Gobierno del Principado de Asturias
Consejería de Economía y Empleo

Unión Europea
Fondo Europeo
de Desarrollo Regional

RESULTADOS SmartAirQuality

Monitorización de la calidad de aire en recintos cerrados a través del uso de IoT y WoT

En Gijón , a 31 de Diciembre de 2017

1. MEMORIA TÉCNICA

Introducción

El proyecto SmartAirQuality (Sistema basado en la medición de la calidad del aire en ambientes interiores no industriales, en este caso se va a medir en hospitales) es ejecutado por Fundación CTIC – Centro entre el 1 de enero de 2016 y el 31 de diciembre de 2017.

El objetivo del proyecto SmartAirQuality es el desarrollo de un sistema que permita medir la calidad del aire en espacios interiores, en este caso en hospitales. Se desarrolla mediante un sistema de Internet y Web de las cosas (Iot/WoT). Se realizará la monitorización de estos datos al sistema en tiempo real. Mediante un estudio inteligente, se logrará visualizar en tiempo real estos datos, dando informes o alarmas en el caso de procesos críticos.

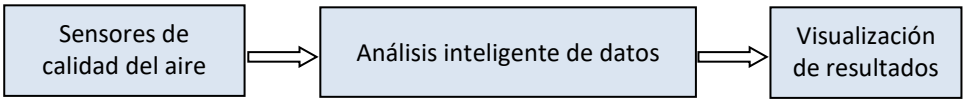

Figura 1: Bloques principales del Proyecto SmartQuality

La ejecución del proyecto SmartAirQuality se dividió en cinco hitos claramente diferenciados:

- Hito 1: Estudio del caso de uso de hospitales.
- Hito 2: Diseño y Desarrollo prototipo IoT.
- Hito 3: Estudio e implementación estandarización WoT.
- Hito 4: Estudio e implementación de comunicaciones WoT.
- Hito 5: Dashboard visualización.

A continuación, se describen las actividades realizadas en cada uno de estos hitos de trabajo y los resultados conseguidos.

Anualidad 2016

Hito 1: Estudio del caso de uso de hospitales.

T1. Estudio de las normativas de calidad del aire para hospitales.

Se ha realizado el estudio de la legislación de la calidad del aire en hospitales tanto nacional como a nivel europeo

La calidad de aire en el interior de hospitales debe cumplir las exigencias recogidas en el Reglamente de

Instalaciones Térmicas en los edificios (RITE) aprobado en Real Decreto 1027/2007, cuyo objeto es establecer las exigencias de eficiencia energética y seguridad que deben cumplir las instalaciones térmicas en los edificios destinadas a atender la demanda de bienestar e higiene de las personas, durante su diseño y dimensionado, ejecución, mantenimiento y uso, así como determinar los procedimientos que permitan acreditar su cumplimiento.

Este documento estable las siguientes normas nacionales de revisión de calidad del aire interior, con las normas:

- UNE 171330 - Calidad ambiental en interiores.
- UNE 171340 - Validación y cualificación de salas de ambiente controlado en hospitales.
- UNE 100713- Instalación de acondicionamiento de aire en hospitales.
- UNE 100012:2005 - Higienización de sistemas de climatización.

Local	UNE 100713:2005				
	Temperatura		Humedad relativa		
	Máxima	Mínima	Máxima	Mínima	Presión sonora máxima
En todo el centro sanitario	26ºC	24ºC	55%	45%	40 dB(A)
Quirófanos	26ºC	22ºC	55%	45%	35 dB(A)

Según la modificación del RITE del 13 de abril de 2013, para instalaciones con una potencia útil mayor de 70 kW, existe la obligación de hacer al menos una revisión anual de calidad de aire interior. El RITE debe ser aplicado a todas las instalaciones térmicas llevadas a cabo en nuevas construcciones, así como en edificios en los que se lleve a cabo una reforma, excluyendo únicamente las instalaciones térmicas de procesos industriales, agrícolas o de otro tipo, en la parte que no esté destinada a atender la demanda de bienestar térmico e higiene de las personas.

Para la parte internacional, se ha recogido una serie de estándares:

- Estándar 62-2016 de ASHRAE.
- Estándar 170 de ASHRAE.
- Estándar 55-2013 de ASHRAE.

Local	ASHRAE			
	Temperatura		Humedad relativa	
	Máxima	Mínima	Máxima	Mínima
En todo el centro sanitario	24ºC	21ºC	60%	30%
Quirófanos	24ºC	20ºC	60%	30%

T2. Requisitos funcionales y especificaciones técnicas.

El objetivo principal de esta tarea fue recopilar los principales requisitos que debe cumplir el sistema propuesto. Respecto a los requisitos funcionales hemos localizado los siguientes:

RF	Requisito funcional sobre
RF1, RF2, RF3	**Captura de información**
RF4, RF5, RF6, RF7, RF8, RF9,RF10	**Procesamiento**
RF11, RF12, F13, RF14, RF15, RF16, RF17, RF18, RF19, RF20, RF21, RF22	**Almacenamiento**
RF23, RF24, RF25, RF26, RF27, RF28, RF29, RF30, RF31, RF32, RF33, RF34, RF35	**Exposición de información**

Respecto a los requisitos no funcionales:

RNF	Requisito funcional sobre
RNF1	**Extensibilidad**
RNF2, RNF3	**Escalabilidad**
RNF4, RNF5	**Rendimiento**
RNF6	**Interoperabilidad**
RNF7, RNF8	**Seguridad**
RNF9, RNF10	**Usabilidad**
RNF11	**Portabilidad**
RNF12	**Software**

El estudio y organización de todos los requisitos del sistema se han basado en la prioridad de la utilización de sensores de calidad del aire, motorizar estos datos mediante algoritmos inteligente y la visualización de los mimos de forma sencilla y amigable para el usuario final que eran nuestras metas.

T3. Modelo Conceptual del sistema.

El diseño conceptual del sistema de información planteado por SmartAirQuality debe responder tanto a los requisitos funcionales y no funcionales definidos para el caso de uso del proyecto como a las posibles exigencias de su aplicación en otro entorno con diferentes características. La flexibilidad del sistema y su grado de interoperabilidad deberá adaptarse a las necesidades de diferentes entornos en que la monitorización de la calidad del aire resulte relevante.

Se debe de desarrollar un sistema de información aplicado a la monitorización y análisis de la información de dispositivos a diferente escala, evaluamos la viabilidad de la aplicación de tecnologías tradicionales de almacenamiento y análisis de datos y la posibilidad de aplicación de tecnologías escalables que se puedan adaptar a casos de uso con mayor exigencia.

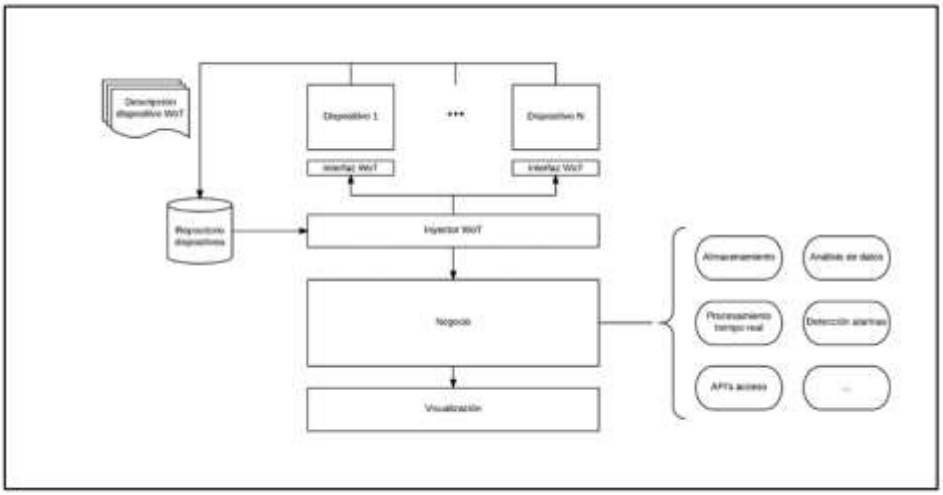

Figura 2: Arquitectura Conceptual.

La primera decisión que hay que tomar es la relativa a la definición del sistema al nivel más amplio. En este aspecto, se suelen contemplar como alternativas la elección de un sistema de información monolítico o la aplicación de un diseño abierto, adscrito al denominado patrón arquitectónico de microservicios.

En este aspecto, teniendo en cuenta los requisitos de escalabilidad del proyecto, asociados al análisis e integración del flujo de datos generados por diferentes sensores de calidad del aire distribuidos en un entorno cerrado, parece conveniente tratar de dividir el trabajo en diversos componentes funcionales que posteriormente trabajen en conjunto mediante alguna técnica de orquestación.

En relación al rendimiento del sistema y su posible escalabilidad, otro concepto a tener en cuenta sería el de la utilización de sistemas distribuidos para la administración de la información y su procesamiento. Estos sistemas, entre los que se engloban las tecnologías Big Data, posibilitan el procesamiento de la información de manera paralela entre diferentes máquinas, a diferencia de un sistema tradicional en que todo el procesamiento se realiza en una única máquina. Estos sistemas, además de permitir procesar un volumen de datos mayor, garantizan la escalabilidad horizontal del sistema, la replicación de datos y la tolerancia a fallos de sus componentes. La aplicación de técnicas Big Data deberá decidirse en base a las diferentes especificaciones funcionales que se identifiquen en el proyecto. A pesar de la adopción generalista de estas tecnologías, no todos los componentes del sistema a desarrollar deben aplicar estas técnicas.

Además de los requisitos de rendimiento de un sistema estas características, también existen los desafíos que suponen el almacenamiento y gestión de grandes volúmenes de datos generados por una red de sensores en forma de series temporales. Para tratar de solventar dicha problemática se tiene que contemplar las diversas tendencias que actualmente se detectan en el mercado del almacenamiento de datos, como el almacenamiento desestructurado, también conocido como NoSQL.

Por último, escenarios de masificación de datos presentan limitaciones en cuanto a la consulta de información que a menudo resultan inasumibles en lo relativo a los tiempos de respuesta. Debido a esto se debe prestar especial atención a los distintos tipos de técnicas de consulta, que principalmente se pueden clasificar en dos grupos: síncronos y asíncronos

Se definen los diferentes módulos independientes y coordinados que definirán tanto a los dispositivos de sensórica como a la arquitectura servidora, representando respectivamente la capa de control y la capa de negocio del sistema a desarrollar en SmartAirQuality.

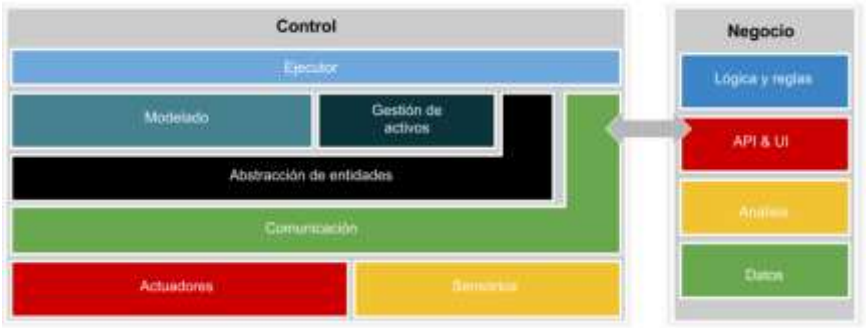

Figura 3: Módulos arquitectura Conceptual.

La capa de control: implica la gestión de una colección de funciones para la continua lectura de datos provenientes de sensores, la aplicación de reglas y lógica sobre la información recibida y el ejercicio del control sobre los sistemas físicos. Tanto la precisión como las frecuencias suelen ser aspectos especialmente relevantes.

Capa de Negocio: conjunto de funciones que recogen datos desde diferentes dominios, principalmente desde la capa de control, realizando a su vez tareas de transformación, persistencia, modelado o analítica con el objetivo de generar un nivel superior de inteligencia operacional. Las operaciones de recopilación de datos y analítica desencadenadas en este nivel, serán complementarias de las que hayan sido desencadenadas en el nivel de control.

Hito 2: Diseño y Desarrollo prototipo IoT.

T1. Prototipo del sistema de medición de calidad del aire.

Se ha realizado una comparación técnica y un estudio de mercado de diferentes sensores que permitirían la medida de parámetros relacionados con la calidad del aire en interiores. Hay numerosos parámetros relacionados con la calidad del aire cuya parametrización resultaría interesante a la hora de garantizar las mejores condiciones posibles de ambientes interiores. Algunos de estos parámetros son: ruido, luminosidad, humedad, temperatura, CO (monóxido de carbono), CO2 (dióxido de carbono), O7A, COVS (compuestos orgánicos volátiles), NOx (óxidos de nitrógeno), SO2 (dióxido de azufre), O3 (ozono), partículas, radón y radiación. Para el desarrollo del proyecto se han probado diferentes sensores que permiten tomar datos de todos los parámetros citados, y en función de los resultados obtenidos se seleccionarán aquellos que cumplan los objetivos de tamaño, peso, volumen, consumo eléctrico y calidad en las medidas.

T2. Diseño de la envolvente.

Primer diseño:

La envolvente o carcasa que habrá que diseñar para este proyecto contendrá tanto el dispositivo de procesamiento (Raspberry Pi), como todos los sensores asociados, necesarios para la captación de datos

de calidad del aire. Esta carcasa también deberá alojar una batería que sirva para alimentar los dispositivos en un entorno en el que muy probablemente no exista alimentación cableada.

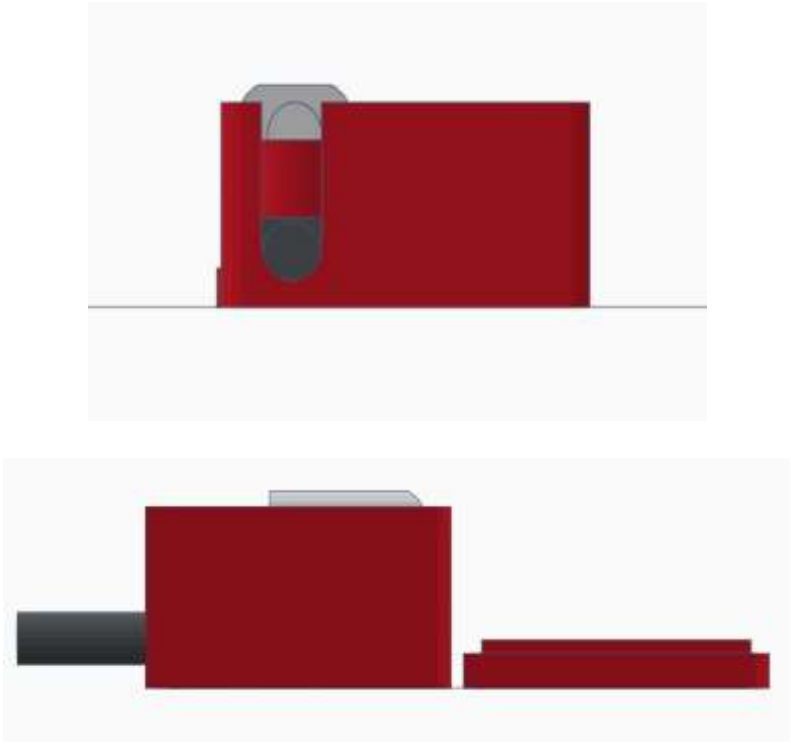

Figura 4: Primer diseño envolvente.

Contiene los huecos necesarios para los sensores. En negro, el sensor de CO2 se encuentra al fondo, anclado con unas patillas que lo sujetan. Arriba y en color blanco, sujeto por un pivote y los soportes que también sujetan el sensor de CO2, se encuentra el sensor de temperatura y humedad (DHT22). El hueco grande al lado de los sensores está pensado para alojar la Raspberry junto a la batería.

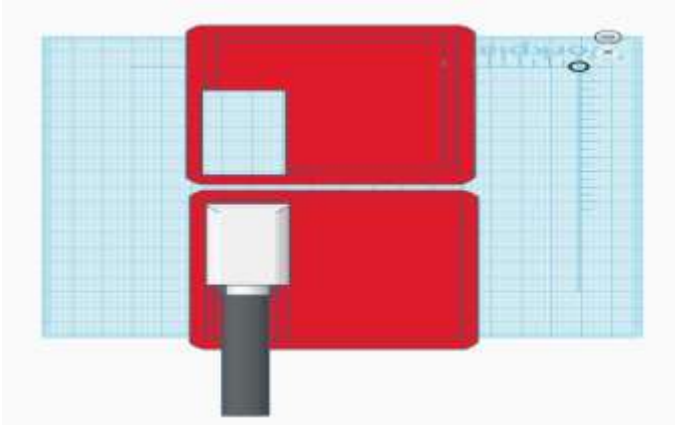

Figura 5: Primer diseño envolvente. Vista superior.

La tapa está diseñada para quedar sujeta sin anclajes al contenedor y tiene una abertura para que el sensor de temperatura funcione adecuadamente. Con el mismo propósito, el sensor de CO2 sobresale del montaje para capturar adecuadamente los datos de concentración de CO2.

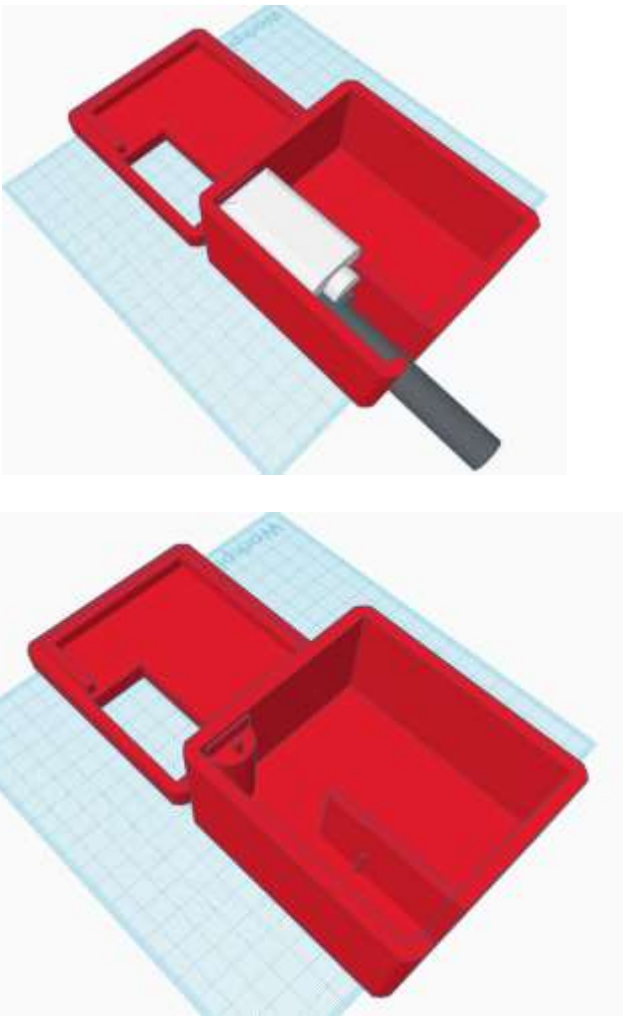

Figura 6: Primer diseño envolvente. Vista isométrica.

Segundo diseño:

Este diseño también cuenta con un formato de caja, en el que irían alojados todos los sensores, así como el dispositivo de procesamiento Raspberry. Toda la sensórica se alojaría en el mismo hueco, ganando así en robustez.

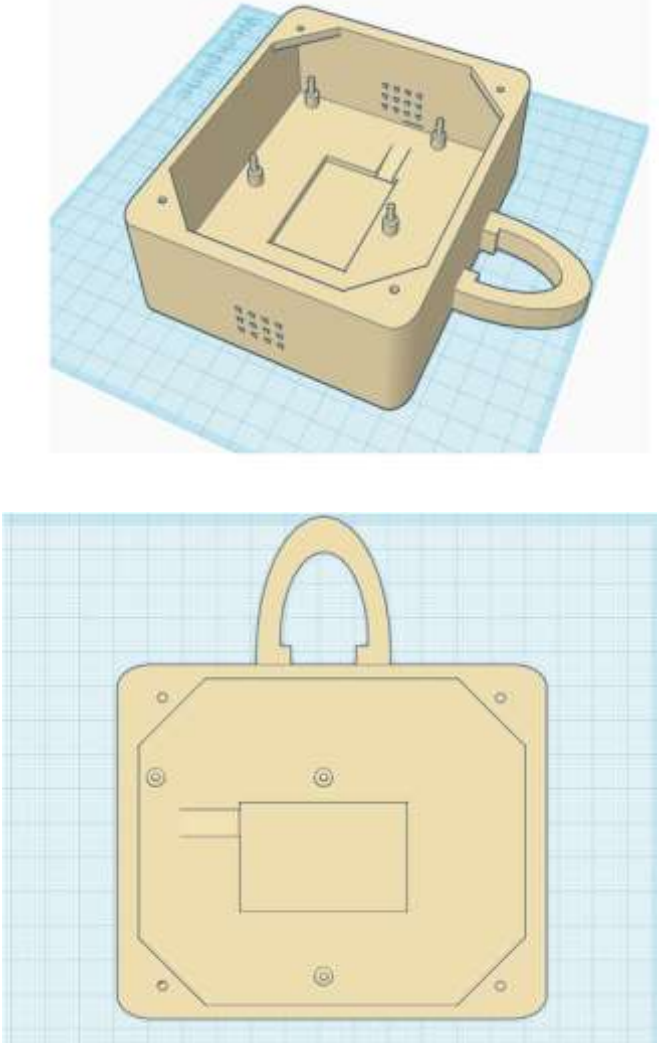

Figura 7: Segundo diseño envolvente.

Como puede verse en la figura, existen unos soportes que servirán de anclaje para la Raspberry e impedirán que se mueva. En el fondo hay un hueco pensado para una futura placa de carga inalámbrica mediante inducción. Además, contaría con un asa que permitiría colgarlo. La tapa, que sería un simple rectángulo, iría atornillada a los agujeros de que dispone la caja en sus cuatro esquinas

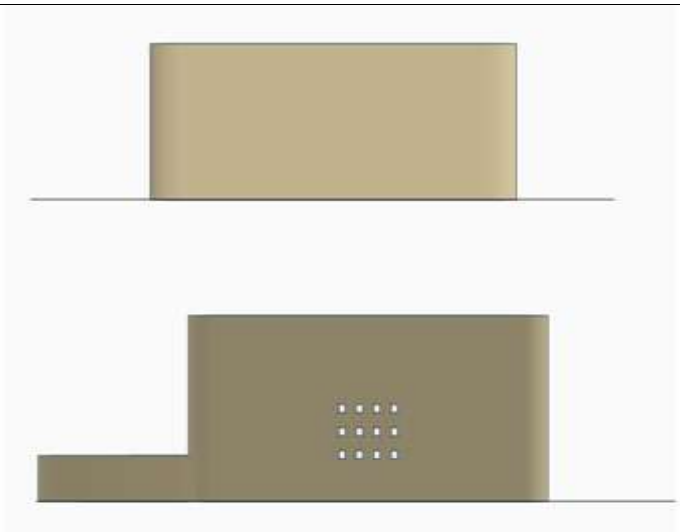

Figura 8: Segundo diseño envolvente. Alzado.

El diseño elegido será el diseño 1 ya que, aunque el segundo es mucho más robusto, el primero permite captar los datos de calidad del aire de una manera más óptima, al tener los sensores adecuadamente orientados hacia el exterior. Por otro lado, el diseño de tapa que encaja, en vez de ir atornillada, permite un montaje y desmontaje más cómodo y sencillo. La colocación de los sensores en la parte exterior permite también evitar el efecto "invernadero" que produciría el propio calor que desprende la Raspberry, y que alteraría los datos tanto de temperatura como de humedad en gran medida, y el de CO_2 en menor medida.

Hito 3: Estudio e implementación estandarización WoT.

T1. Estudio del estado del arte.

El concepto de Internet de las Cosas (Internet of Things, IoT) integra una serie de técnicas y tecnologías que dotan de nuevas capacidades a los diferentes dispositivos de nuestra vida diaria, permitiendo que estos puedan recabar datos de su entorno, almacenarlos y compartirlos.

Una tecnología clave la forman los sensores, encargados de captar las características físicas del entorno. Su principal objetivo es obtener información a través de medidores de temperatura, humedad, radiación, etc. Los sensores son capaces de almacenar y procesar información utilizando microcontroladores, y de transmitirla a través de un canal de comunicación, normalmente inalámbrico

Estándares IoT

En este apartado se recogen algunos los estándares más importantes en materia de IoT:

- *WoT Community Group*: Estándar propuesto por W3C para el mundo de IoT y basado en tecnologías web (WoT). Se basa en protocolos ya existentes como HTTP y CoAP para el acceso a las Things; JSON-LD para la definición y representación de las mismas y WebSocket para el sistema de comunicaciones.
- *AllJoin*: En este estándar se propone un modelo en el cual un dispositivo contiene una serie de

aplicaciones, formadas a su vez por eventos y acciones que emite y recibe. Cada uno de estos dispositivos se anuncia en la red, exponiendo sus características en forma de metadatos en los que se incluyen sus aplicaciones, eventos y acciones.

- *Thread:* Estándar diseñado por Thread Group para conectar un gran número de dispositivos entre sí y con la nube. Se utilizan estándares abiertos como ZigBee o CoAP y protocolos como IPv6 y 6LoWPAN.

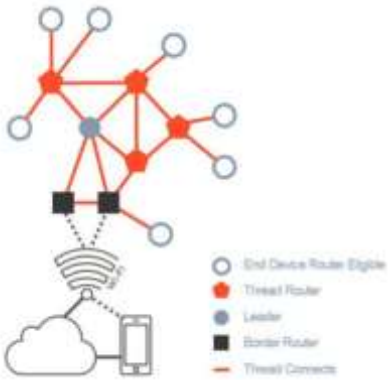

Figura 9: Diagrama Thread.

- *IoTivity*: Se trata de una propuesta de estandarización emergente creada por compañías como Microsoft, Samsung o Intel, que pretende unificar la manera en que los dispositivos se conectan e interactúan entre ellos. Las librerías y APIs que ofrece esta plataforma se conocen con el nombre de IoTivity y se han desarrollado para multitud de plataformas como Linux, Android o Raspberry.
- *SensorThings API (OGC)*: Pretende ser un estándar de interconexión de sensores y dispositivos IoT a través de la web. Se trata de un estándar abierto y de libre acceso, uso y modificación. Se basa en estándares abiertos y de amplia aceptación como protocolos web.
- *Sensor ML*: Este estándar planteado por OGC se basa en una API que pretende unificar las comunicaciones entre dispositivos de forma semántica, en este caso basada en el formato XML para almacenar los estados y las propiedades de las cosas.
- *IEEE Internet of Things*: Dentro de la organización IEEE existe una asociación encargada de definir y consensuar estándares, la IEEE Standard Association, que actualmente desarrolla un estándar referido a IoT.

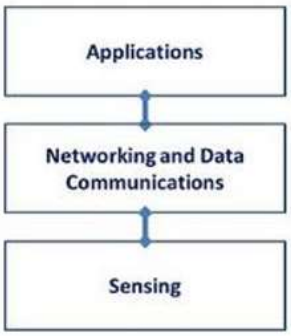

Figura 10: Diagrama IEEE IoT

Las diferentes capas son:

1. Aplicación (Applications): aquellos programas que se comunica, recaban y/o muestran datos de los sensores.
2. Redes y Comunicaciones (Networking and Data Communications): la manera en que tanto los sensores como las aplicaciones se conectan entre si y comparten datos.
3. Sensores (Sensing): dispositivos finales que miden estados del mundo físico.

A continuación, hemos comparado todos los estándares, en la que se tiene en cuenta los siguientes aspectos:

- Reconocimiento como estándar: si es estándar se encuentra reconocido y certificado por alguna organización de estandarización oficial.
- Estándar abierto: si las bases son de libre acceso y modificación.
- Heterogeneidad de sensores: si el estándar permite la interconexión de sensores de múltiples tipos.
- Definición de Things: si aportan un modelo de definición de dispositivos.

Estándar	*Reconocimiento como estándar*	*Estándar abierto*	*Heterogeneidad de sensores*	*Definición de las Things*
WoT-W3C	✗	✓	✓	✓
AllJoin	✗	✓	✓	✓
Thread	✗	✗	✓	✗
IoTivity	✓	✓	✓	✓
Sensor Things API	✓	✓	✓	✗
Sensor ML	✓	✓	✓	✓
IEEE IoT	✗	✓	✓	✓

Servicios

Dentro del mundo de Internet de las Cosas, las empresas han ido generando soluciones que permiten crear una red de sensores y actuadores de mayor o menor medida, basadas en tecnologías existentes, e incluso han creado sus propios protocolos

- *IBM Waston Iot*: Conjunto de APIs diseñadas por IBM como solución para integrar dispositivos IoT a través de una conexión a Internet, implementadas en diferentes lenguajes de programación.

- *AT&T: T&T:* es una compañía multinacional americana de telecomunicaciones y servicios de telefonía que ofrece una serie de servicios que permiten consultar y saber qué dispositivos deben estar conectados, qué plataformas utilizar y permiten a las empresas hacer una transición a Internet de las cosas con el mínimo impacto posible.
- *Thing Worx*: Forma parte de la compañía Parametric Technology Corporation, se dedica a la creación de soluciones IoT para empresas de manera rápida y centralizada.

Figura 11: Diagrama Thing Worx.

- *AWS IoT:* Plataforma de código abierto que permite conectar dispositivos tanto con los servicios web de Amazon (AWS) como con otros dispositivos. Utiliza mensajes MQTT y HTTP para enviar y recibir mensajes desde y hacia los dispositivos.
- *Microsoft Azure*: Azure IoT implementa una arquitectura basada en un hub que concentra los dispositivos IoT dentro de una red, la cual cuenta con comunicaciones seguras y bidireccionales entre dispositivos. Las soluciones ofrecidas se basan en dos grupos: orientadas a monitorización remota y basadas en mantenimiento predictivo.

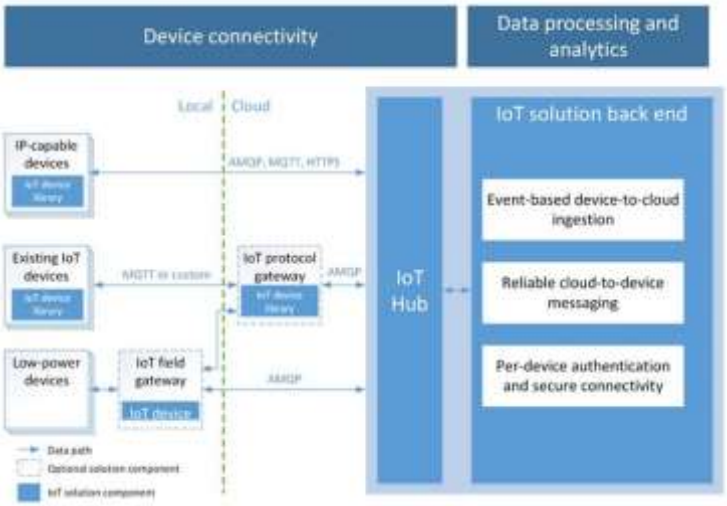

Figura 12: Diagrama Microsoft Azure IoT.

Protocolos de comunicación

El número de dispositivos conectados a Internet aumentan a gran velocidad, convirtiendo en un reto conectar y comunicar de manera correcta a todos ellos. Es fundamental permitir que los dispositivos, además de conectarse entre ellos, puedan conectarse con servidores capaces de compartir datos.

- *REST API*: REST (Representational State Transfer) es una arquitectura web que utiliza como soporte el protocolo HTTP (Hipertext Transfer Protocol). Con ella es posible crear servicios accesibles por cualquier dispositivo que entienda HTTP
- WebSockets: Las aplicaciones en tiempo real necesitan mantener una conexión abierta, estable y ligera que permita la transmisión bidireccional de datos entre cliente y servidor. WebSocket aparece para cubrir esta necesidad, aportando un modo de interconectar aplicaciones web con aplicaciones de escritorios y móviles, usando un sistema de paquetes ligero

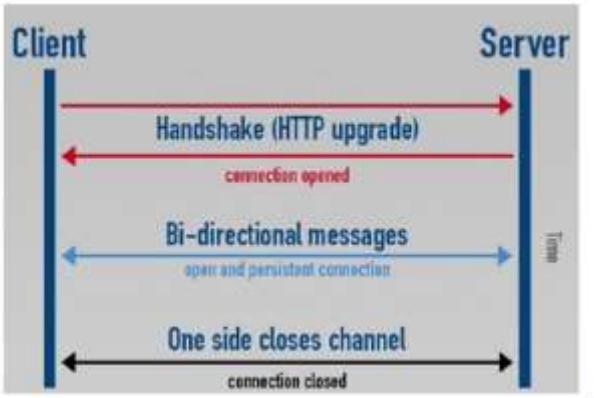

Figura 13: WebSockets.

- *MQTT:* Este protocolo se encuentra estandarizado por OASIS y es un estándar ISO/IEC. Se caracteriza por ser ligero, asíncrono y que permite una comunicación cliente-servidor abierto y fácil de implementar. Este protocolo está pensado para su uso en dispositivos o localizaciones donde el código a implementar sea limitado o las capacidades de la red sean muy reducidas.

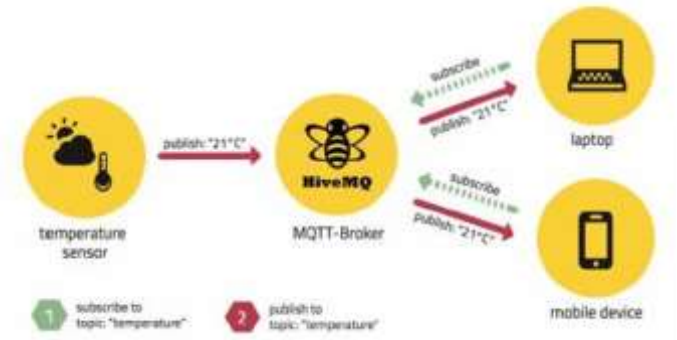

Figura 14: MQTT

- *CoAP:* Es un protocolo pensado para dispositivos simples que les permite comunicarse de forma interactiva mediante Internet. Debido a su poca carga de mensajes es ideal para dispositivos con poca capacidad, limitaciones de batería o acceso limitado a Internet.
- *STOMP:* Protocolo orientado a texto, muy similar a HTTP, que permite a los clientes conectarse a un bróker para que se puedan comunicar fácilmente entre diferentes tipos de aplicaciones y plataformas. Utiliza el protocolo TCP para enviar y recibir mensaje.
- *AMQP:* Este protocolo implementa un gran número de funcionalidades como colas de mensajes, publicación y suscripción a recursos, enrutamiento flexible o seguridad. Por ende, ofrece múltiples posibilidades de configuración y se adapta a redes empresariales de gran tamaño.
- *RabbitMQ:* Se trata de un bróker polígloto, es decir, capaz de recibir y enviar mensajes a protocolos STOMP, WebSocket, MQTT y AMQP, por lo que se puede utilizar como puente entre dispositivos que utilicen diferentes protocolos.
- *XMP*P: Protocolo de mensajería instantánea libre y de código abierto, está pensando para la comunicación en tiempo real y para la detección de presencia. Utiliza los servicios de mensajería de aplicaciones muy conocidas como Facebook o WhatsApp.

Los protocolos recogidos anteriormente son solo una parte de todos los que pueblan IoT, no obstante, CoAP y MQTT son los más utilizados en esta tecnología. Se han tenido en cuenta los siguientes aspectos: Estándar: si el protocolo se encuentra o no estandarizado por alguna organización de estandarización o certificación.

- Transporte: qué protocolo de transporte utiliza.
- Comunicación: qué modelo de comunicación utiliza.
- Autodescubrimiento: si el protocolo implementa funciones de autodescubrimiento.
- RAM KB: espacio que ocupa el protocolo.
- Seguridad: si implementa o no seguridad.
- Topología: qué estructura tiene la red que utiliza para las comunicaciones.

Protocolo	Estándar	Transporte	Comunicación	Auto desc.	RAM KBs	Seguridad	Topología
HTTP/ REST	✓ RFC 2068	TCP, UDP	Petición/ respuesta	✗	10KB	HTTPS	Cliente/ servidor
WebSockets	✓ RFC 6455	TCP	Conexión bidireccional	✗	200KB	✗	Cliente/ servidor
MQTT	✓ PRF 20922	TCP	Publicación/ suscripción	✗	10KB	TLS	Árbol
CoAP	✓ RFC 7252	UDP	Petición/ respuesta	✓	10KB	DTLS	Árbol
STOMP	✗	TCP	Publicación/ suscripción	✗	10KB	¿?	Cliente/ servidor
AMQP	✓ ISO/IEC 19464	TCP	Publicación/ suscripción	✗	10KB	TLS	Árbol
RabbitMQ	✗	TCP	Publicación/ suscripción	✗	¿?	TLS	Árbol
XMPP	✓ RFC 6120 RFC 6121 RFC 6122	TCP	Publicación/ suscripción	✗	10KB	TLS + SASL	Cliente/ servidor

Protocolos de autodescubrimiento

Los protocolos de autodescubrimiento permiten encontrar dispositivos y servicios en una red. Una vez encontrado el servicio, el usuario será capaz de acceder a él y utilizarlo. Cada protocolo de autodescubrimiento está formado como mínimo por dos participantes: un cliente, encargado de solicitar la búsqueda de un servicio, y un servidor, que aloja a todos los servicios.

- *UPnP*: Es un protocolo de autodescubrimiento abierto y un estándar ISO. Se encuentra apoyado e implementado en plataformas como Windows y en muchos dispositivos. Este protocolo permite obtener una dirección IP mediante HHTP dentro de una red, obteniéndola de un servidor DNS o DHCP.
- *Physical Web*: Se trata de una tecnología que permite descubrir servicios y realizar acciones por proximidad. Se basa en beacons Bluetooth, pero también soporta el descubrimiento a través de Wifi con UPnP.
- *Wifi Aware*: Este protocolo se basa en la distancia entre dos dispositivos Wifi Aware para realizar conexiones solo en el caso de que el dispositivo detectado coincida con las preferencias del usuario. Se trata de una conexión M2M que permite la comunicación de pequeños paquetes de datos. Para mayores tasas de transferencia se usará Wifi.

Seguridad

El número de dispositivos conectados a internet es cada vez mayor, lo que supone que la cantidad de datos expuestos en la red aumente considerablemente. Como consecuencia, los aspectos relacionados con la seguridad de la información dan cada vez más que hablar.

- *Brechas IOT:* Los dispositivos conectados a la red suelen tener una complejidad baja, de forma que los fabricantes implementan sus propias soluciones, descartando en muchos casos seguir un

operativo global como sucede con los ordenadores o Smartphone.

- *Seguridad en el software:* La mayoría de los dispositivos utilizan versiones ajustadas de sistemas operativos comunes para abaratar costes de fabricación, lo que supone un riesgo para la seguridad, ya que cuando se detectan vulnerabilidades sobre dichas plataformas son explotables sobre todos los dispositivos que las incorporan, facilitando a los atacantes la entrada a infinidad de dispositivos.
- *IoTSF:* La Internet of Things Safety Foundation es una fundación sin ánimo de lucro que se encarga de comprobar que los dispositivos IoT no padecen vulnerabilidades o defectos, además de ofrecer asistencia para la seguridad a los proveedores y usuarios finales.

T2. Diseño de arquitectura Wot y adaptabilidad a sensórica SMARTAIRQUALITY.

Con objeto de asegurar la interoperabilidad y extensibilidad del sistema desarrollado, el proyecto SmartAirQuality se adherirá a los estándares definidos por el grupo de trabajo en Web de las Cosas (Web of Things) promovido por el consorcio W3C.

La iniciativa para la Web de las Cosas (WoT), implica consideraciones técnicas a diversos niveles: diseño arquitectónico de sistemas informáticos, protocolos de comunicación entre dispositivos, etc.

Como paso previo a la implementación de los estándares se ha realizado un exhaustivo estado del arte, tanto en tecnologías habilitadoras para la web de las cosas, como en los mecanismos de especificación, implementación y definición de dispositivos (Things).

Web of Things es una iniciativa del consorcio W3C para combatir las deficiencias en interoperabilidad entre las diferentes plataformas de IoT. Los desarrolladores deben enfrentarse con silos aislados de datos, altos costes de desarrollo y, por tanto, importantes limitaciones en el mercado. Se trata de una situación muy similar a la que se producía en Internet antes de la especificación de los protocolos web, con la proliferación de diferentes plataformas de comunicaciones no interoperables entre sí.

Esta fragmentación se ataca mediante una plataforma de APIs independientes para el desarrollo de aplicaciones aportando, además, fórmulas para el autodescubrimiento e interoperabilidad entre ellas. La aproximación de la Web of Things se basa en la definición de metadatos para describir los datos y los modelos de interacción expuestos a las aplicaciones, así como los requerimientos en cuanto a comunicaciones y seguridad que deben cumplir las diferentes plataformas que implementen el estándar. Otro aspecto relativo a la web de las cosas es la necesidad de permitir a las diferentes plataformas compartir un significado común de la información durante los procesos de intercambio de datos. Para ello, se busca habilitar la expresión de definiciones semánticas de las “Things” y de las características específicas de los diferentes dominios en los que operan.

El desarrollo de esta tarea en el marco del proyecto SmartAirQuality es complejo dado que exige la comunicación constante con el grupo de trabajo (Work Group) de W3C para asegurar la compatibilidad de los sistemas desarrollados con un estándar que actualmente está sometido a un proceso de cambio y actualización continuo (Se realizan cambios y actualizaciones al menos una vez al mes).

A pesar de tratarse de una tarea ardua y complicada, garantizar la adecuación a las directivas del W3C, asegurará que el sistema desarrollado esté en la vanguardia tecnológica e investigadora en materia de Web of Things. Algunos de los resultados y experiencias obtenidas durante este proyecto han sido

comunicados al grupo de trabajo para su incorporación al estándar.

Las acciones para la estandarización en Web of Things del proyecto SmartAirQuality se están desarrollando en los siguientes ejes:

1. Estandarización arquitectónica. La arquitectura de sistemas en la web de las cosas descansa sobre un modelo de artefacto llamado: WoT Servient. El WoT Servient es el nodo principal de la web de las cosas. Su función es albergar, gestionar e interconectar “Things”.

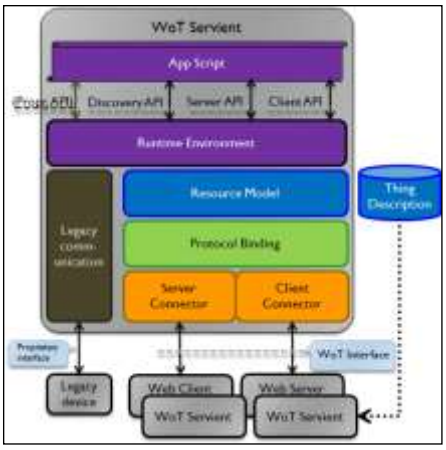

Figura 15: Wot Servient

2. En el proyecto SmartAirQuality, se está adecuando el diseño de todos los dispositivos (sensores) y sistemas (procesamiento de datos, etc.), para adaptarse al modelo de WoT Servients. Actualmente, se está desarrollando un starter para el desarrollo rápido de WoT Servients basados en el modelo de microservicios sobre el framework Java Spring Boot.
3. Descripción semántica. La interoperabilidad entre dispositivos se define mediante descriptores de things (Thing Descriptions). La descripción de servicios en el ámbito de la web of things se realiza mediante especificaciones semánticas. A finales del mes de enero de 2017, ha comenzado el proceso de recopilación de ontologías para la definición de los diferentes dominios de negocio. No es objeto del proyecto SmartAirQuality reinventar definiciones semánticas para dominios pre-existentes. Por tanto, se seleccionarán y extenderán (utilizando los mecanismos habilitados a tal fin), las ontologías ya publicadas. En la fecha actual, se han definido especificaciones únicamente a nivel sintáctico, que definen las mediciones tomadas por los diferentes dispositivos.
4. Descubrimiento y autodescubrimiento de Things. El componente central para el registro y descubrimiento de Things dentro de la web es el Thing Repository. Se trata de un modelo de repositorio, bien centralizado, bien distribuido o bien local, en el que los diferentes dispositivos se registran utilizando su definición semántica (Thing Description), ofrecen sus servicios para su utilización por parte de otras “Things” y pueden localizar y conectar con otros dispositivos en la misma red. Actualmente, no existe una especificación formal para la implementación de repositorios de Things. Para el proyecto SmartAirQuality, se está desarrollando un repositorio respaldado por una base de datos distribuida Mongo DB (dado que el formato de intercambio de descripciones de Things es JSON, que es precisamente el modelo de documento gestionado por MongoDB). El repositorio que está siendo desarrollado para este proyecto expone, a su vez, sus servicios de directorio como una “thing” virtual, de forma que pueda ser utilizado por

cualesquiera otros dispositivos.

5. Comunicaciones. La definición de protocolos de comunicación en la Web of Things se realiza únicamente a nivel de aplicación, delegando en otros estándares de bajo nivel las particularidades de cada interacción entre dispositivos. Los protocolos predefinidos como estándar son CoAP, REST, Web Sockets o MQTT. Para el proyecto SmartAirQuality, se ha implementado el protocolo REST para comunicaciones síncronas y el protocolo Kafka para asíncronas. El protocolo Kafka no está incluido por el momento entre los estándares del W3C, por este motivo, se está realizando la migración a otro protocolo M2M soportado: MQTT. Sin embargo, todas las especificaciones al respecto del protocolo MQTT (aunque está considerado estándar) no han sido desarrolladas hasta la fecha. El proyecto SmartAirQuality está siendo, en la actualidad, la primera prueba de concepto para este protocolo dentro del grupo de trabajo del W3C.

El sistema desarrollado en el proyecto SmartAirQuality se ajustará al modelo arquitectónico propuesto por el grupo de trabajo Web of Things, denominado "homehub". Se trata de un modelo de sistemas y comunicaciones diseñado específicamente para la integración de sensórica (especialmente ambiental) en el hogar (o en cualquier otro tipo de instalaciones).

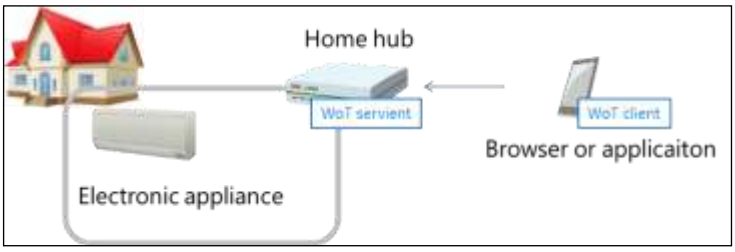

Figura 16: Esquema del modelo homehub (fuente W3C)[1]

T3. Implementación.

En los meses finales de la anualidad de 2016 se ha iniciado la implementación del sistema diseñado como resultado de la ejecución de la tarea "T2. Diseño de la Arquitectura WoT y Adaptabildiad a sensórica".

[1] https://github.com/w3c/wot/blob/master/architecture/wot_servient_on_homehub.png

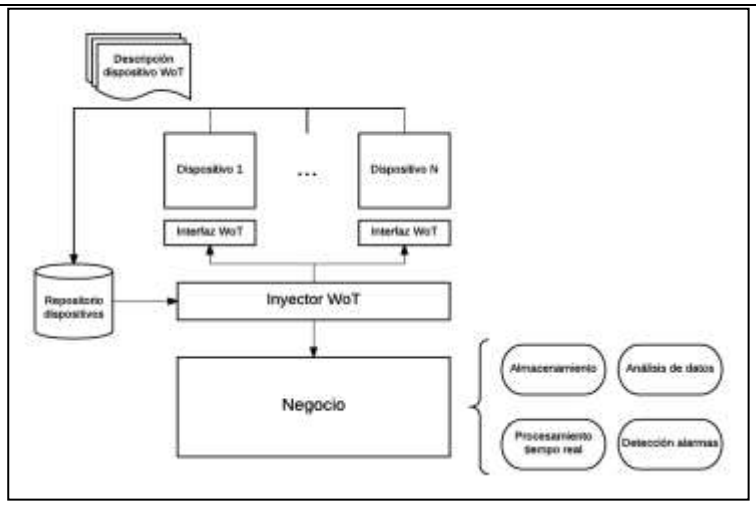

Figura 17: Esquema de la arquitectura (vista de implementacion)

El desarrollo del sistema se ha llevado a cabo siguiendo una metodología iterativa multifase. De acuerdo con la planificación inicial, se espera llevar a cabo tres iteraciones de una duración aproximada de dos meses para cada una de ellas, quedando un mes como holgura para asumir las desviaciones y ajustes que puedan darse en cada una de ellas.

A la finalización de cada iteración, se espera obtener una versión plenamente funcional y testeable del sistema, que será refinada, ampliada y mejorada en las iteraciones posteriores.

Así mismo, cada una de estas iteraciones está compuesta por tres fases que suponen, a su vez, iteraciones sobre el mismo producto.

1. Fase 1. Desarrollo de microservicios. En la primera fase, se trabajará sobre los requerimientos específicos de cada uno de los componentes del sistema actuando como nodos aislados. En esta fase, se hará uso extensivo de sistemas de apoyo como mocks y stubs que simularán las interacciones con otros componentes del sistema.
2. Fase 2. Interacción Ad-Hoc. En la segunda fase de la iteración se desarrollan, reutilizando los interfaces de comunicaciones construidos en la fase anterior sobre la base de sistemas mock y stub, los protocolos (ad-hoc) para intercomunicar los microservicios que conforman la arquitectura del sistema SmartAirQuality.
3. Fase 3. Web of Things. Finalmente, los protocolos e interfaces ad-hoc desarrollados en la fase 2 serán adaptados a las especificaciones marcadas por el grupo de trabajo para la Web of Things siendo, a la finalización de la iteración, plenamente interoperables bajo el estándar.

A fin de la anualidad 2016, ha concluido la primera iteración sin desviaciones sensibles respecto a la planificación realizada inicialmente.

El modelo iterativo multifase planteado permitirá, en la Fase 3 de la siguiente iteración recuperar, de nuevo, la compatibilidad con el estándar. Debido al bajo nivel de madurez de las especificaciones y al continuo cambio al que están sometidas, las variaciones producidas en las mismas, eran esperadas y la causa principal para seleccionar el modelo de desarrollo iterativo multifase adoptado.

Desde el punto de vista tecnológico, a la finalización de la anualidad 2016, se han desarrollado los siguientes componentes de la arquitectura:

- Plataforma cloud. De acuerdo con el diseño arquitectónico, el sistema SmartAirQuality, operará siguiendo un modelo de microservicios interoperables sobre un cluster que pueda operar en configuraciones centralizadas, cloud o edge.
- Sistema de recepción de datos de sensórica. El envío de datos, por parte de los dispositivos sensores, hacia la arquitectura central de análisis se realiza mediante un protocolo de comunicaciones asíncrono (pub/sub) implementado sobre una cola de mensajes Kafka[2].
- Sistema de almacenamiento de datos. En el sistema SmartAirQuality está planteada la concentración y agregación de toda la información recibida en tiempo real (near) por parte de los sensores. Esta información será almacenada de forma centralizada (al menos, desde el punto de vista lógico pues, en realidad, el almacén de datos puede ser distribuido). La implementación del modelo de datos centralizado ha sido llevado a cabo utilizando el sistema NoSQL: Cassandra[3].
- Soporte para la ejecución de algoritmos de análisis de datos en tiempo real. El análisis de los datos crudos recibidos por la sensórica en tiempo real (streaming) es uno de los objetivos del proyecto SmartAirQuality. Para ello, se ha incorporado al sistema una plataforma para la ejecución de algoritmos (o cualquier otro tipo de operaciones) de forma paralelizable sobre el cluster.
- Servicio de directorio para dispositivos. Este servicio actúa como repositorio de things (Thing Repository) centralizado y sus funciones son las de proporcionar mecanismos de indexación y búsqueda de los diferentes dispositivos que están volcando mediciones al sistema central de SmartAirQuality.
- Servicio de datos al exterior (visualización). Durante la anualidad de 2017 se espera desarrollar un sistema para la visualización de la información proporcionada por el sistema SmartAirQuality que asista al operador humano en la toma de decisiones de acuerdo a los niveles de contaminantes en el aire. Como paso previo al desarrollo de este cuadro de mandos o dashboard, se está desarrollando un servicio que actúe como fachada para el nodo central de monitorización, sirviendo la información temporalizada y previamente analizada, lista para su visualización.

Anualidad 2017

Hito 3: Estudio e implementación estandarización WoT.

T2. Diseño de arquitectura Wot y adaptabilidad a sensórica SMARTAIRQUALITY

Según se ha indicado en la explicación de los trabajos desarrollados en esta tarea durante 2016, durante dicha anualidad se diseñó prácticamente en su totalidad la arquitectura WoT deseada, articulada en

[2] https://kafka.apache.org/
[3] http://cassandra.apache.org/

torno a los siguientes ejes, que no volveremos a explicar para no repetir información contenida en este mismo informe:

1. Estandarización arquitectónica.
2. Adecuación de los sensores y sistemas al modelo WoT Servients.
3. Descripción semántica.
4. Descubrimiento y autodescubrimiento de Things.
5. Comunicaciones.

A principios del año 2017 se iniciaron los trabajos del *Working Group*[4] del grupo Web of Things del W3C. Este grupo se encuentra actualmente en funcionamiento activo y está planeado que finalice sus operaciones a finales del año 2018. El objetivo principal del WG es proporcionar una estandarización definitiva de los bloques fundamentales de la Web of Things identificados por el *Interest Group*, que son los que actualmente se encuentran definidos y que han sido descritos de manera general en la descripción de los trabajos de esta tarea durante la anualidad 2016.

El trabajo del WG se traduce a su vez en actualizaciones recurrentes a los estándares que requieren de un reajuste del diseño planteado. Por ello, los trabajos de esta tarea durante la anualidad 2017, han consistido en la finalización del diseño de la arquitectura, incluyendo la adaptación del diseño planteado en 2016 a dichas actualizaciones y a las buenas prácticas discutidas dentro del grupo.

A continuación, se resumen los trabajos más destacables que se han realizado en esta tarea durante el inicio de la anualidad 2017.

- Centralización de esfuerzos en una plataforma para un desarrollo más dinámico (Python) respecto al *starter* en Java Spring Boot previamente mencionado. El objetivo es poder trazar un mejor paralelismo con los trabajos realizados por el grupo W3C sobre Node.JS.
- Integración de simplificaciones y propuestas de actualización sobre el formato de la Thing Description. Por ejemplo, la manera en la que se describen los endpoints dentro de la Thing Description.
- Aclaraciones en la terminología o eliminación y modificación de campos para mayor simplicidad. Todas estas cuestiones afectan a bajo nivel el diseño de las *Electronic Appliances* y el *Home Hub* en SmartAirQuality.
- Discusiones al respecto del encaje del patrón arquitectónico REST dentro del contexto de la Scripting API y los WoT Servients. REST resulta un componente ubicuo dentro de la Web, por lo que se evaluó la posibilidad de proporcionarle una posición más prominente dentro del estándar. Esto a su vez se hubiese traducido en modificaciones en el diseño del WoT Servient y la Scripting API.

T3. Implementación.

A lo largo de los primeros meses del año 2017, las especificaciones del grupo de trabajo para la Web of Things del consorcio W3C han sufrido varias actualizaciones, quedando por ello el sistema SmartAirQuality fuera de compatibilidad con el estándar.

[4] https://www.w3.org/2016/12/wot-wg-2016.html

El modelo iterativo multifase planteado permitió, en la Fase 3 de la siguiente iteración recuperar, de nuevo, la compatibilidad con el estándar. Debido al bajo nivel de madurez de las especificaciones y al continuo cambio al que están sometidas, las variaciones producidas en las mismas, eran esperadas y la causa principal para seleccionar el modelo de desarrollo iterativo multifase adoptado.

Desde el punto de vista tecnológico, a la finalización de la anualidad 2017, se han desarrollado los siguientes componentes de la arquitectura:

- Plataforma cloud. Durante la anualidad 2017, se ha realizado la parametrización óptima y, especialmente, su empaquetamiento como producto final que permite su redespliegue en otros clusters externos.
- Sistema de recepción de datos de sensórica. El envío de datos, por parte de los dispositivos sensores, hacia la arquitectura central de análisis se realiza mediante un protocolo de comunicaciones asíncrono (pub/sub) implementado sobre una cola de mensajes Kafka[5].
- Sistema de almacenamiento de datos. En el sistema SmartAirQuality está planteada la concentración y agregación de toda la información recibida en tiempo real (near) por parte de los sensores. La implementación del modelo de datos centralizado ha sido llevado a cabo utilizando el sistema NoSQL: Cassandra[6].
- Soporte para la ejecución de algoritmos de análisis de datos en tiempo real. El análisis de los datos crudos recibidos por la sensórica en tiempo real (streaming) es uno de los objetivos del proyecto SmartAirQuality. Para ello, se ha incorporado al sistema una plataforma para la ejecución de algoritmos (o cualquier otro tipo de operaciones) de forma paralelizable sobre el cluster.
- Servicio de directorio para dispositivos. Este servicio actúa como repositorio de things (Thing Repository) centralizado y sus funciones son las de proporcionar mecanismos de indexación y búsqueda de los diferentes dispositivos que están volcando mediciones al sistema central de SmartAirQuality.
- Servicio de datos al exterior (visualización). Durante la anualidad de 2017 se desarrolla un sistema para la visualización de la información proporcionada por el sistema SmartAirQuality que asista al operador humano en la toma de decisiones de acuerdo a los niveles de contaminantes en el aire.

Hito 4: Estudio e implementación estandarización WoT.

T1. Recepción de datos.

El objetivo de esta tarea consiste en definir un sistema de comunicación que conecte los dispositivos con una base de datos donde se almacenen y procesen los datos emitidos por dichos sensores para su posterior tratamiento.

[5] https://kafka.apache.org/
[6] http://cassandra.apache.org/

Figura 18: Dispositivo con sensores.

Para ello se analiza el diseño del módulo de recepción de datos y el sistema de mensajería a utilizar en el proyecto.

A la hora de desarrollar un sistema de información aplicado a la monitorización y análisis de la información de dispositivos a diferente escala, se han evaluado las tecnologías tradicionales de almacenamiento y análisis de datos y la posible aplicación de tecnologías escalables que se puedan adaptar a casos de uso con mayor exigencia.

Teniendo en cuenta los requisitos de escalabilidad del proyecto, asociados al análisis e integración del flujo de datos generados por una red de sensores distribuidos en un entorno cerrado, se divide el trabajo en diversos componentes funcionales que posteriormente trabajen en conjunto mediante una técnica de orquestación.

En relación al rendimiento del sistema y su posible escalabilidad, se utilizan sistemas distribuidos para la administración de la información y su procesamiento. Estos sistemas, se engloban las tecnologías Big Data y posibilitan el procesamiento de la información de manera paralela entre diferentes máquinas, a diferencia de un sistema tradicional en que todo el procesamiento se realiza en una única máquina. Estos sistemas, además de permitir procesar un volumen de datos mayor, garantizan la escalabilidad horizontal del sistema, la replicación de datos y la tolerancia a fallos de sus componentes.

Respecto al rendimiento de un sistema estas características, el almacenamiento y gestión de grandes volúmenes de datos generados por una red de sensores en forma de series temporales es de vital importancia. Para tratar de solventar dicha problemática se utiliza el almacenamiento desestructurado, también conocido como NoSQL.

Definición del sistema de mensajería y comunicaciones.

Protocolo de comunicación.

El sistema de mensajería utilizado para las comunicaciones entre los dispositivos de medición y la base de datos tendrá que ser un protocolo que permita la comunicación de miles de datos por segundo y que asegure la llegada de los mismos al destino. Para ello, existen diferentes protocolos, los cuales se

expondrán a continuación con el objetivo de seleccionar aquel que mejor se adapte a esta solución. Estos protocolos servirán como puente para comunicar la aplicación integrada en el dispositivo con el protocolo propio de la base de datos.

HTTP REST.

HTTP (Hypertext Transfer Protocol) es el protocolo utilizado en internet para la difusión de hipertexto. El hipertexto es un texto estructurado que utiliza enlaces entre nodos que contienen hipertexto. Este protocolo fue estandarizado por el IETF (Internet Engineering Task Force) y el W3C (World Wide web Consortium), llegando a la versión final HTTP/1.1 en 1997 con el estándar RFC 2068. Actualmente, la versión 2 de HTTP se encuentra estandarizada e implementada por la mayoría de navegadores.

HTTP es un protocolo de la capa de aplicación utiliza normalmente el protocolo TCP para enviar las peticiones (GET, POST, PUT, DELETE, etc.) aunque es posible adaptarlo para que usa protocolos menos fiables como UDP

REST (REpresentational State Transfer) es una arquitectura web que usa como soporte el estándar HTTP. Con ella es posible crear servicios que pueden usarse en cualquier dispositivo que entienda el lenguaje HTTP. Por otra parte, al hacer una API REST, el estado nunca se guarda en el servidor, todas las instrucciones que se deben realizar han de venir adjuntas en la consulta que realiza el cliente.

Obviamente, hay ocasiones en las que es necesario guardar variables de estado como las sesiones.

La arquitectura REST se basa en aplicar restricciones a las interacciones que pueden realizarse en los distintos componentes. Cualquier aplicación que se atenga a estas restricciones se considerará RESTful y tendrá propiedades no funcionales como escalabilidad, portabilidad, simplicidad, etc. Las restricciones que se aplican a REST son las siguientes:

- Cliente-servidor: Debe existir una separación entre cliente y servidor, por ejemplo, los clientes no almacenan ningún tipo de dato, esto solo lo hace el servidor, lo que le da portabilidad en la parte del cliente. Los servidores por su parte, no son conscientes del estado en que se encuentra en cliente, lo que los hace simples y mejor escalables
- Sin estado: Un servidor no sabe en qué estado se encuentra un cliente, las peticiones del cliente contienen toda la información necesaria. En todo caso, el servidor guardará un registro de los clientes para su autenticación.
- Cacheable: Las respuestas del servidor deberán ser cacheables, es decir, que una respuesta que siempre sea igual podrá ser almacenada y reutilizada en el cliente sin necesidad de repetir la petición.
- Sistema por capas: El cliente no debería saber si se está conectando con el servidor final o con un intermediario. Los servidores intermedios permiten la escalabilidad.
- Código bajo demanda: No es algo necesario, pues no todas las páginas ejecutan código, pero sí que existe la opción de responder a una petición con código que aporte una funcionalidad en el lado del cliente.
- Interfaz uniforme: permite que la arquitectura sea más simple y modular, lo que permite a cada parte desarrollarse de manera independiente. Los requisitos para una interfaz uniforme son los siguientes:

- Uso correcto de las URIs: en una página web, la URL nos permite navegar entre páginas, secciones y archivos del sitio web. Cada una de estas secciones se denominan recursos en REST y será sobre ese recurso sobre el que queramos ejecutar una determinada acción. La URI permite identificar unívocamente el recurso deseado, la URL permite además conocer su ubicación. Una URI no debería denotar una acción sino un recurso sobre el que realizar una acción. Ejemplo: (casa/salón/temperatura) sería una URI correcta, mientras que (casa/salón/temperatura/obtener) no sería correcta, puesto que indicamos la acción a realizar en la misma.

  Por otro lado, tenemos las queries de acceso a datos o de filtrado, en ellas indicamos con ciertos parámetros que es lo que estamos buscando en concreto. Para ello nunca se pasarán los parámetros dentro de la URI, sino como parámetros HTTP. Por ejemplo:

  (casa/salón/temperatura/desde/2014/página/5)

  sería un uso incorrecto, mientras que:

  (casa/salón/temperatura?desde=2014&orden=descendente&página=2)

  sería un uso correcto del filtrado.

- Uso correcto del HTTP: Una vez definida la URI sobre la que se quiere hacer una operación, HTTP nos ofrece una serie de operaciones predefinidas que se pueden realizar sobre una URI dada. Estas son:
  - GET: Obtener un recurso.
  - POST: Crear un recurso.
  - PUT: Editar un recurso.
  - DELETE: Eliminar un recurso.
  - PATCH: Editar partes de un recurso
- Implementar Hipermedia: Cuando se hace una petición a una página web, por ejemplo, con un GET, ésta deberá enviar en la respuesta las URL de los recursos que aloja, para que así el usuario no tenga que buscarlos por otros medios.

MQTT.

Protocolo ligero, asíncrono y que permite una comunicación cliente/servidor por publicación/suscripción abierta y fácil de implementar. MQTT está pensado para su uso en dispositivos o localizaciones donde el código a implementar sea limitado o las capacidades de la red sean muy reducidas. Fue diseñado en 1999 por Andy Stanford-Clark y Arlen Nipper.

Se encuentra estandarizado como MQTT v3.1.1 por OASIS y es un estándar ISO/IEC con el identificador PRF 20922. Funciona normalmente sobre TCP, pero puede estar encapsulada en otros protocolos de red como pueden ser ZigBee. Es un protocolo ideal para la comunicación M2M por su bajo coste computacional y de red. Aunque su acrónimo solía ser Message Queue Telemetry Transport ya no hay ningún tipo de cola implementada en el protocolo, salvo en algún caso concreto.

La arquitectura que implementa MQTT de publicación/suscripción se basa en que un cliente emite mensajes en un tópico, no hacia un dispositivo concreto. Luego, los dispositivos que quieran consultar la información publicada tienen que suscribirse a ese tópico y automáticamente empezarán a recibir la

información. En la siguiente figura se representa este proceso con más detalle.

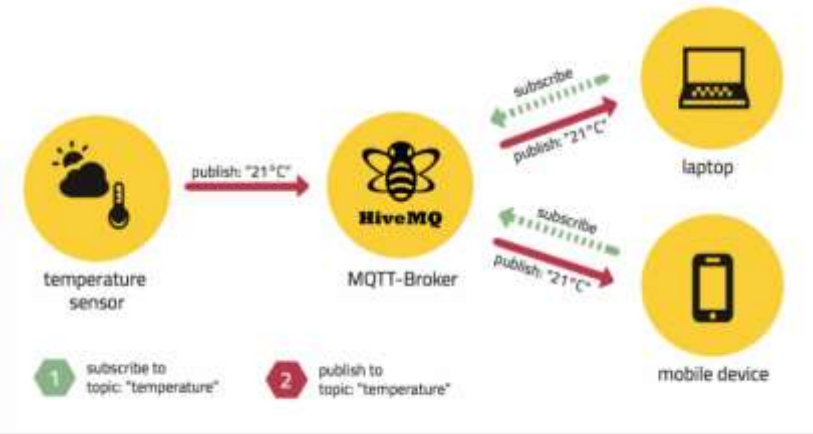

Figura 19: Arquitectura MQTT

CoAP.

Constrained Application Protocol es un protocolo pensado para su uso en dispositivos simples, que les permite comunicarse de forma interactiva mediante internet. Este protocolo se encuentra estandarizado como RFC 7252. Debido a su poca carga de mensajes es ideal para dispositivos con poca capacidad, limitaciones de batería o acceso limitado a internet. Está diseñado, como HTTP, para la transferencia de documentos, lo que lo hace perfecto para trabajar a través de internet. Una de sus diferencias con HTTP es que mapea los caracteres a enteros para conseguir una transmisión más ligera y además sin incurrir en costes computacionales extra. Al contrario que HTTP, CoAP no utiliza TCP sino UDP, implementando los reintentos de envío y demás comprobaciones en la capa de aplicación, de nuevo, incurriendo en muy bajo coste computacional.

También implementa soporte multicast. El soporte multicast se hace especialmente útil en IoT, pues permite comunicarse con muchos dispositivos de un tipo o cualidad idéntica, o a los que se quiera acceder o cambiar algún parámetro, al mismo tiempo.

CoAP se ha diseñado para tener las siguientes cualidades:

- Soporte para URIs y diferentes tipos de contenido.
- Descubrimiento de recursos ofrecidos por otros servidores CoAP de confianza.
- Suscripción simple a los recursos.

Al soportar HTTP, CoAP es capaz de utilizar los métodos REST. Los nodos exponen los recursos en direcciones URL accesibles a través de internet. También aporta una manera de descubrir los recursos de los nodos que se encuentran actualmente disponibles en tu red local.

Base de datos.

La base de datos que se ha utilizado para este proyecto es la base de datos Kafka, a continuación, se dará un breve resumen de sus capacidades y por qué se utiliza esta sobre el resto de bases de datos

distribuidas.

Kafka es una base de datos de código abierto diseñada por Apache y que tiene como objetivo el tratamiento de datos en tiempo real como una cola de mensajes. Utiliza el patrón publicador/subscriptor y es ampliamente escalable. Todas estas características la hacen una base de datos muy atractiva para aplicaciones empresariales.

Kafka recibe mensajes procedentes de múltiples fuentes y los organiza en lo que se denominan tópicos. Cada uno de estos mensajes se almacena en el tópico correspondiente, acompañado de una marca de tiempo en el que ha sido registrado. A estos tópicos se pueden suscribir clientes y aplicaciones y tomar datos en tiempo real, o por el contrario solicitar un rango de tiempo en el cual quieren obtener una serie de datos. Todo ello se realiza a través de las APIs que integra la base de datos:

- Producer API: permite publicar entradas en uno de los tópicos de Kafka.
- Consumer API: permite suscribirse a uno o más tópicos.
- Streams API: Permite enriquecer un flujo de datos de manera cómoda y aprovechando las cualidades de clustering de Kafka.
- Connector API: Permite establecer un flujo de datos entre Kafka y otras aplicaciones.

Kafka combina de manera equilibrada tanto la arquitectura publicador/subscriptor como el almacenamiento, integrándolo en una arquitectura altamente escalable y es, a día de hoy, la opción más extendida y utilizada a la hora de almacenar series temporales.

Proceso de captación de datos.

A la hora de tomar los datos de los dispositivos, estos siguen un cauce prefijado que permite almacenar los datos en una base de datos Cassandra y mostrar esos mismos datos a través de la plataforma de dashboards Grafana, toda esta parte no entra dentro del alcance de este documento. Sobre lo que trataremos es sobre el proceso que se sigue exclusivamente para comunicar los datos desde los dispositivos sensores hasta la base de datos principal (Kafka). Para ello se puede observar el esquema de arquitectura expuesto en la Figura 20.

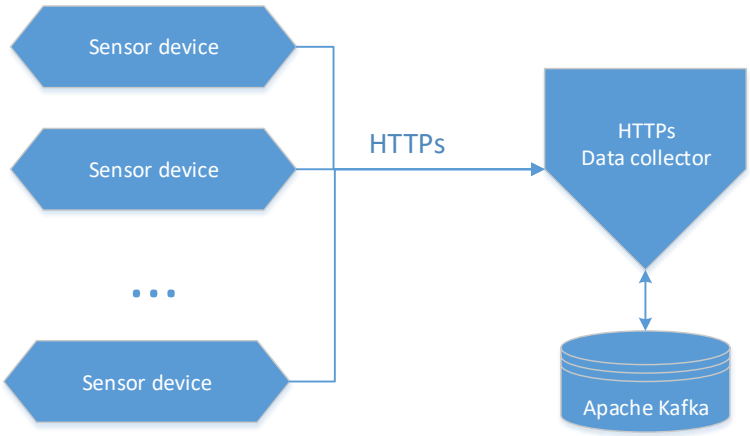

Figura 20: Sistema de mensajería.

En primer lugar, el dispositivo sensor se encuentra situado en un carro de limpieza, por lo que no siempre va a estar tomando medidas, ni siempre podrá enviarlas, por lo que tendrá que guardar un registro de medidas realizadas sin conexión que se enviarán en cuanto se establezca una conexión con la red. Esta comunicación se realizará mediante HTTPs con un servicio REST que recibe peticiones y las traduce al protocolo que entiende la base de datos Kafka.

Este servicio REST se encuentra implementado como un colector de datos, incluyendo funciones que permiten escribir datos en Kafka de manera sencilla. Este servicio no está ofrecido de manera nativa por Kafka, se ha realizado exclusivamente para este caso de uso.

A modo de conclusión de esta definición del mecanismo de mensajería, indicar que a lo largo de la tarea se han analizado los protocolos más utilizados dentro de IoT, cada uno con sus ventajas e inconvenientes. Aunque la base de datos Kafka pueda utilizarse de manera directa mediante las diferentes librerías disponibles para cada protocolo, se ha decidido elegir para el proyecto el protocolo HTTPs como método de comunicaciones por varias razones:

- Permite definir funciones y métodos de manera estándar (GET, POST…)
- Permite securizar las comunicaciones y añade otra capa más de autenticación tanto para leer como para escribir.
- Es estándar, escalable y flexible. Permite miles de peticiones por segundo tanto lectura como escritura. Por ello permitirá conectar todos los dispositivos que necesite el sistema de forma que se transmitan los datos entre dichos dispositivos y la BBDD Kafka. En las pruebas de laboratorio desarrolladas en CTIC, se conectan 12 dispositivos.

La arquitectura del sistema de mensajería , como se ve en la Figura 20, se encuentra compuesta por un conjunto de Things, un elemento intermedio que traduce de HTTPs al protocolo de Kafka y la propia base de datos que almacena las series de datos temporales que se van generando en los dispositivos sensores.

T2. Gestión de las comunicaciones.

En esta tarea se han implementado los protocolos de comunicaciones que conectan los dispositivos sensores, leen y guardan los datos a medir, los procesan y los envían a la base de datos global del sistema SAQ.

Cada dispositivo sensor o Thing está compuesto por varios sensores que miden cuatro variables del entorno: humedad, temperatura, CO_2 y ruido. Estos sensores se encuentran acoplados a una Raspberry Pi que se encarga de alimentarlos a 5V y leerlos periódicamente tal y como se muestra en la siguiente figura.

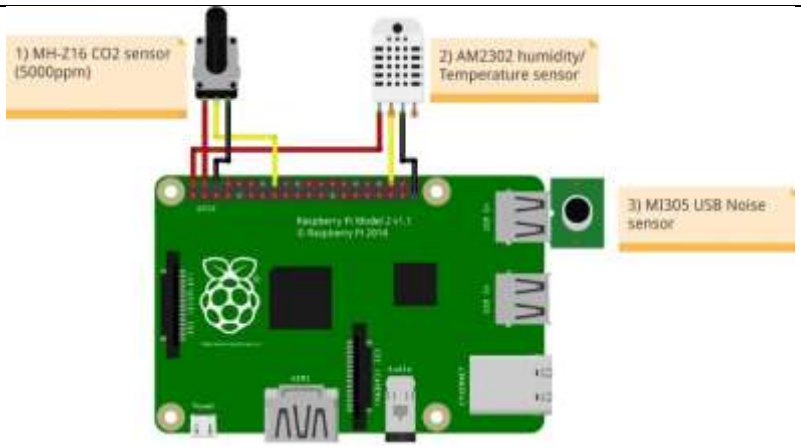

Figura 21: Esquema de montaje.

Cada uno de estos sensores se conecta a la Raspberry utilizando un protocolo de comunicaciones distinto, a saber:

- DHT22 (Humedad y temperatura): Se comunica a través de los pines GPIO de manera directa, generando un determinado voltaje que se traduce en datos de temperatura y humedad que pueden ser interpretados por un programa.
- MH-Z16 (CO_2): Este sensor se comunica mediante una interfaz PWM (Pulse Width Modulation), la lectura no es tan sencilla como en el anterior caso ya que es necesario traducir las señales que llegan al pin a señales en formato PWM ya que Raspberry no tiene este tipo de puertos. Los datos de CO2 que se obtienen se codifican en PWM de 0 a 5000ppm (partes por millón) en función del tiempo que esté la señal a 1 durante un segundo.
- MI305 (ruido): Este sensor se conecta mediante USB y es reconocido automáticamente por los drivers de audio de la Raspberry ya que es un micrófono común, lo que lo hace accesible de forma automática. Para obtener los datos es suficiente con acceder a las funcionalidades del driver que permiten obtener la potencia actual de sonido que se está midiendo.

Para recabar todos estos datos de manera que sean manejables y compartan una estructura básica, en la Raspberry se utiliza un software recolector. Este software se encarga de hacer un muestreado periódico de todos los sensores en base a sus descripciones y de almacenar localmente los datos en una base de datos SQLite local a la Raspberry. Para cada sensor el proceso es ligeramente distinto, pero el resultado se da en un formato predeterminado que se comparte entre todos los sensores. El formato de estos datos es el siguiente, representado en JSON:

```
{
 "name": "CO2",
 "value": 456,
 "timestamp": 1520853233.14948,
 "unit": "ppm"
}
```

La Raspberry Pi se conecta con el sistema a través de una comunicación inalámbrica. Una Raspberry Pi cuenta con dos protocolos de conexión con la red: WiFi y Ethernet 10/100. Debido a la naturaleza del

proyecto, se utilizará el protocolo WiFi, ya que permite conexión a internet en movimiento siempre que se permanezca en rango del punto de acceso. Con WiFi podemos también conectar múltiples Raspberry a una misma red sin preocuparnos por colisiones o ancho de banda, ya que muchas pueden compartir el medio (~100 dispositivos sin degradación del servicio).

Una vez establecida la conexión a internet podremos enviar los datos al servidor donde se encuentre el servicio de mensajería Kafka. Este puede estar en la misma red que las Raspberry o en un ordenador remoto a través de internet.

Existe la posibilidad de que no haya una conexión con la base de datos o que los datos no lleguen por algún fallo intermedio. En estos casos, los datos locales no se eliminan y se acumulan con los datos nuevos que van llegando de los sensores. En cuanto se reestablezca la comunicación con el servidor, todos estos mensajes se irán enviando en paquetes de un tamaño predeterminado para no saturar ni la Raspberry ni el servidor.

Una vez los datos se encuentran en la correspondiente cola de Kafka, han de ser movidos de nuevo, esta vez a una base de datos persistente (Cassandra). Para ello existe un servicio independiente escrito en Apache Spark que se encarga de leer las colas de Kafka y escribirlas en Cassandra utilizando el campo "uuid" como identificador. En la figura siguiente se muestra un extracto de la base de datos de Cassandra en el que se puede ver cómo se organizan las variables por UUID y timestamp.

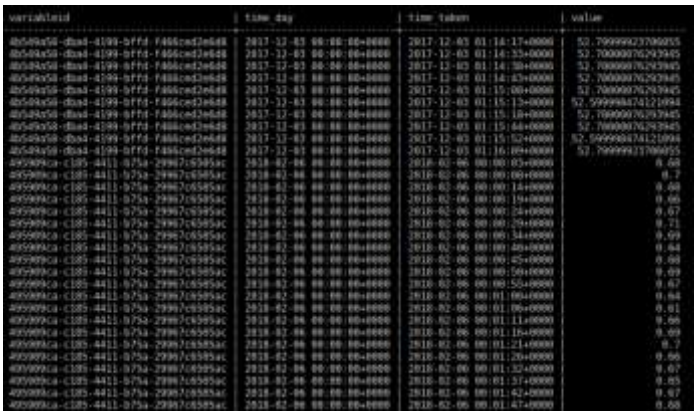

Figura 22: Consulta a la base de datos de Cassandra.

Banco de pruebas.

Para comprobar el correcto funcionamiento de todos los sistemas, se ha realizado un banco de pruebas utilizando diferentes dispositivos completos conectados a una red WiFi común tanto a ellos como al servidor donde se aloja el backend. Estas pruebas en entorno de laboratorio se han realizado distribuyendo un total de 15 motas en diferentes ubicaciones del edificio de CTIC. A continuación, se detallan las pruebas diseñadas junto a los resultados obtenidos:

Prueba	Resultado Esperado	Resultado obtenido
Conexión a la red WiFi	Todos los dispositivos se conectan a la red, tienen acceso a la misma y se ven entre si y a los servidores.	Conexión exitosa, todos los dispositivos son visibles.

Prueba sensor Temperatura/Humedad	Los datos de Temepratura y humedad llegan al servidor de Kafka así como a Cassandra.	Los datos llegan correctamente.
Prueba sensor CO2	Los datos de CO2 llegan al servidor de Kafka así como a Cassandra.	Los datos llegan correctamente.
Prueba sensor de ruido	Los datos del ruido llegan al servidor de Kafka así como a Cassandra.	Los datos llegan correctamente.
Prueba en movimiento	Mediante el uso de baterías, se enviarán datos de forma continua mientras la Raspberry circula por la planta de un edificio.	La Raspberry toma los datos correctamente y los envía en cuanto tiene conectividad WiFi.
Prueba de carga	Ambos dispositivos son capaces de enviar datos de manera prolongada y al mismo tiempo.	Los dispositivos han estado conectados y enviando datos durante más de una semana sin interrupción.

Tras la realización de las pruebas, se determina que el sistema funciona correctamente y que es capaz de manejar grandes volúmenes de datos. También se demuestra que las Raspberry son capaces de obtener, almacenar y enviar toda la información de los sensores, tanto en reposo como en movimiento.

Hito 5: Dashboard de visualización

T1. Panel de control.

En este proyecto se pretende realizar una medición continua de la calidad del aire, para ello se utilizarán factores claves que determinan esa calidad. Esos factores son, entre muchos otros, la humedad, la temperatura y el nivel de CO_2. Para ello se ha implementado un panel de visualización que representará los datos recabados de uno de los dispositivos.

Este panel de control se basa en la herramienta Grafana, ampliamente utilizada para la representación de series temporales en gráficas de múltiples tipos. Los datos mostrados se extraen en tiempo real de la base de datos en la que los dispositivos que leen los sensores guardan los datos recopilados. La herramienta Grafana se ocupa de recoger esos datos y representarlos en forma de gráfica. Estos datos son operables, es decir, pueden aplicárseles operaciones (medias, máximos, mínimos…), pueden establecerse límites, alertas, avisos por correo electrónico… Además, Grafana es altamente personalizable, pudiendo elegir no sólo qué datos mostrar, sino también el tipo de gráfica en el que se representa, filtrado de valores, indicadores numéricos, tablas de calor, tamaño y posición de las mismas…

A continuación, se expondrán algunas de las configuraciones utilizadas en este proyecto.

Configuración de una base de datos para Grafana.

La configuración de la base de datos a la que accede Grafana puede realizarse de varias maneras, pero la más sencilla es la que ofrece la propia interfaz, en la que se puede poner la URL y el puerto donde se encuentra alojada la base de datos.

Esta base de datos puede estar desplegado en el mismo servidor en el que se encuentra Grafana o estar en un servidor remoto. Para eso se tiene la opción "Access" que se puede configurar como proxy (remoto) o local. También se pueden introducir credenciales de acceso de diversos tipos a la base de datos.

En el piloto del proyecto, la base de datos está desplegada en el mismo servidor en el que se encuentra Grafana.

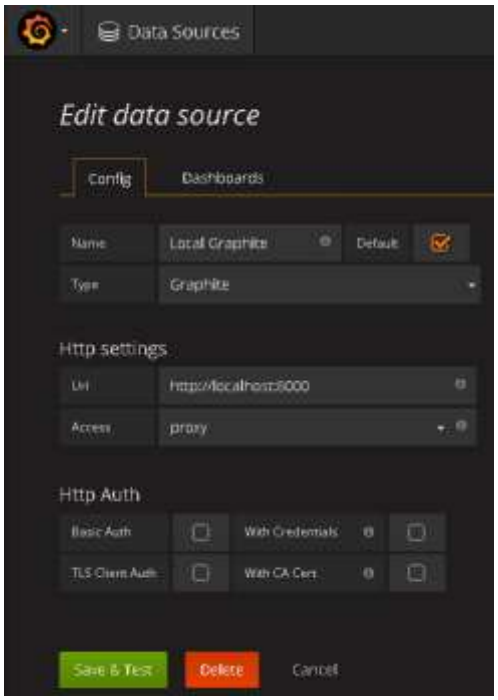

Figura 23: Configuración BD Grafana

Configuración de un dashboard en Grafana.

Un dashboard es el contenedor en el que se representarán los distintos datos y gráficas. Este dashboard se almacena en un fichero Json en el propio servidor donde se encuentra alojado Grafana y se lee cada vez que se carga la página.

Para el proyecto SmartAirQuality se han implementado 3 dashboards a través de los que poder visualizar los datos diarios, tanto datos medios como datos máximos, y los datos divididos por los diferentes pisos o plantas.

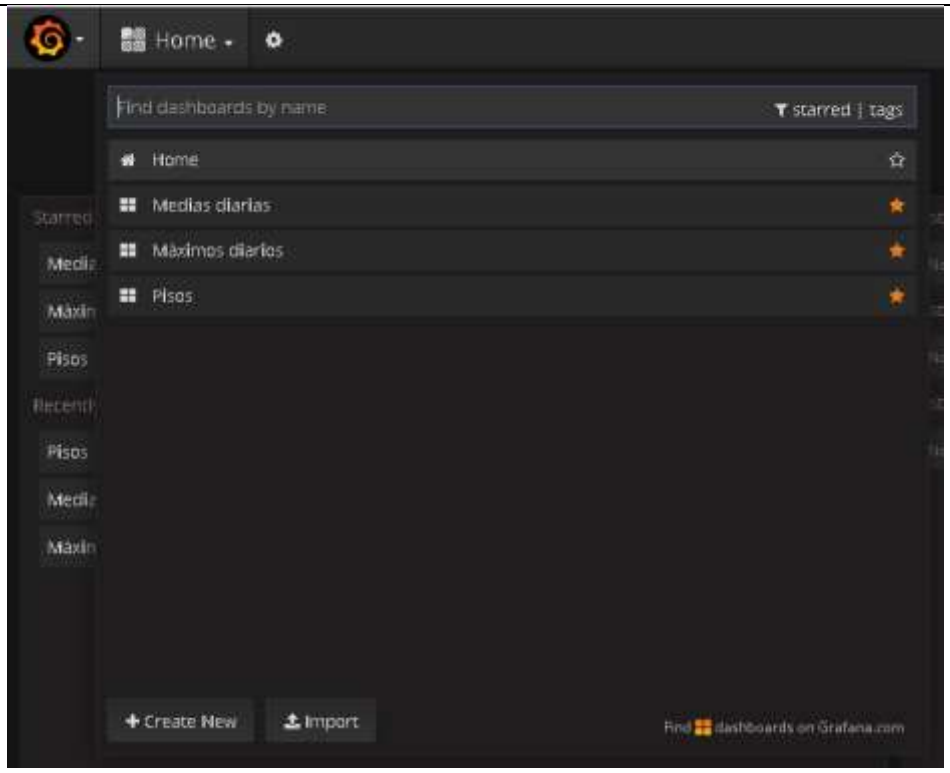

Figura 24: Creación de un dashboard

Configuración de una gráfica de Grafana.

En Grafana todos los elementos gráficos que se muestran en los dashboards se almacenan como ficheros Json en los que se indican todas las propiedades de cada elemento.

Configuración de un panel tipo manómetro.

Este panel muestra un dato en un momento dado junto a una representación visual del mismo en formato manómetro de aguja basada en colores.

A partir de estos componentes se ha implementado el panel de control que permite la monitorización de varios sensores en tiempo real. Desde este panel se puede visualizar los datos que se desee en el intervalo de tiempo configurado. Los indicadores en tiempo real situados a la derecha permiten tener una vista general de la situación actual, ya que implementan marcas en los límites recomendables de cada variable y cambian de color en función de si los valores recibidos están en esos límites o no.

En la siguiente figura se puede ver el panel de control implementado para la mota 15, en el que se ven a la derecha los valores en tiempo real de las variables Temperatura, Humedad, Presión, CO_2, NO, SO_2 y polvo, y a la izquierda la serie temporal de cada variable con la variación de dicho valor a lo largo de los 10 días anteriores.


Temperatura
25ºC
Humedad
55%
Presión
1020bar
CO2
492ppm

Figura 25: Panel de control con la visualización de los datos recogidos por la mota 15.

Según se muestra en la figura anterior, a través de este panel de control se miden y visualizan en tiempo real los datos de 7 sensores por cada dispositivo, lo que da sobrado cumplimiento a los indicadores de gestión y/o avance de la tarea.

T2. Análisis inteligente de datos.

Dados los datos recogidos a través de las metodologías determinadas a lo largo de este proyecto, se han enfocado y ejecutado tres procesos de análisis de datos, a través de diversas técnicas de inteligencia artificial enfocadas a diversos problemas. Estos procesos son:

- Predicción del comportamiento en series temporales.
- Análisis de calidad.
- Análisis de datos.

A continuación, se detallan los trabajos realizados para cada uno de los procesos de análisis de datos. Cabe destacar que se ha realizado un trabajo de pre-procesado común a los tres, dado que cualquier proceso de análisis de datos requiere una exhaustiva etapa de pre-procesado que permita analizar dichos datos, preparándolos adecuadamente para el posterior uso de los algoritmos considerados. Nótese la especial importancia de dicha fase inicial, ya que todo desarrollo posterior está basado en dichos datos e, inherentemente, en la calidad de éstos.

La principal tarea del pre-procesado consiste en el filtrado de los datos. Dichos datos pueden presentar entradas erróneas por diversos motivos (fallo en la captura del dato, fallo en la comunicación del dato,

fallo en el almacenamiento, etc.), los cuales pueden empeorar los resultados obtenidos a partir de los algoritmos generados.

1.- Predicción del comportamiento en series temporales.

En la *Figura* 26 se muestra de forma resumida el proceso llevado a cabo para dicho análisis.

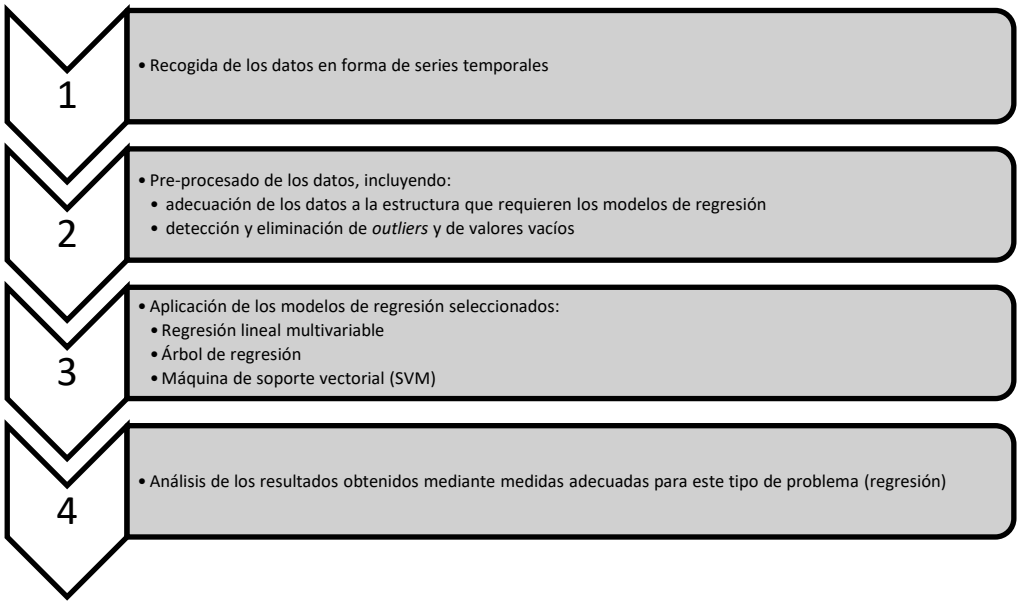

Figura 26. Esquema del procesamiento correspondiente a la predicción del comportamiento en series temporales

Dados los pasos mostrados previamente, ha de prestarse especial atención al segundo punto: el pre-procesado de los datos. En este caso, se precisa transformar los datos de modo que, en el momento del entrenamiento del modelo, este disponga de los datos de los instantes previos para la predicción del instante en estudio.

Para ello, ha sido necesario determinar el tamaño de la ventana temporal a utilizar, habiendo sido 20 los valores previos analizados. Así, para cada uno de los individuos, se analizarán los 20 valores anteriores en dicha serie temporal para generar la predicción del momento actual.

Dicho tamaño ha de ser controlado, ya que ampliarlo podría generar el resultado opuesto al esperado: introducir ruido al modelo, además de ralentizarlo.

La naturaleza de este problema permite modelarlo con técnicas de regresión, donde a partir de las variables independientes (variables conocidas) se estimará la variable dependiente (variable a predecir).

a) Regresión lineal.

Se trata de una técnica estadística ampliamente conocida que permite estudiar la relación entre variables, siendo adaptable a diversos casos de uso, como el que aquí se presenta. Existen dos enfoques, donde se dispone de dos variables en estudio (regresión simple), y donde existe un número mayor de variables (regresión multivariable). En ambas, el objetivo es analizar la relación entre la variable dependiente y una o más variables independientes o predictoras, respectivamente.

Sean y la variable dependiente, $x_1, \dots, x_n$ las variables independientes, $\beta_0, \dots, \beta_n$ los parámetros que definen la regresión y ε el término aleatorio, donde $n = 1$ para la regresión simple y $n > 1$ para la regresión multivariable. La recta y el hiperplano de éstos vienen definidos respectivamente mediante $y = \beta_0 + \beta_1 x_1 + \varepsilon$ y $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n + \varepsilon$.

b) Máquinas de soporte vectorial (SVM) [regresión].

Una máquina de soporte vectorial (Support Vector Machine, SVM) es un conjunto de algoritmos de machine learning supervisado aplicable tanto a problemas de clasificación como de regresión, como es el caso aquí estudiado.

La aplicación básica de una SVM radica en un problema de clasificación binaria, donde mediante un hiperplano de separación, se determinan dos espacios caracterizando a cada una de las clases. Dicho hiperplano se definirá de modo que exista la mayor separación posible entre ambas clases en función de distintas características. Es esta separación lo que define a este tipo de algoritmos, buscando el hiperplano con distancia máxima con los puntos más cercanos a éste. Los nuevos datos serán clasificados a partir de dicha separación.

Si bien las SVM son utilizados con mayor frecuencia para problemas de clasificación, existe una versión aplicable a problemas de regresión. En este caso, la idea de desarrollo está basada en la realización de un mapeo de los datos disponibles para entrenamiento $x \in X$ a un espacio de mayor dimensionalidad F de forma no lineal mediante $\varphi: X \to F$, pudiendo así realizarse una regresión lineal.

c) Árboles de regresión.

Un árbol de regresión permite predecir los valores de la variable objetivo a partir de diversas variables de entrada, mediante construcciones lógicas con cierta similitud a los sistemas basados en reglas. Existen diversos tipos de árboles de regresión, como son el M5P, el M5Rules, o los Random Forest.

Dichos árboles constan de nodos internos, nodos hoja y ramas. Los nodos internos contienen los tests a aplicar sobre alguna de las variables analizadas. Los nodos hoja son aquellos nodos finales del árbol, donde se determinará el valor de salida obtenido por parte del árbol. Por último, las ramas comunican los nodos entre sí, permitiendo continuar la serie lógica planteada. Ha de tenerse en cuenta la profundidad del árbol a la hora de su diseño, ya que una excesiva profundidad podrá deberse a un mayor overfitting a los datos de entrenamiento, llevando dicha situación a peores resultados finales.

Conclusión del análisis de predicción del comportamiento en series temporales:

La comparación de los resultados obtenidos se ha realizado a través del uso de la correlación entre los resultados generados mediante las predicciones y el valor real conocido, así como dos métricas ampliamente conocidas para el análisis de los resultados de este tipo de problemas de regresión.

Los resultados se muestran en la Tabla 1, donde se incluyen los valores para la correlación y dichas dos medidas con respecto a los tres modelos estudiados.

Tabla 1. Resultados obtenidos para la predicción del comportamiento de series temporales

Modelo de regresión	Correlación	MAE	MAPE
Regresión lineal multivariante	0.9998	0.014	0.0007
Árbol de regresión	0.9994	0.055	0.0028
Máquina de soporte vectorial	0.9999	0.012	0.0006

Los resultados obtenidos por los tres modelos apenas producen errores notables mediante cualquiera de las tres medidas consideradas. Sin embargo, uno de los modelos, la regresión multivariable, presenta una ventaja importante que es el hecho de producir modelos más sencillos a los propuestos por los otros dos modelos. Así, se ha seleccionado la regresión lineal multivariante sobre los otros dos modelos, ya que, ante unos resultados similares, proporciona una gran ventaja: la interpretabilidad del modelo obtenido.

2.- Análisis de calidad.

El análisis de la calidad del aire se basa en la información recogida acerca de aquellas variables con influencia sobre el estado en el que se encuentra la atmósfera del lugar. Dicha información muestra gran dependencia temporal, aunque a diferencia del caso anterior, la variable objetivo estará definida por distintas categorías asociadas al nivel de calidad.

Para conocer la calidad del aire, se ha requerido que, de forma periódica, por parte de distintos operarios se proporcione la calidad del aire percibida en ese momento mediante una escala de cinco valores:

- 1 – Muy mala
- 2 – Mala
- 3 – Normal
- 4 – Buena
- 5 – Muy buena

Será dicha información la utilizada como variable objetivo, mientras que las distintas variables recogidas automáticamente por los sensores serán las tratadas como variables independientes.

De modo análogo al enfoque anterior, se ha generado un esquema en la *Figura* 27 estructurando los pasos desarrollados en este apartado.

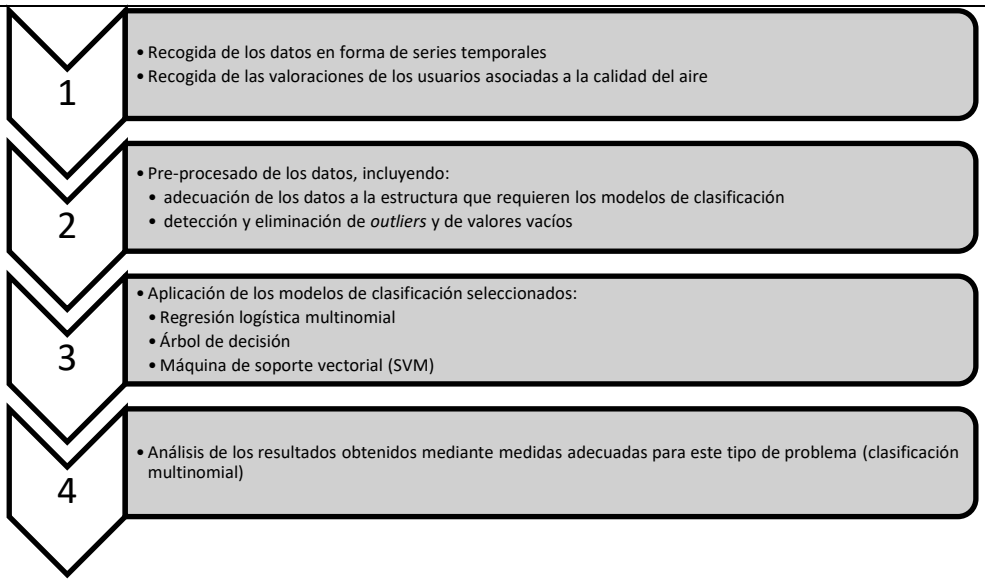

Figura 27. Esquema del procesamiento correspondiente al análisis de la calidad del aire

Al igual que en el anterior apartado, el paso de pre-procesado ha sido indispensable para el correcto funcionamiento del sistema. En este caso, los operarios han valorado cada hora la calidad percibida del aire en ese momento. Además, se han obtenido los datos de dicho periodo para las tres variables estudiadas (CO_2, temperatura y humedad). Dado que dichos datos han sido recogidos de forma continua con un tiempo entre recogidas de apenas unos segundos, se han agregado dichos datos por periodos de 10 minutos. Así, cada individuo proporcionará información de una hora, donde dispondremos de seis valores por cada una de las tres variables estudiadas, así como el valor dado por los operarios para la calidad del aire.

Para contrastar de una forma más veraz los resultados se han simulado 100 particiones train-test diferentes fijando diferentes semillas aleatorias, de modo que los resultados sean más conclusivos en cuanto a la calidad de cada algoritmo.

A partir de estos datos, se han aplicado los siguientes modelos de clasificación multinomial.

a) Regresión logística multinomial.

La regresión logística multinomial se trata de una adaptación del caso de regresión presentado en el apartado anterior, pero enfocada al caso de clasificación no binaria, es decir, donde la variable dependiente es categórica con más de dos clases posibles. Se trata de un modelo lineal generalizado donde la componente aleatoria asume que la distribución de la variable dependiente es multinomial, determinada por un vector de probabilidades de éxito para cada categoría.

b) Árboles de decisión.

Este tipo de modelo matemático es análogo al caso de los árboles de regresión detallado en el caso previo. Del mismo modo, está basado en construcciones lógicas, y su estructura está basada en nodos (internos, hoja) y ramas. Las características de todos ellos son las mismas, si bien existe la diferenciación en los nodos hoja, es decir, los nodos donde se devuelve el resultado obtenido por el modelo. En este caso, los valores posibles serán las diferentes clases que componen el problema de clasificación en

estudio

c) Máquinas de soporte vectorial (SVM) [One-vs-All].

El último método seleccionado para este problema es una de las distintas versiones de las máquinas de soporte vectorial (SVM) descritas en el primer apartado. En concreto, nos hemos centrado en su versión de clasificación multiclase.

Como se ha detallado en el apartado correspondiente, las SVM han sido diseñadas inicialmente para la clasificación binaria mediante la definición de un hiperplano que separe dos espacios característicos de cada clase. Para nuestro caso, se ha considerado la variante One-vs-All, cuyo funcionamiento se basa en la clasificación de cada una de las clases (One) frente al resto de clases (All).

Conclusión del análisis de calidad:

A lo largo de este análisis de calidad, nos encontramos ante problemas de clasificación. Por ello, ha sido necesario utilizar medidas asociadas a dicho tipo de problema para el análisis de la calidad de los resultados.

Los resultados obtenidos para esta problemática en media de las 100 simulaciones realizadas, y con respecto a cada una de las métricas consideradas son presentados en la Tabla 2.

Tabla 2. Resultados obtenidos para la predicción del análisis de calidad

Modelo de regresión	*Accuracy*	*Precision*	*Recall*
Regresión logística multinomial	0.293	0.205	0.218
Árbol de decisión	0.373	0.203	0.372
Máquina de soporte vectorial	0.387	0.224	0.468

Como se puede apreciar, los resultados del árbol de decisión y la máquina de vector soporte son muy similares, por lo que ambos métodos podrían ser aplicables. Sin embargo, los árboles de decisión son aquellos que mejor se adaptan a la tipología del problema, gracias a la determinación de la clase final en base a la concatenación de reglas sencillas acerca de las diferentes variables tenidas en consideración para el análisis. Nótese también la interpretabilidad que proporciona este tipo de método frente a otros considerados como cajas negras.

3.- Análisis de fallos

Este tercer enfoque consiste en la detección de fallos a lo largo del sistema de recogida, de modo que se pueda localizar en qué instante temporal ha habido algún error de procesamiento. Dicha detección permitirá poder identificar si alguno de los sensores está provocando un elevado número de errores, de modo que pueda ser reparado o sustituido si así se requiere.

Este problema consiste en la detección de existencia o no existencia de un error en el instante en estudio. Así, nos encontramos ante un problema de clasificación binaria, tal y como se indica a lo largo del informe técnico final.

De modo que sea posible el entrenamiento de un modelo, ha sido necesario etiquetar de forma manual

aquellas ocasiones que han sido detectadas por un experto como casos erróneos en la recogida de los datos.

Al igual que en los anteriores casos, se proporciona un esquema del procesamiento realizado para este enfoque en la *Figura* 28.

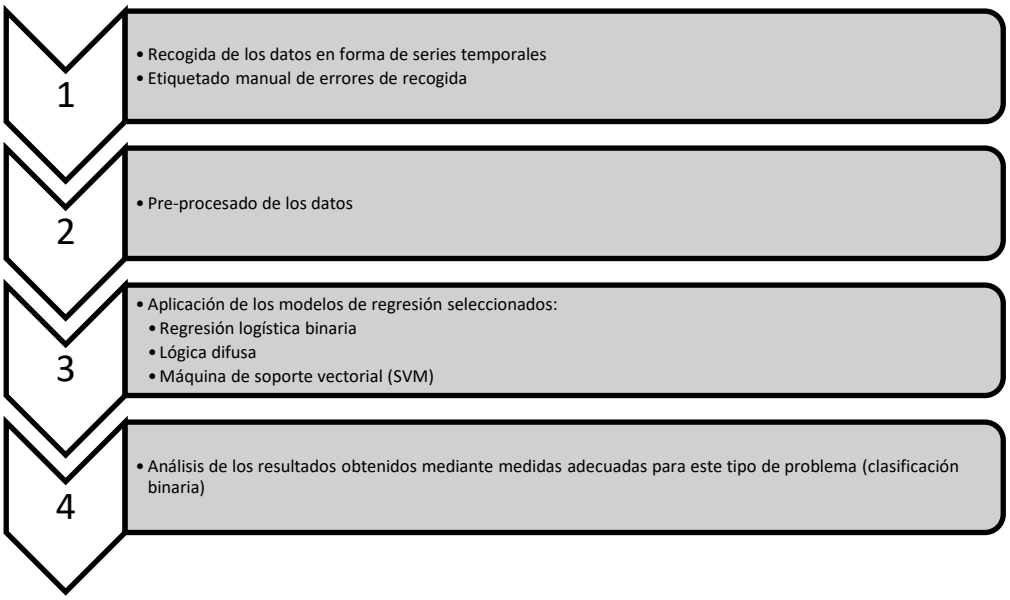

Figura 28. Esquema del procesamiento correspondiente al análisis de errores

En este caso, al encontrarnos ante un problema de clasificación, hemos recurrido a las mismas medidas que en el segundo enfoque para la determinación de la bondad de los resultados. De todos modos, su definición ha de ser adaptada a clasificación binaria.

Además, hemos procedido a simular 10 particiones diferentes fijando distintas semillas aleatorias, del mismo modo que en el enfoque previo.

En este caso, se ha buscado determinar la existencia de fallos a lo largo del proceso a partir de los datos disponibles tras su recogida. Se ha considerado la posibilidad de existir fallo o no, es decir, el problema aquí considerado está caracterizado como una clasificación binaria.

a) Regresión logística binaria.

Al igual que la regresión logística multinomial, nos encontramos ante una adaptación de la regresión mostrada en el primer caso de estudio, donde la variable objetivo pasa a ser categórica, si bien tomando únicamente dos valores (fallo, no fallo).

Del mismo modo que la regresión logística multinomial, se trata de un modelo lineal generalizado con las mismas características determinadas en la descripción de éste, pero donde el vector de probabilidades está asociado exclusivamente a dos categorías.

b) Máquinas de soporte vectorial (SVM) [binaria].

Como se indica en el primer caso de estudio en el apartado correspondiente a las máquinas de soporte vectorial, la definición original de este tipo de modelo está diseñada para modelos de clasificador binaria, donde un hiperplano genera dos espacios, tal que cada uno de ellos caracteriza cada una de las categorías disponibles. Como se indica en dicho apartado, tal hiperplano será definido de tal forma que se maximice la distancia de éste a los distintos individuos de cada clase en el entrenamiento.

c) Lógica difusa.

La lógica difusa permite manejar la posible imprecisión asociada a los datos recogidos, de modo que se puedan paliar sus efectos en los resultados. De este modo, se podrá definir un método de clasificación con un enfoque diferente al mostrado con las dos técnicas previas. En concreto, la lógica difusa se caracteriza por definir conjuntos difusos donde un elemento no ha de pertenecer a éste o no, de forma categórica, sino que puede pertenecer a un conjunto con un determinado grado de pertenencia, habitualmente tomando valores entre 0 y 1, donde 0 representa la no pertenencia al conjunto, y 1 la pertenencia total a éste.

Conclusión del análisis de fallos:

Tras las simulaciones realizadas, se han obtenido los resultados que se muestran en la Tabla 3.

No se proporcionan los resultados de un modelo de lógica difusa, a pesar de indicarse su uso a lo largo del análisis de fallos, ya que las características finales de los datos utilizados no resultan ser las adecuadas para dicha metodología, llevándonos a su descarte.

Tabla 3. Resultados obtenidos para la predicción del análisis de fallos

Modelo de regresión	*Accuracy*	*Precision*	*Recall*
Regresión logística binaria	0.939	0.327	0.004
Máquina de soporte vectorial	0.998	0.996	0.981

Como se observa en los resultados proporcionados, si bien la primera de las métricas es muy alta en ambas, no ocurre lo mismo con las otras dos, siendo claramente el segundo de los métodos, la máquina de soporte vectorial, la que mejores resultados ha proporcionado. Dado que existe una proporción muy grande de no errores frente a errores de recogida, el primero de los métodos devuelve en la gran mayoría de las ocasiones el mismo valor (0), de ahí que los resultados sean buenos en el primer valor, mientras que los otros dos, sensibles a los falsos positivos y falsos negativos, den resultados muy deficientes.

Por tanto, en este caso de estudio, los resultados han mostrado una mejor actuación por parte de las máquinas de soporte vectorial. Al encontrarnos ante un problema de clasificación binaria, las SVM muestran una mayor robustez que el resto de metodologías a la hora de determinar la clase de nuevos individuos, siempre y cuando se eviten situaciones de overfitting.

T3. Componentes de visualización avanzada.

Durante la tarea T3, el sistema de visualización desarrollado en la tarea T1 se ha completado con un segundo panel de control con componentes de visualización avanzados, donde se combinan los datos adquiridos por varios sensores en un solo panel.

La razón de ser de estos componentes de visualización avanzada, viene motivada por la existencia de varios dispositivos cada uno con múltiples sensores, lo que incrementa los requerimientos del sistema haciendo que sea necesario no solo visualizar datos, sino tener también la capacidad de agregarlos y compararlos mediante un panel de control más avanzado.

Para ello se ha diseñado una interfaz, basada también en Grafana, en la que se agrupan los datos de múltiples dispositivos, permitiendo verlos de manera individual, colectiva, acumulada, por días…

En este proceso se han utilizado las herramientas que dispone Grafana para el tratamiento de datos, como por ejemplo la herramienta "media" que permite calcular la media para un valor en un intervalo de tiempo concreto, obteniendo así las medias diarias de todas las variables, o como la función de agregación, utilizada para poder contrastar varios sensores a la vez en un mismo gráfico, donde se puede seleccionar qué dispositivos mostrar y cuáles no.

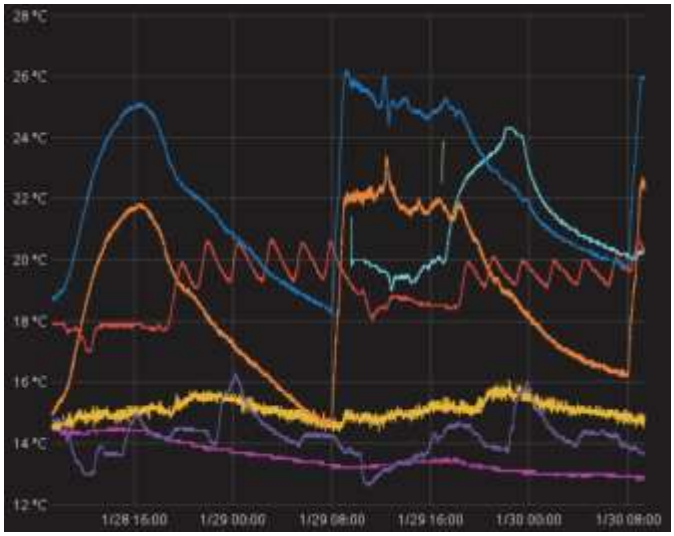

Figura 29: Ejemplo de función de agregación donde se muestran de forma simultanea los datos de temperatura tomados por 6 motas diferentes.

Esta agregación se realiza mediante la implementación de una métrica como se muestra en la figura siguiente.

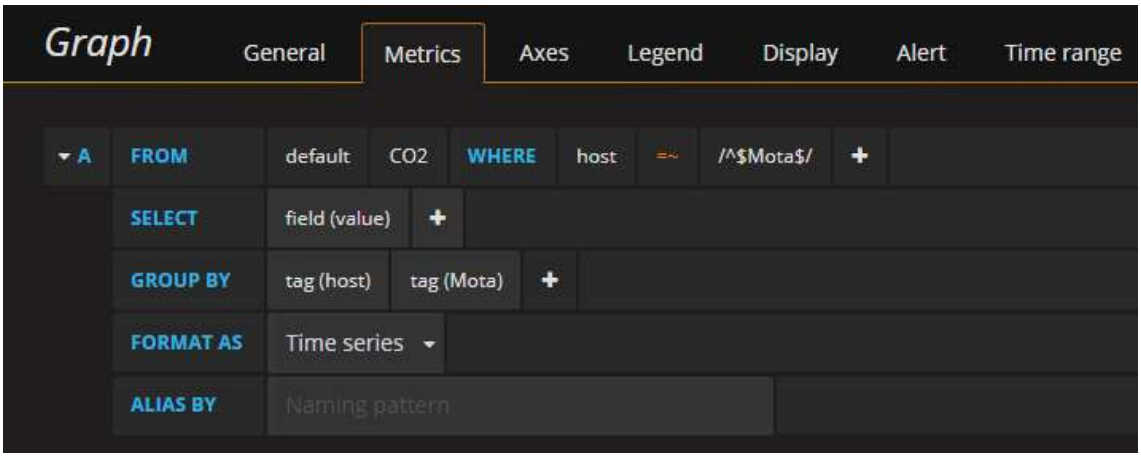

Figura 30: Ejemplo de métrica de Agregación: Agregación de valores CO_2 agrupados por motas.

Una vez aplicada, Grafana se encarga automáticamente de realizar las consultas a la base de datos, obtener los valores almacenados, pintar y etiquetar cada uno con un color distintivo en la gráfica.

Figura 31: Panel de visualización avanzado. Datos agrupados (48 últimas horas).

Por otro lado, Grafana nos permite agrupar los datos en intervalos horarios, haciendo la media, mediana, moda, máximo… de los mismos. Para ello, se indicará un flag en el campo "GROUP BY" que se muestra en la Figura 30. denominado time(). A este flag se le pasa como parámetro el intervalo de tiempo deseado, en el caso de la Figura 31 serán 24 horas. Así, Grafana se encargará de recopilar los datos y aplicarles el filtro para mostrar los valores diarios deseados en la gráfica.

Esto permite obtener diferentes valores, como por ejemplo la media diaria de varias variables durante los últimos 90 días, tal y como muestra la siguiente figura.

Figura 32: Medias diarias (90 días)

A modo de conclusión de este panel de control, podemos afirmar que el proyecto SmartAirQuality ha implementado dos paneles de control, uno sencillo en el que se muestran las variables individuales en tiempo real, y otro avanzado en el que se muestran de forma agregada los datos de múltiples variables, lo que permite tener diferentes opciones de agregación y comparación de valores.

Según se ha visto, con una parametrización sencilla de Grafana a través de la creación de diferentes métricas, se puede obtener un número de componentes de visualización para cada panel de control muy elevado, lo que sin duda da cumplimiento a los indicadores de seguimiento y control establecidos para este Dashboard de visualización.

2. RESULTADOS PREVISTOS Y RESULTADOS CONSEGUIDOS

2016

Durante el periodo de tiempo que cubre la primera parte de este informe se han conseguido todos los resultados planificados:

- Se ha realizado un estudio de la normativa de calidad del aire en hospitales.
- Se han recolectado los requisitos funcionales y no funcionales necesarios para este sistema y sus especificaciones técnicas.
- Se ha hecho el modelo conceptual del sistema.
- Elección e implementación de los elementos sensores en el prototipo de monitorización IoT.
- Diseño de la envolvente del sistema de adquisición y estudio de su ubicación.
- Se ha realizado el estudio del estado del arte en Web de las cosas.
- Diseño de la arquitectura Wot y la adaptabilidad a la sensórica SMARTAIRQUALITY al 60%.
- Implementación del sistema se ha empezado al 25%

Durante el siguiente periodo de ejecución del proyecto se seguirá trabajando en la implementación y diseño final de la arquitectura y adaptabilidad a la sensórica.

Los hitos finales comprenderán la recepción de datos y gestión de comunicaciones por una parte y el panel de control, análisis inteligente de datos y componentes de visualización avanzada.

2017

Durante el periodo de tiempo que cubre la segunda parte de este informe se han conseguido todos los resultados planificados:

- Se ha completado y finalizado el diseño de la arquitectura Wot y la adaptabilidad a la sensórica SMARTAIRQUALITY.
- Implementación del sistema que se había empezado el año anterior.
- Diseño del módulo de recepción de datos.
- Realización de la comunicación de los datos entre sensores y base de datos.
- Implementación del panel de control.
- Sistema de análisis de datos inteligente.
- Creación de componentes para permitir la visualización avanzada.

Con el fin de las tareas anteriormente descritas se da por completado el proyecto exitosamente.