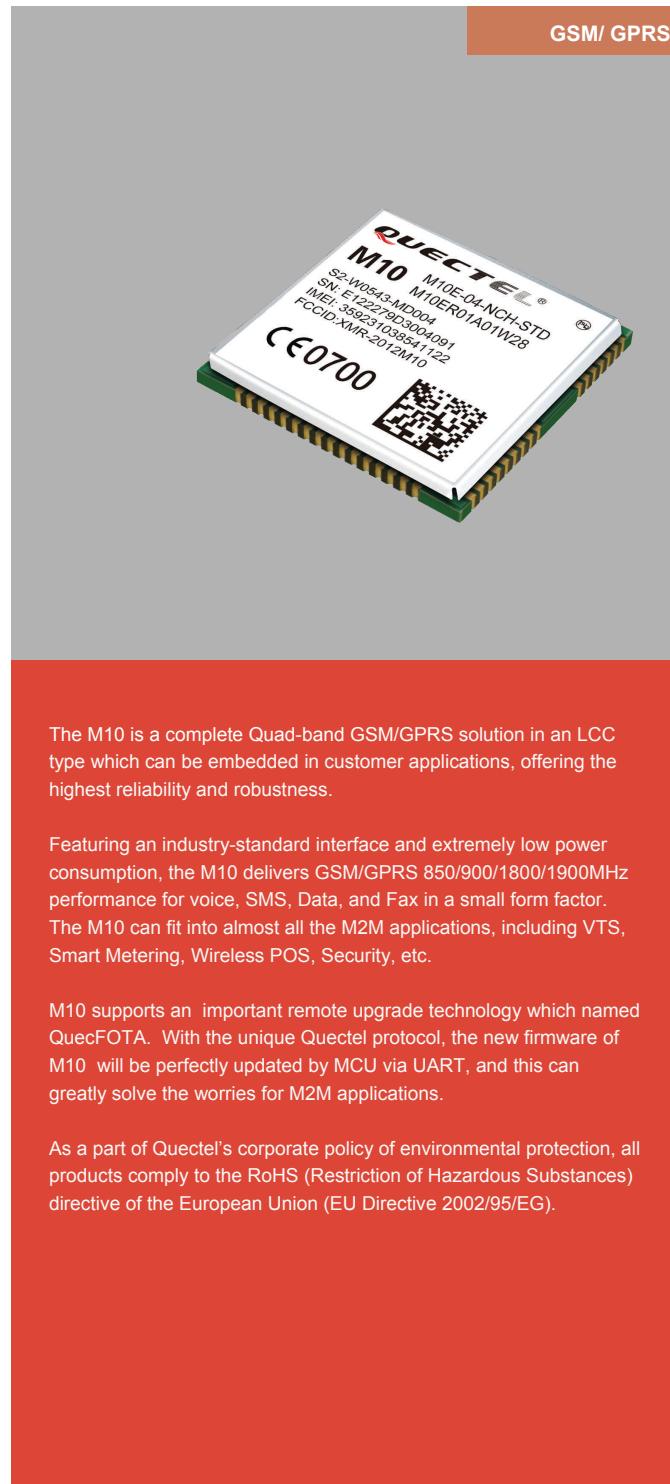
Quectel M10

**Quad-band
GSM/ GPRS Module**



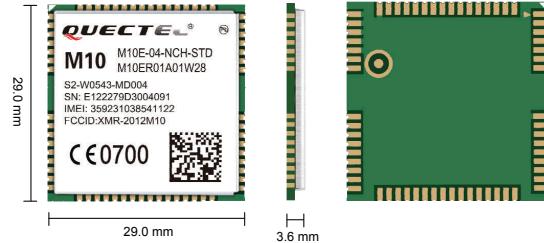
Key benefits

- ❑ Quad-band GSM/GPRS module with a size of 29.0 x 29.0 x 3.6mm
- ❑ LCC type suits for customer application
- ❑ Embedded powerful internet service protocols, multiple Sockets & IP addresses
- ❑ Based on mature and field-proven platform, backed up by our support service, from definition to design and production
- ❑ Certificates: CE/ GCF/ FCC/ PTCRB/ IC/ANATEL/ NORM/ COFETEL/ ICASA/ NCC/ China TA



Quectel M10

Quad-band GSM/GPRS Module



General Features

Quad-band	850/ 900/ 1800/ 1900 MHz
GPRS Multi-slot Class	12, 1~12 configurable
GPRS Mobile Station	Class B
Compliant to GSM Phase 2/2+	Class 4 (2W @ 850/ 900 MHz) Class 1 (1W @ 1800/1900MHz)
Supply Voltage Range	3.3~4.6V 4.0V nominal
Low Power Consumption	1.3mA @ DRX=5 1.2mA @ DRX=9
Operation Temperature	-40 °C to +85 °C
Dimensions	29.0 x 29.0 x 3.6mm
Weight	Approx. 6.0g
Mux Driver	GSM 07.10
Control via AT commands	GSM 07.07 ,07.05 and other enhanced AT Commands
SIM Application Toolkit	

Specifications for Fax & Data

Fax	Group 3, Class 1& 2
GPRS Class 12	Max. 85.6 kbps (uplink & downlink)
PBCCH Support	
Coding Schemes	CS 1, 2, 3, 4
CSD	Up to 14.4 kbps
USSD	
Non Transparent Mode	
Protocols	PPP/ TCP/ UDP/ HTTP/ FTP/ MMS/ SMTP/ MUX

Specifications for SMS via GSM /GPRS

Point-to-point MO and MT	
SMS Cell Broadcast	
Text and PDU Mode	

Specifications for Voice

Speech Codec Modes	Half Rate (HR) Full Rate (FR) Enhanced Full Rate (EFR) AMR Half Rate AMR Full Rate
Echo Arithmetic	Echo Cancellation Echo Suppression Noise Reduction
Hands-free Operation	

Interfaces

External SIM	3V/ 1.8V
Analog Audio Interfaces	Two channels
RTC Backup	
SD Card Interface	
Serial Port and Debug Port	
Antenna Pad	
GPIOs	

Certifications

CE	FCC	GCF	ICASA	PTCRB	UCRF
IC	NCC	NAL	Rogers	ANATEL	China TA

Apéndice B

Código fuente de interés

Todo el código desarrollado para llevar a cabo este proyecto se puede encontrar en el repositorio público *Github* dedicado al efecto y accesible mediante la siguiente dirección: <https://github.com/macano953/PFC>.

Índice de Figuras

1.1.	El programa FI-PPP	4
1.2.	Uso de GEs en instancias Fi-Ware	5
1.3.	Visión general de la arquitectura de Fi-Ware	6
1.4.	Recursos de NGSI	7
1.5.	Ejemplo de uso: velocímetro a tiempo real en app móvil	7
1.6.	Visión general de la arquitectura del <i>Data/Context Management</i>	8
1.7.	Estructura de un <i>data element</i>	9
1.8.	Estructura de un <i>context element</i>	9
1.9.	Integración NGSI para <i>Data/Context Management</i>	10
1.10.	Interacciones típicas a través del Context Broker	10
2.1.	Elementos principales en una red de sensores	15
2.2.	Topologías posibles en una red ZigBee	16
2.3.	Comparativa de la pila de capas en Bluetooth y BLE	16
2.4.	Telefónica Thinking Things	17
2.5.	Smart Citizen Board	18
2.6.	Libelium WaspMote	18
2.7.	Gas Sensor Board para WaspMote	18
2.8.	Placa Arduino Uno con <i>shield</i> para módulo ZigBee	19
2.9.	Libelium Smart Environment desplegado en una ciudad	20
2.10.	WaspMote v1.2: frontal	21
2.11.	WaspMote v1.2: trasera	21
2.12.	Shield Gas Sensor Board: detalle	22
2.13.	Detalle del modelo Smart Environment	23
2.14.	Compatibilidad socket-sensores.	24
2.15.	Modelo de red	25
3.1.	Arduino IDE / WaspMote IDE	27
3.2.	Carga de programas en WaspMote	28
3.3.	<i>Gas board</i> presente en el interior del dispositivo	29
3.4.	Ejemplo de curva de calibración: sensor de humedad Sencera 808HSV5	30
3.5.	Ejemplo de curva de calibración: ΔEMF	31
3.6.	Ejemplo de curva de calibración: R_s/R_o	32
3.7.	Orion Context Broker: interacciones.	35
3.8.	Interfaz web phpMyAdmin	38
4.1.	Cloud: el apartado para <i>cloud hosting</i> en Fi-Ware	41
4.2.	Instalando Orion Context Broker	42
4.3.	Abriendo puertos y creando un grupo de seguridad	42
4.4.	Orion Context Broker desplegado y accesible	43
4.5.	Wirecloud: capa más alta de la arquitectura Fi-Ware	43
4.6.	Interconexión de widgets en el <i>wiring</i>	44

4.7.	Estructura básica (gráfica) de un widget	45
4.8.	Marketplace en Fi-Lab	45
4.9.	Marketplace en Fi-Lab	46
4.10.	Visualización de puntos de interés en Google Maps	47
4.11.	Visualización inicial de sensores en Google Maps	47
4.12.	Visualización final de sensores en Google Maps	48
4.13.	Últimas 15 lecturas registradas para diferentes sensores	48
4.14.	Gráfico de valores mínimos, medios y máximos	49
4.15.	Inspector de sensores: diferentes verticales	49
4.16.	Alertas y notificaciones: ajustes	50
4.17.	Alertas SMS: recepción de la alerta	50
4.18.	Alertas email: recepción de la alerta	51

Índice de Tablas

2.1.	Grado de protección IP: primer dígito	12
2.2.	Grado de protección IP: segundo dígito	13
2.3.	Comparativa de protocolos para comunicaciones inalámbricas	14
2.4.	Arduino UNO: especificaciones técnicas	19
2.5.	Wasp mote v1.2: especificaciones técnicas	22
2.6.	Presupuesto del gateway: modelo basado en Wasp mote	25
2.7.	Presupuesto del gateway: modelo basado en Arduino	26
3.1.	Parámetros a los que suscribir el Context Broker.	37
4.1.	Estructura básica de un widget (.wgt)	45

Bibliografía

- [1] Dr. Axel Rauschmayer, *Speaking Javascript*, 3 ed., O'Reilly, 2014.
- [2] Luke Wellin, Laura Thomson, *PHP and MySQL Web Development*, 4. ed, Addison-Wesley Professional, 2008.
- [3] ACGIH, *Threshold Limit Values and Biological Exposure Indices*, 7th Ed, 2001.
- [4] https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Main_Page
- [5] <http://ec.europa.eu/digital-agenda/en/future-internet-public-private-partnership>
- [6] <http://edu.fi-ware.org/>
- [7] <http://technical.openmobilealliance.org/>
- [8] <http://www.nema.org/Standards/ComplimentaryDocuments/ANSI-IEC-60529.pdf>
- [9] <http://www.thinkingthings.telefonica.com>
- [10] <http://www.smartcitizen.me>
- [11] <http://www.thinkingthings.telefonica.com>
- [12] <http://www.zigbee.org/>
- [13] <http://research.nokia.com/publication/12259>
- [14] <http://www.cooking-hacks.com/>
- [15] <http://arduino.cc/>
- [16] <http://www.who.int/mediacentre/news/releases/2014/air-quality/en/>
- [17] <http://www.php.net/>
- [18] <http://conwet.fi.upm.es/wirecloud/>
- [19] <http://developers.esendex.com/>