ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Integración en la Plataforma SmartSantander de un Sensor PM2.5 basado en MQTT

(Integration of a PM2.5 MQTT Sensor into the SmartSantander Platform)

Para acceder al Título de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Ian Granado Rodríguez

Octubre 2018

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Ian Granado Rodríguez
Director del TFG: Luis Sánchez González
Título: "Integración en la Plataforma SmartSantander de un Sensor PM2.5 basado en MQTT"
Title: "Integration of a PM2.5 MQTT Sensor into the SmartSantander Platform"

Presentado a examen el día: 26 de octubre de 2018

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Elena Mediavilla Bolado
Secretario (Apellidos, Nombre): Jorge Lanza Calderón
Vocal (Apellidos, Nombre): Luis Sánchez González

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado Nº
(a asignar por Secretaría)

Índice

Indice de Figuras

Indice de Tablas

Capítulo 1: Introducción

1.1. MOTIVACIÓN DEL PROYECTO

Vivimos en un mundo cada vez más globalizado e interconectado, en el que, hoy en día, más de la mitad de la población mundial reside en áreas urbanas, en comparación con el 30% que lo hacía en 1950. Según la ONU [1], las próximas décadas traerán consigo más cambios, sobre todo, en lo que al tamaño y distribución de la población se refiere. Las previsiones auguran que en 2050 un 66 % de las personas viva en ciudades. Esto dará lugar a un incremento de 2.500 millones de habitantes. De entre los múltiples retos que esta tendencia implica, uno de los más importantes es la degradación medioambiental que esta presión demográfica supondrá, en especial en lo relativo a la calidad del aire en las ciudades.

Actualmente, la contaminación atmosférica está cobrando importancia y protagonismo debido principalmente al intenso e incesante incremento del tráfico de

vehículos que soportan nuestras urbes. Concretamente, las partículas micrométricas, presentes en la polución generada por los vehículos a motor, son altamente perjudiciales para la salud de la población en general y especialmente para aquellas personas aquejadas de alguna afección respiratoria.

Según datos de la Organización Mundial de la Salud (OMS) 9 de cada 10 personas respiran aire con altos niveles de contaminación, lo que lo convierte en un gran problema de salud a nivel mundial [2]. Asimismo, las estimaciones de la OMS muestran que 7 millones de personas mueren cada año a causa de la contaminación.

Así pues, al encontrarnos ante estos escenarios de masificación y contaminación resulta necesario plantear estrategias de monitorización del aire que faciliten, en primer lugar, un mejor conocimiento de su calidad para a continuación, plantear protocolos que permitan la reducción de los niveles de polución existentes. Cuanto más refinado sea ese conocimiento más precisas podrán ser las actuaciones puestas en marcha para mantener los niveles de polución por debajo de los límites perjudiciales para la población. Del mismo modo, esto permitirá la mejor toma de decisiones por aquellas personas que pudieran tener una especial sensibilidad a la contaminación.

Con el objetivo de alcanzar una mayor sostenibilidad medioambiental, está surgiendo un nuevo modelo de ciudades denominadas ciudades inteligentes o “Smart Cities”. Estas ciudades se apoyan en el uso de las Tecnologías de la Información y las Comunicaciones (TIC) para lograr la eficiencia en la utilización de los recursos disponibles.

Un aspecto, que hoy se considera fundamental, de dichas ciudades consiste en poder detectar las concentraciones de partículas micrométricas en el aire de una manera sistemática. Esto les permite poner en marcha mecanismos que engloban desde avisar a los ciudadanos hasta la restricción o el reordenamiento del tráfico, con el fin de reducir y/o mantener los niveles de partículas adecuados y así evitar daños tanto medioambientales, como para la salud de los ciudadanos.

Siendo conscientes de los retos globales a los que hemos hecho referencia, queremos formar parte de este esfuerzo internacional que se está llevando a cabo de cara a la sostenibilidad y preservación del planeta. Además, consideramos relevante poder contribuir a la mejora de la salud de la población a través de la monitorización de la calidad del aire. Para ello, este Trabajo de Fin de Grado (TFG) se ha propuesto el

objetivo de implementar un sensor de partículas PM2.5 e integrar las observaciones generadas en la plataforma para el Internet de las Cosas (IoT) SmartSantander.

1.2. OBJETIVOS DEL PROYECTO

A partir del objetivo general descrito en la Sección anterior, se han definido una serie de objetivos concretos con el doble objetivo de estructurar las tareas a llevar a cabo durante la realización de este TFG y de permitir un seguimiento durante la realización de las mismas.

De este modo se definieron los siguientes objetivos concretos:

(O1) Integrar la plataforma hardware que soporta al sensor de partículas micrométricas.

(O2) Adaptar e instanciar el firmware del sensor en la plataforma hardware

(O3) Desplegar el servidor MQTT para la gestión a nivel local de las observaciones generadas por el sensor desarrollado.

(O4) Desarrollar un sistema de visualización de los datos capturados por el sensor compuesto por un back-end de almacenamiento y un dashboard para visualización en tiempo real.

(O5) Analizar las plataformas LASS y SmartSantander por ser los sistemas en los que está basado y con los que interactuará, respectivamente, el sistema de monitorización implementado en este TFG.

(O6) Integrar los datos generados por los sensores micrométricos en la plataforma SmartSantander.

1.3. ESTRUCTURA DEL PROYECTO

En este primer Capítulo se han presentado brevemente los aspectos generales y las principales motivaciones en torno a los cuales se formula este trabajo. Igualmente, se han descrito los objetivos concretos con los que se pretende avanzar, mediante los desarrollos realizados en este TFG, hacia la visión de una monitorización fina de las partículas micrométricas en el aire de las ciudades.

En el Capítulo 2 se hace un repaso no exhaustivo al marco tecnológico de este TFG. En este sentido se resume el estado del arte en lo que respecta a la monitorización ambiental en las ciudades inteligentes, la plataforma para la IoT de SmartSantander, el

protocolo MQTT y los programas con los cuales se monitorizarán los datos recibidos del sensor.

La creación e implementación del sensor que se ha desarrollado en este TFG se describe en el Capítulo 3. Además, se presenta el procedimiento de instalación del sistema de monitorización y la interfaz virtual.

En el Capítulo 4 se detalla el proceso a través del cual se integra el sensor PM2.5 que se ha fabricado en la plataforma SmartSantander.

Por último, en el Capítulo 5 se presentan los principales resultados y las conclusiones que se extraen de la realización de este TFG y se tratan algunas de las limitaciones con las que nos hemos encontrado durante la realización del proyecto. Además, se hará referencia a aquellos aspectos relacionados con el tema que nos ocupa y que, a nuestro juicio, deben ser objeto de una mayor investigación en el futuro.

Capítulo 2: Marco de Desarrollo

2.1. MONITORIZACION AMBIENTAL EN LA SMART CITY

El uso de tecnología inteligente en las Smart Cities está motivado por el ahorro tanto de tiempo, como de recursos [3]. Para ello, se recurre a la monitorización, que consiste en una práctica basada en sensores y que resulta fundamental a la hora de articular la sostenibilidad y la eficiencia de dichas ciudades inteligentes. Así pues, uno de los tipos de monitorización con el que nos podemos encontrar en una Smart city, es la llamada "medioambiental" cuyo objetivo pasa por controlar desde la calidad del aire y del agua entre otros.

Dicha monitorización en las ciudades inteligentes se basa en la generación de ingentes cantidades de datos provenientes de sensores para su posterior gestión. Del mismo modo, los datos generados a través de los procesos de la ciudad están destinados a

facilitar la regulación de procesos urbanos dentro de un bucle de detección y actuación humano-maquina.

2.1.1. Cuestiones Ambientales

Después de haber avanzado en qué consiste la monitorización medioambiental, vamos a dedicar este apartado a comentar más detenidamente los diferentes aspectos que en ella se engloban.

El aire

Debido a lo perjudicial que resulta para la salud la polución atmosférica, esta problemática es una de las más acuciantes en la actualidad. Para detectarla, se lleva a cabo un control de la calidad del aire utilizando sensores que se encuentran al aire libre; los cuáles tienen en cuenta diversas emisiones como pueden ser: las de dióxido de carbono (CO2), óxido nítrico (NO), dióxido de nitrógeno (NO2) y partículas micrométricas (PM).

El agua

Las Smart Cities como todo núcleo urbano captan y hacen uso de una gran variedad de fuentes, que van desde ríos a aguas recicladas. Algunos procesos de tratamiento resultan esenciales para el aprovechamiento y transformación del agua para su posterior consumo. El agua sobrante y la que cae en forma de lluvia, se lleva a la red de saneamiento de manera que pueda ser depurada; bien para volver a ser consumida o para que sea liberada en el medio [4].

La función de los sensores pasa por evitar inundaciones de la red de saneamiento, así como, la escasez del agua de consumo. Para ello se monitorizan tanto los niveles del agua existentes en la red, como su presión en las tuberías. Asimismo, gracias a un control constante se puede utilizar agua menos depurada para un proceso que no requiera su total depuración, como puede ser el de regadío.

El ruido

El aumento de la población y de la actividad frenética en las ciudades ha provocado un aumento proporcional del ruido [5]. Aunque la principal fuente de ruido en las ciudades es el tráfico, hay otras fuentes que causan también graves molestias a la ciudadanía, como las actividades al aire libre o el ocio nocturno. Los sensores se encargan de monitorizar los niveles de ruido, preferentemente en lugares en los que suele haber una gran concentración de personas.

Los residuos

Los servicios de limpieza diarios de recogida y tratamiento de residuos se basan en rutas predefinidas que suponen un elevado coste económico para las ciudades, algunas veces innecesarios debido a que no es extraño que se recojan contenedores vacíos y se pasen por alto otros contenedores llenos, generando sobrecostes y proporcionando un mal servicio a los ciudadanos.

Para gestionar este problema se utilizan sensores; los cuales, permiten conocer de antemano el nivel de llenado de los contenedores, de forma que sea posible optimizar las rutas de recogida. Además, se pueden tener sensores adicionales tanto de temperatura como de ubicación con los que poder saber si se está produciendo un incendio o se ha movido el contenedor.

2.2. SMARTSANTANDER

SmartSantander es un proyecto de investigación científica perteneciente al 7ª Programa Marco de la Comisión Europea, en el que se han diseñado, desplegado y validado una plataforma para la experimentación en la IoT compuesta por más de 20.000 dispositivos (sensores, captadores, actuadores, cámaras, terminales móviles, etcétera) por toda la capital cántabra, formando un espacio virtual donde los objetos se comunican entre sí y trasmiten información para las personas con el fin de mejorar su bienestar (calidad de vida).

El campo de pruebas que la ciudad ofrece, crea un esquema de colaboración, entre lo público y lo privado, cuestión que ha sido desde el inicio, uno de los motores del proyecto SmartSantander, no ya sólo en el Proyecto Europeo que le dio nombre, sino en el global de la idea de Santander como ciudad inteligente El núcleo principal de las instalaciones que comprenden más de 20000 dispositivos, se localiza en la ciudad de Santander y sus alrededores, incluyendo puntos singulares de la Comunidad de Cantabria [6].

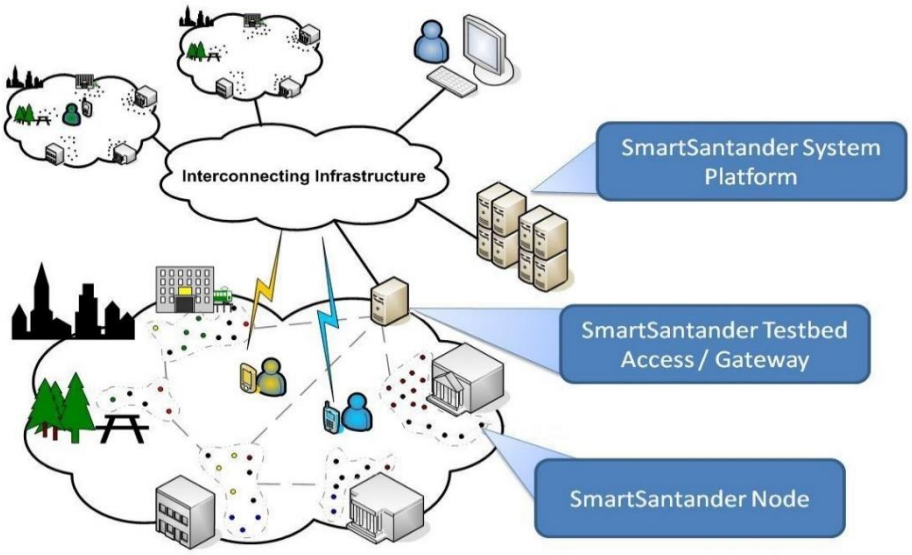

Figura 1 Representación Gráfica de SmartSantander

SmartSantander es una aplicación real y a gran escala de la llamada "computación ubicua", y esta computación es un paradigma de la telemática que tiene por objeto la integración de pequeños dispositivos y sensores en todo los aparatos y objetos de nuestra vida cotidiana, de tal forma que se puedan interconexionarse entre ellos, pudiéndose intercambiar de este modo información útil entre ellos. Lo que se ha venido también a llamar comunicación Máquina a Máquina (M2M) [7].

Como vemos en la Figura 1, los dispositivos que utiliza son de muy pequeño tamaño (nodos), con una capacidad de computación muy limitada, pero a los que es posible acoplar sensores de diversa naturaleza para que puedan captar datos del entorno. Estos datos fluyen a través de la red hasta un punto de concentración donde la información que captan los sensores es tratada con el objeto de obtener un servicio útil.

Para la consecución de los objetivos del proyecto, la plataforma software y la infraestructura física desplegada deben ofrecer una serie de características necesarias para que la futura experimentación pueda realmente suponer un avance cualitativo en la investigación e innovación de los campos de la IoT, las Ciudades Inteligentes y la Internet del Futuro.

La plataforma SmartSantander, se basa en una arquitectura formada por tres niveles:

- Nivel de Dispositivo IoT, que proporciona el sustrato necesario compuesto por los propios dispositivos; estos son recursos limitados y exportan datos fiables que aseguran con una serie de medidas.

- El nivel de Gateway IoT, que enlaza los dispositivos IoT en los bordes de la red a una infraestructura de red central.
- El nivel de Servidor, que dispone de dispositivos de gran capacidad, los cuales son conectados directamente con la infraestructura de red central. Los servidores pueden usarse como repositorios de datos IoT, y como servidores de aplicación que pueden ser configurados para ofrecer una gran variedad de diferentes servicios IoT y aplicaciones.

2.3. LASS

LASS consiste en un proyecto sin ánimo de lucro iniciado por Wuulong Hsu que comenzó como un proyecto local taiwanés con el objetivo de monitorizar la contaminación del aire y se está extendiendo por todos los continentes [8]. El alcance de los despliegues que usan el proyecto LASS se puede observar en la Figura 2.

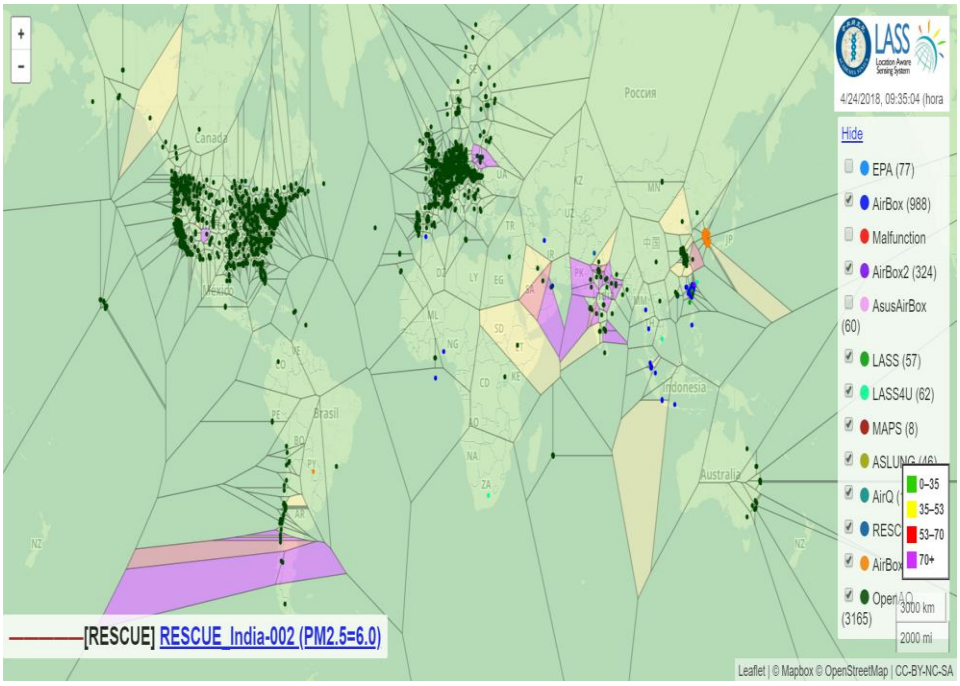

Figura 2: Dispositivos que actualmente utilizan LASS para monitorizarse

El funcionamiento de LASS se basa en una plataforma de código abierto, la cual puede ser equipada con una gran variedad de sensores que son capaces de medir diferentes aspectos de la calidad del aire, desde partículas PM2.5 o CO2 hasta temperatura y humedad.

En cuanto a su arquitectura, la plataforma está formada por sensores que a través de Internet envían la información que recogen a un servidor central utilizando para ello el protocolo Message Queing Telemetry Transport (MQTT). Dicha arquitectura hace

posible acceder a los datos de manera directa desde cualquier dispositivo conectado a Internet, bien a través de un servicio web (ej. basado en Google Maps) o directamente obteniéndolos del sensor; en ambos casos, el dispositivo se suscribe al servidor. En la Figura 3 se puede observar dicha arquitectura [9].

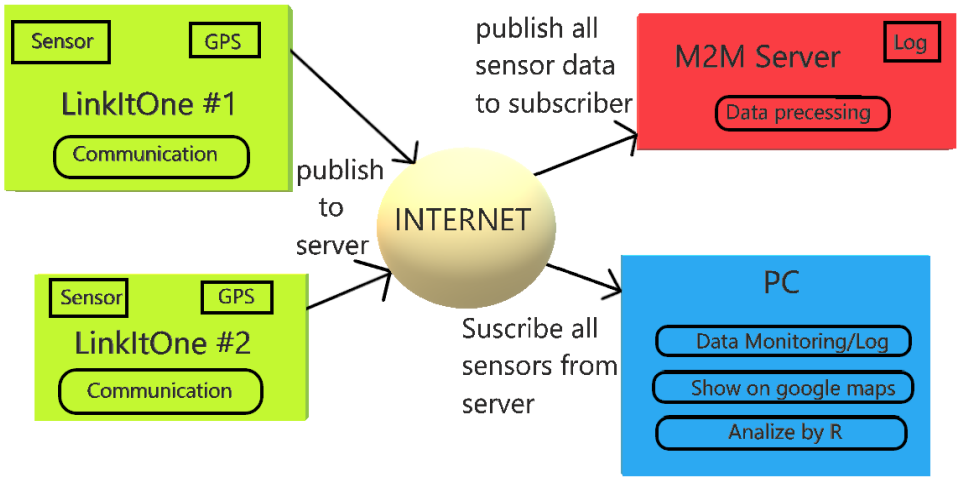

Figura 3. Arquitectura de la plataforma LASS

Así pues, el proyecto pretende que el usuario pueda montar de una forma sencilla su dispositivo. Una vez montado deberá conectarlo al servidor propio de LASS y suscribirse al mismo con el topic LASS/#. Conviene señalar que el usuario no solamente es responsable de montar el sensor, sino también, de su mantenimiento. Dicha arquitectura puede ser apreciada en la Figura 4.

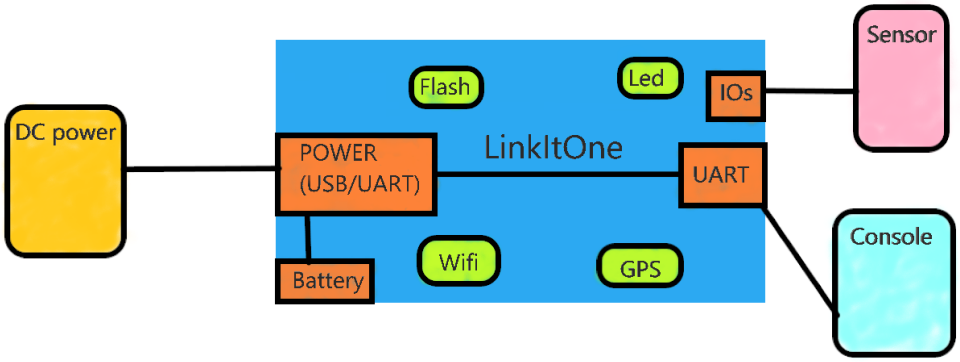

Figura 4. Arquitectura de los dispositivos que integran LASS

2.4. EL PROTOCOLO "MESSAGE QUEING TELEMETRY TRANSPORT (MQTT)"

2.4.1. Arquitectura del protocolo

El protocolo MQTT fue ideado por Andy Standford-Clark y Arlen Nipper en 1999. Se trata de un protocolo usado para la comunicación machine-to-machine(M2M) /" Internet of things", así como de un estándar ISO [10]. Trabaja por encima de la capa TCP/IP, entre las de transporte y aplicación. En la Figura 5 se puede observar la relación de capas OSI en MQTT.

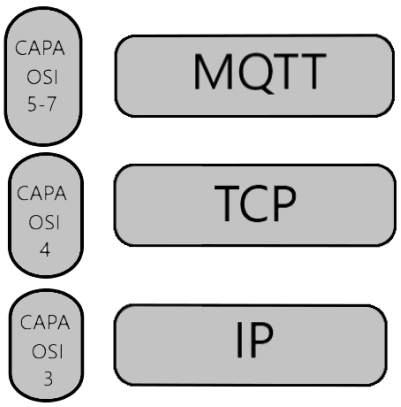

Figura 5. Relación Capas OSI y MQTT. A la derecha se puede ver las capas OSI y a la izquierda su equivalente MQTT

Su función principal consiste en el transporte de publicaciones y suscripciones de muy poca carga mediante mensajes Cliente/Servidor [11]. Estas características han propiciado su uso en varias industrias entre las que se encuentran las comunicaciones de sensores. Cada mensaje está formado por un encabezado fijo de 2 bytes, un encabezado opcional, el propio mensaje, que tiene una capacidad máxima de 256 bytes y el nivel de calidad de servicio (QoS).

El cliente puede ser tanto editor como suscriptor. Cualquier dispositivo conectado a un servidor MQTT (broker) a través de una red y ejecutando una biblioteca MQTT puede ser cliente.

El broker es la parte central de cualquier protocolo de publicación/suscripción; puesto que es el encargado de recibir, filtrar y decidir en qué publicaciones están interesados los clientes; para, posteriormente, envíales sólo aquellos mensajes

procedentes de las publicaciones a las que se encuentren inscritos. Además, también se ocupa de la autentificación y autorización de los clientes.

El protocolo utiliza una arquitectura basada en estrella en la que el servidor ocupa el nodo central mientras que los clientes utilizan los nodos periféricos, estando todos ellos unidos al servidor/bróker tal como se observa en la Figura 6. Dicho nodo central se encarga de hacer de intermediario entre los diferentes clientes, transmitiendo y gestionando el conjunto de mensajes [12].

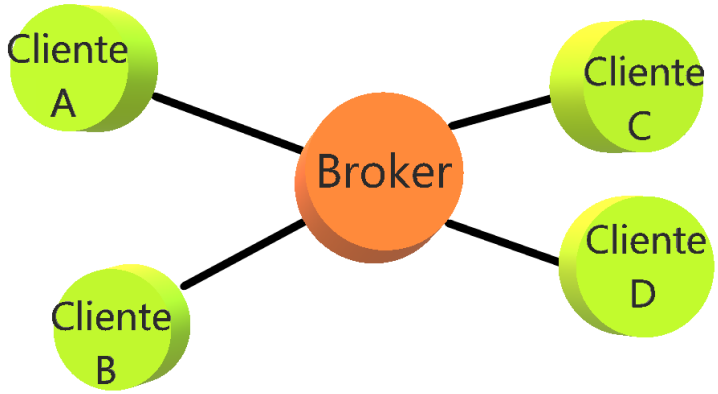

Figura 6. Arquitectura de MQTT

El funcionamiento del protocolo empieza con el establecimiento de la conexión con el cliente enviando un mensaje CONNECT al broker, el cual a su vez responde con un mensaje CONNACK.

Para la transmisión de datos el cliente se subscribe a un tema, denominado *topic* que posee una estructura jerárquica, a través del envió de un paquete PINGREQ al servidor [13]. Dicho servidor recibe múltiples topics y cuando encuentra un paquete del correspondiente topic, lo envía al cliente mediante un publish.

2.4.2. Manejo de la Calidad de Servicio (QoS)

La calidad del servicio (QoS) hace referencia a la priorización del tráfico, así como al mecanismo de control de recursos reservados. Cuanto más alto sea el nivel de QoS, más confiable será, sin embargo, mayores serán también los requisitos de latencia y ancho de banda. Dicha calidad del servicio, en MQTT, consta de 3 niveles:

El QoS 0 es el más simple al constar únicamente de una secuencia de paquetes PUBLISH. El funcionamiento de este nivel es el siguiente: el editor envía un mensaje al bróker, el cual lo envía al suscriptor. En ambos casos solo se envía una vez debido a que

el nivel no cuenta con mecanismos de control para saber si el mensaje ha sido transmitido correctamente y tampoco se almacenan los mensajes.

El QoS 1 está formado por secuencias de paquetes PUBLISH/PUBACK tanto entre el editor y el servidor, como entre el servidor y el suscriptor. Los mensajes contienen un mecanismo de acuse de recibo que permite constatar la correcta transmisión del mensaje y en el supuesto de no ser recibido en un cierto intervalo de tiempo, se reenvían. Esto puede producir que el suscriptor reciba varias copias del mismo mensaje.

Por ultimo nos encontramos con el nivel QoS 3, que asegura la correcta transmisión del mensaje una sola vez. Su funcionamiento está basado en dos pares de paquetes, los PUBLISH/PUBREC y los PUBREL/PUBCOMP.

2.4.3. Seguridad en MQTT

La seguridad en el protocolo se ha implementado de dos maneras: la primera consiste en encriptar los mensajes a través de un usuario y contraseña. La segunda forma pasa por encriptar la red con SSL, aunque hay que tener en cuenta que SSL al no tratarse de un protocolo ligero, puede provocar la sobrecarga de la red.

Por último, no debemos olvidar que MQTT no fue diseñado teniendo en cuenta la seguridad; por este motivo, a pesar de haber sido agregado posteriormente, el nivel de seguridad no es demasiado robusto.

2.5. NODE-RED

Node–Red es una herramienta open-source de programación basada en flujos desarrollada por IBM Emerging Technology y actualmente forma parte de la fundación JS [14]. Dicha herramienta utiliza el lenguaje JavaScript para crear funciones, de modo que permite conectar diversos dispositivos hardware, APIs y servicios online.

La programación basada en flujos describe el comportamiento de una red de nodos, en la cual cada uno cumple una función específica; siendo la red responsable de la información que viaja entre los nodos.

Concretamente, Node-Red se basa en Node.js que utiliza un navegador web para acceder al editor de flujos. En el editor las aplicaciones se crean arrastrando los nodos desde la paleta al área de trabajo, así como conectándolos entre sí. La paleta de los nodos es posible ampliarla, gracias a una comunidad activa que contribuye creando y

publicando sus nodos como archivos JSON. Se pueden apreciar todos elementos en la Figura 7.

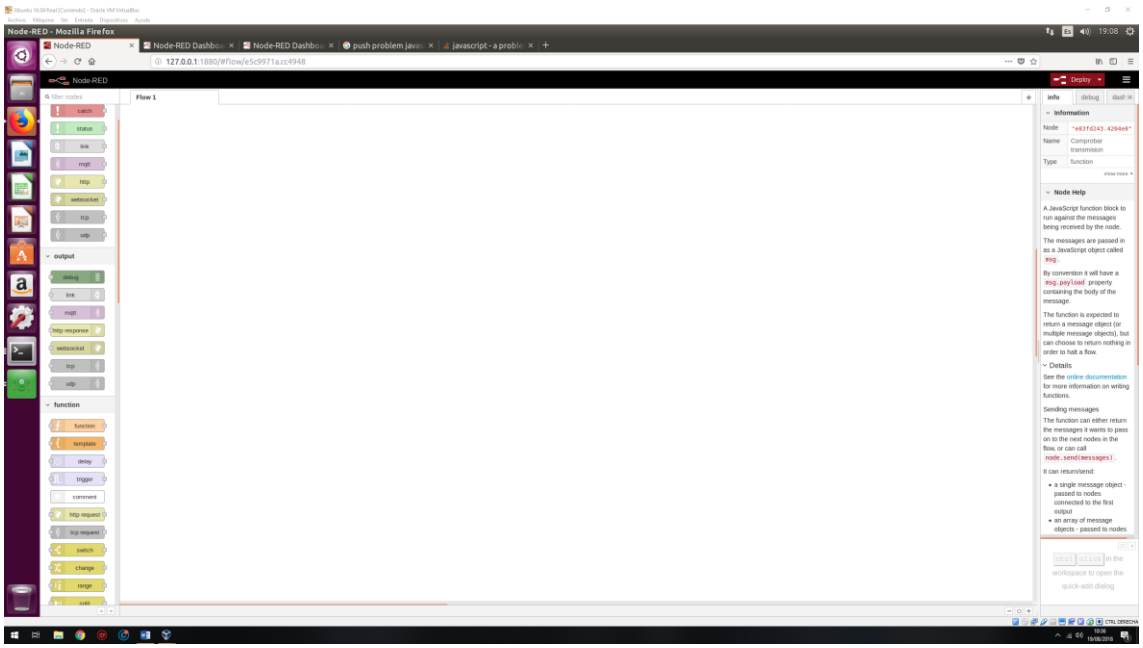

Figura 7. Node-Red ejecutándose en el navegador web Mozilla Firefox

Capítulo 3: Sistema de Monitorización PM2.5

En este capítulo se describen las tareas llevadas a cabo para implementar el sistema de monitorización de partículas PM2.5. Para ello, principalmente se han abordado una serie de tareas que se pueden dividir en dos actividades claramente diferenciadas. Por un lado, está la implementación e integración de los sensores que se encargarán de hacer las observaciones de la calidad del aire. Esta actividad, implica la integración del hardware del dispositivo, así como la configuración de su firmware para su correcto funcionamiento. Por otro lado, se ha desarrollado un servicio que se encargará de recoger las medidas obtenidas por el sensor y presentarlas a través de un interfaz gráfico para su visualización. En esta actividad, se ha llevado a cabo el despliegue de los servidores con los que el sensor se comunica, así como las bases de

datos para el almacenamiento de los datos y del interfaz gráfico para la representación de los valores obtenidos.

3.1. DESCRIPCIÓN DEL SENSOR

El sensor que se ha ensamblado podrá medir los niveles de Partículas Micrométricas de menos de 2.5 μm ($PM_{2.5}$, partículas finas) y de menos de 10 μm (PM_{10}, partículas respirables), humedad, temperatura, así como geo-localizar dichas medidas a través de la longitud y latitud.

Para ello se va a hacer uso de una serie de dispositivos que se enumerarán y describirán a continuación.

3.1.1. LinkIt One

Este dispositivo es un PC embebido con un rendimiento destacable para su reducido tamaño. Es comparable a las más conocidas Raspberry Pi [15]. El dispositivo LinkIt One ha sido diseñado por Seeed Studio y MediaTek. La placa de desarrollo LinkIt One es una placa de código abierto de alto rendimiento diseñada para implementar dispositivos embebidos y de IoT. La placa consta del SoC MediaTek Aster MT2502, así como de un conjunto de chips como WiFi o GPS. Asimismo, proporciona un conjunto de pines de entrada/salida con características similares a los pines de las placas Arduino, para facilitar su conectividad a sensores u otros periféricos entre otros [16].

Dimensiones	108mm x 75mm x 33mm
Peso	102.50g
Chip	MT2502A
Velocidad de Reloj	260MHz
RAM	4MB
Flash	16MB
Corriente continua por pin E/S	1mA
Pins Analogicos	3
Salida Digital	3.3V
Entrada Analogica	5V
Tarjeta SD	32GB
Posicionamiento	GPS

GSM	850/900/1800/1900 MHz
WiFi	802.11 b/g/n
Bluetooth	BR/EDR/BLE(Dual Mode)

Tabla 1. Especificaciones técnicas de LinkIt One.

3.1.2. Sensor PMS3003

Este sensor, desarrollado por Plantower, es capaz de valorar la calidad del aire a través de la medida de las concentraciones de partículas micrométricas PM_1, $PM_{2.5}$ y PM_{10} en mg/m3 [17].

Dimensiones	65 × 42 × 23 mm
Rango de medición	1.0 to 2.5 ; 2.5 to 10 (mm)
Recuento de eficiencia	50%@0.3um 98% @> = 0.5 um
Tiempo de respuesta	≤ 10 sec
Suministro de voltaje en CC	5 V
Corriente de funcionamiento	120 mA
Corriente de espera	≤ 200 μA
Rango de temperatura operativa	-20 ~ + 50 °C
Rango de humedad operativa	0 to 99%

Tabla 2. Especificaciones técnicas de PMS3003

3.1.3. Sensor de temperatura y humedad

Este sensor está formado por un sensor capacitivo que se utiliza para medir la humedad relativa y un termistor de coeficiente de temperatura negativo (NTC) para medir la temperatura. Aunque sean parámetros aparentemente independientes de la calidad del aire, se ha estudiado como sus valores son importantes a la hora de aplicar los modelos de calidad del aire [18].

Dimensiones	40 × 20 × 11 mm
Rango de temperatura operativa	-40 ~ + 80 °C
Rango de humedad operativa	5 to 99%
Suministro de voltaje en CC	3.3 - 6V
Corriente de funcionamiento	1 – 1.5 mA
Tiempo de respuesta	6 - 20 sg

Tabla 3. Especiaciones técnicas de Grove.

3.1.4. Antenas GPS, WiFi/bluetooth y GSM

Estas antenas proporcionan la conectividad del sistema a través de redes externas.

Por un lado, permiten la recepción de datos provenientes de los satélites GPS para la obtención de la posición del sensor en términos de latitud y longitud.

Las segundas, permiten el intercambio de datos inalámbricamente tanto a través de redes WiFi como Bluetooth.

Las últimas permiten el intercambio de información a través de redes móviles GSM.

3.1.5. Batería ion-litio

Batería con la cual se alimentarán los diferentes dispositivos conectados a la placa base, así como la propia placa.

3.2. IMPLEMENTACIÓN DEL SENSOR

La implementación que se ha realizado, se ha acometido en dos fases: por un lado, el ensamblaje del hardware en la cual se han conectado los diferentes dispositivos con la placa base y, por otro lado, la carga del software, en la que se ha instalado el firmware necesario para el correcto funcionamiento del sensor.

3.2.1. Hardware del sensor

Para empezar, se conectaron las antenas GSM, GPS y WiFi/Bluetooth a la parte trasera de la placa por medio de sus correspondientes interfaces [19]. El resultado final una vez conectados todos los periféricos se puede apreciar en la Figura 8 y Figura 9 respectivamente.

Para conectar los sensores, se utilizaron cables que terminaban en conectores macho, los cuales se conectarían a los pines de la placa base.

Para el sensor de temperatura y humedad se hicieron las siguientes conexiones:

- cable amarillo al pin D2
- cable blanco al pin D3
- cable rojo al pin 3V3
- cable negro al pin GND

Figura 8. Placa base conectada a las diferentes antenas

Para el sensor de partículas las conexiones fueron las siguientes:

- cable azul al pin TX D1
- cable verde al pin RX D0
- cable naranja al pin GND
- cable purpura al pin 5V

Para finalizar se conectó la batería y se conectó al ordenador a través del puerto USB de la placa para su configuración mediante el puerto serie.

3.2.2. Software del sensor

Se comenzó por descargar e instalar tanto el entorno de desarrollo para placas Arduino (Arduino IDE) como el driver de puerto COM USB para LinkIt ONE. Aun no siendo una placa Arduino como tal, la LinkIt One utiliza los mismos procedimientos de carga de programas por lo que el entorno de desarrollo es muy apropiado tanto para el desarrollo de código como para su posterior carga y supervisión.

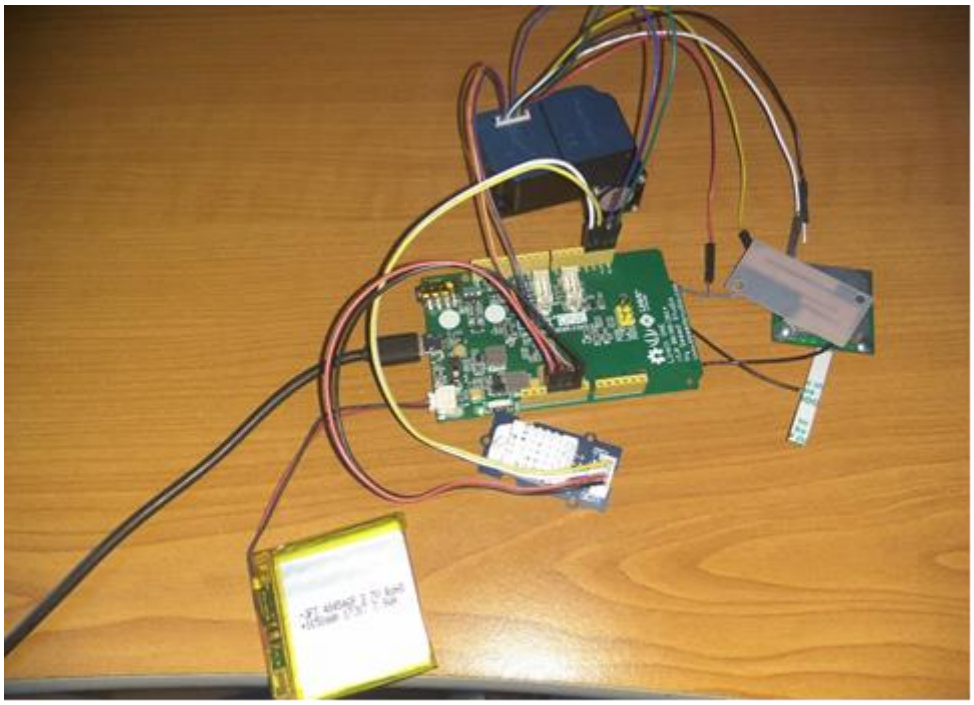

Figura 9. Hardware final del sensor

Dentro del IDE de Arduino se comenzó por descargar un gestor específico para la placa LinkIt One. Para ello se fue a la pestaña "Herramientas", y dentro del menú "Placa" se abrió el "Gestor de tarjetas". Dentro del gestor de tarjetas se instaló "LinkIt ONE by Seeed Studio and MediaTek Labs". Para finalizar esta parte se fue otra vez a "Placa" pero esta vez para seleccionar la nueva opción "LinkIt ONE".

Para actualizar el firmware de la placa se fue a la placa base y se cambió el interruptor a la opción "Mass Storage Bootup". Después, se procedió a ir a Arduino y en la pestaña "Herramientas" se clico en "Quemar Bootloader". Se abrió una nueva ventana en la cual tras darle a descargar se exigía cumplir 2 exigencias que eran primero retirar el usb del ordenador, así como apagar la placa y después volver a encenderla y reconectarla. Tras estos pasos se completó la actualización del firmware de la placa LinkIt ONE.

Ahora se procedió a establecer el puerto del dispositivo. Para ello lo primero fue poner el interruptor de la placa en "Normal Bootup Mode". Tras ello se fue al "panel de control", se clico en el" administrador de dispositivos" y en el apartado "puertos" se buscó el COM del "MTK USB Modem Port". Para terminar, en Arduino se fue a la pestaña "Herramientas", en el apartado "puertos" se seleccionó el COM indicado previamente.

Más tarde se descargaron e instalaron las siguientes librerías: “DHT_linkit”, “HP20x_dev”, “AWSArduinoMediaTekLibrary”, “Grove_Digital_Light_Sensor”, “PubSubClient” y “HardwareLibrary”. Para instalarlas se fue a la pestaña “programas”, en “incluir librerías” se seleccionó “añadir librería zip” [19].

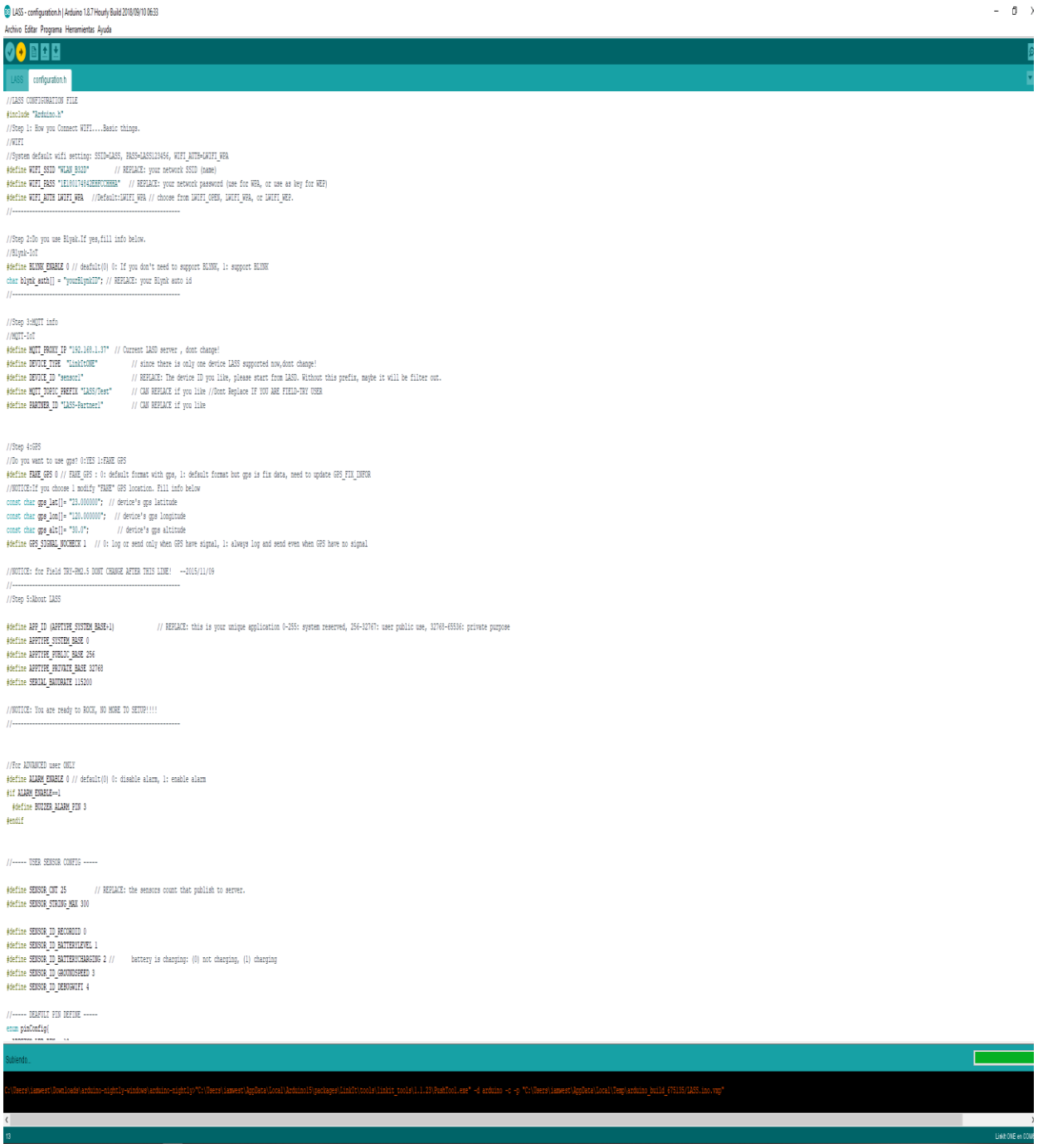

Figura 10. Archivo configuration.h en Arduino

Se descargaron el proyecto LASS.ino y el archivo configuration.h [20].

Para finalizar, se abrió configuration.h (presentado en la Figura 10) y se procedió a cambiar los siguientes parámetros [19]:

- WIFI_SSID: para este parámetro es necesario indicar el SSID de la red WiFi a la que se desea conectar el sensor.
- WIFI_PASS: debe contener la contraseña se escribió la contraseña del wifi.
- WIFI_AUTH: se seleccionó la autentificación del wifi.
- MQTT_PROXY_IP: se seleccionó la IP del ordenador.
- DEVICE_ID: se puso el nombre con el cual se iba a denominar al sensor.
- MQTT_TOPIC_PREFIX: se añadió parte del topic.

Una vez realizadas estas modificaciones se procedió a subir el proyecto al sensor.

3.3. SISTEMA DE MONITORIZACIÓN

El sistema de monitorización es el servicio que se ha implementado para recoger los datos que generan los sensores y trasladarlos hacia un interfaz gráfico para su visualización.

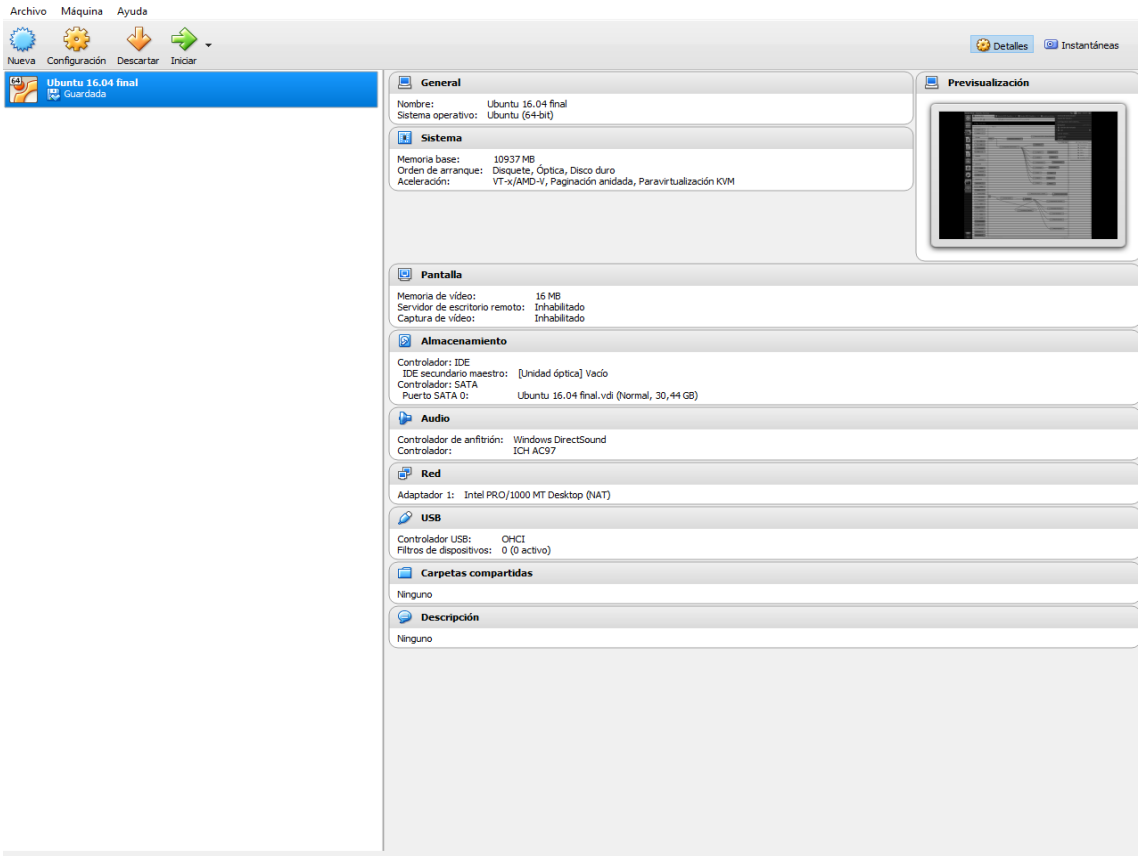

Figura 11. Sistema operativo Ubuntu en Oracle VM VirtualBox.

Todo este sistema se ha implementado sobre una máquina virtual con el fin de poder hacer este sistema completamente reproducible en cualquier entorno lo que aumenta el impacto de este TFG.

Se comenzó por ejecutar el sistema operativo Ubuntu 16.04 en una máquina virtual, para lo cual se usó el programa Oracle VM VirtualBox [21] tal como se aprecia en la Figura 11.

Inmediatamente después se procedió a instalar *Node.js* utilizando el terminal de Ubuntu, dicho programa es la base de *node-red* [22]. Para comprobar su correcta instalación se comprobó su versión.

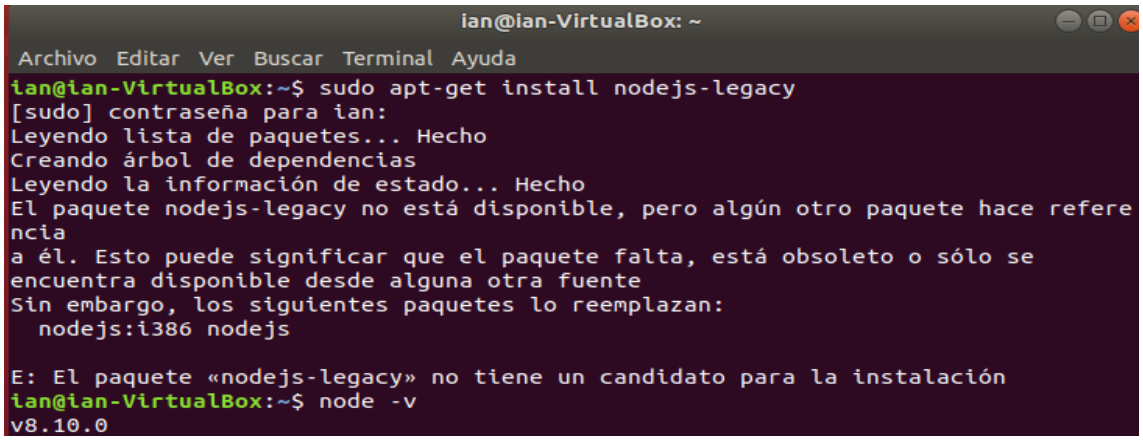

Figura 12. Instalación y verificación de Node.js

A parte de *Node.js* también era necesario el administrador de paquetes *npm*, el cual se instaló y actualizó su versión.

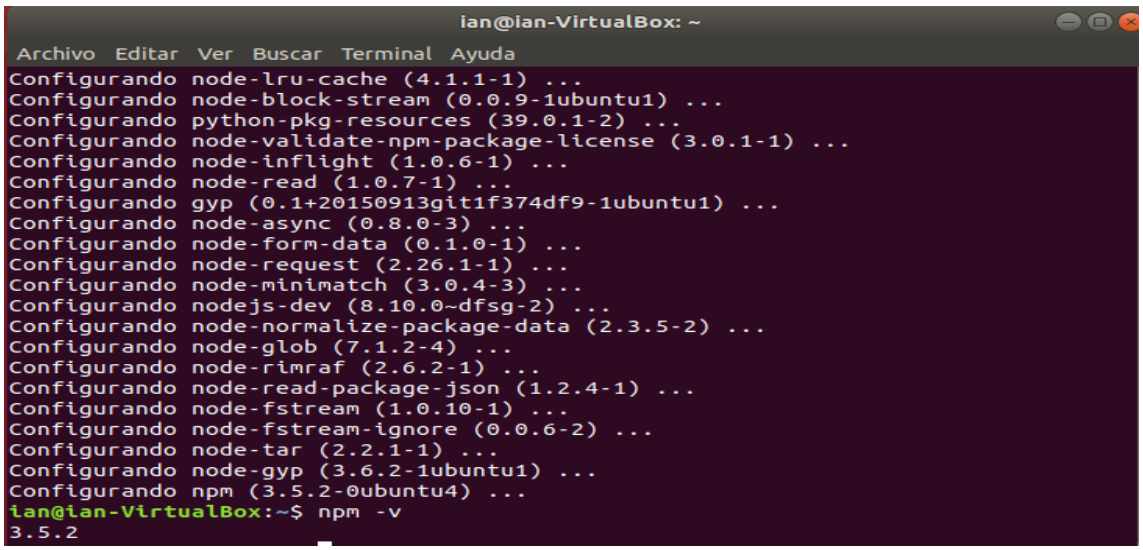

Figura 13. Verificación de npm.

Tras instalar tanto *Node.js* como *npm* se procedió a instalar *node-red*; para su correcto funcionamiento también se tuvo que habilitar el uso del puerto 1880.

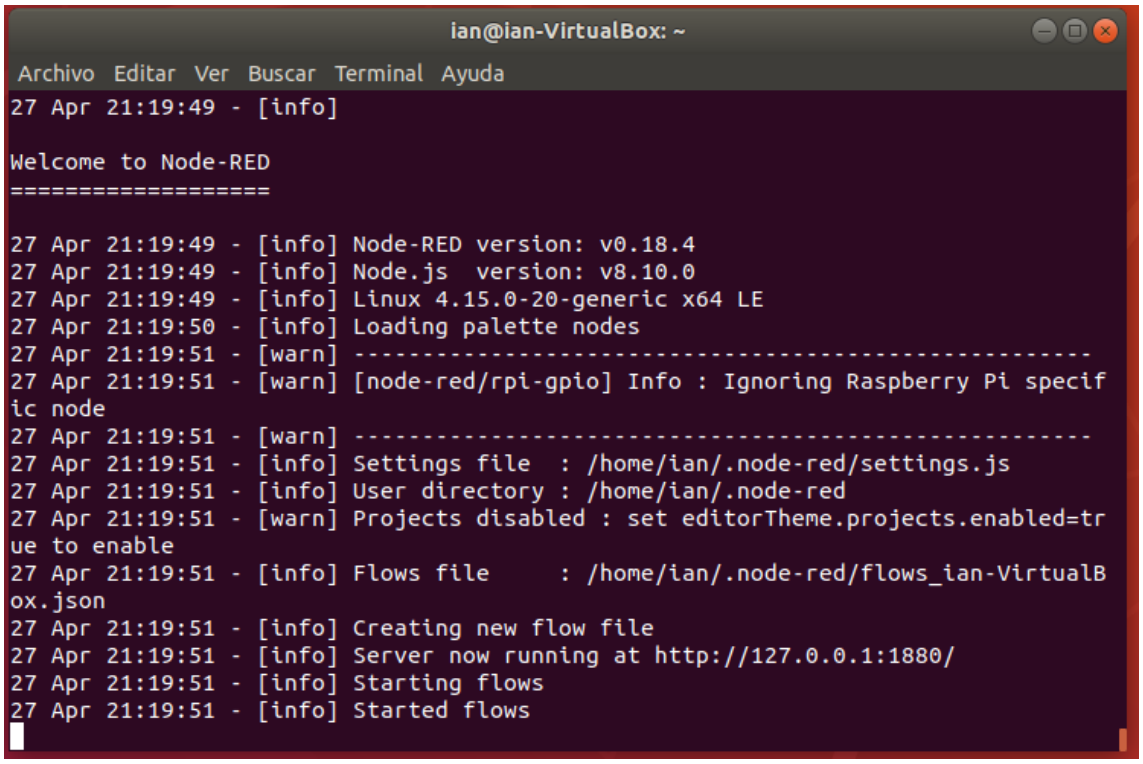

Figura 14. Resultado en el terminal al ejecutar Node-Red

Es importante destacar que el sistema de monitorización que se implementado permite la inyección de medidas desde otros sensores aparte del que se ha desarrollado en este TFG. De hecho, la implementación del sistema de monitorización se acometió antes de tener completamente ensamblado el sensor por lo que se buscó un sensor con características similares al que posteriormente se iba a utilizar, para que los datos de dicho sensor idénticos al que se iba a utilizar después. Para localizar dicho sensor se accedió al servidor MQTT del proyecto LASS (gpsensor.ddns.net) y se configuró una suscripción a cualquier mensaje publicado en un topic que contuviera la secuencia de caracteres “LASS/” [23].

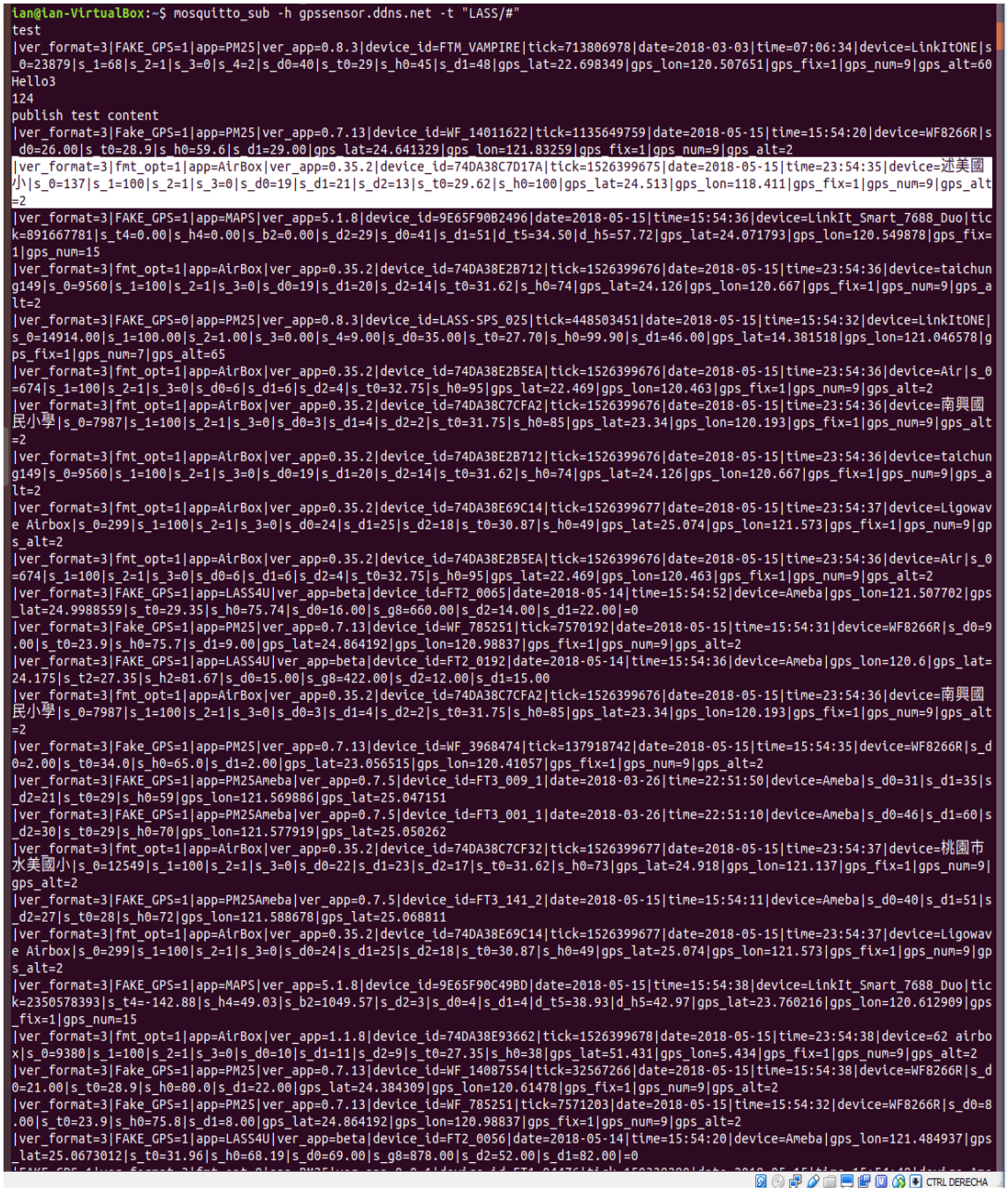

Figura 15. Resultados al buscar sensores con el topic "LASS/" y el servidor "gpsensor.ddns.net".

Para la recepción de los datos del sensor se empleó el bróker *mosquitto* que genera un servidor MQTT. No se ha comentado con anterioridad, pero el firmware instalado en los sensores compila la información obtenida y la envía mediante mensajes MQTT al servidor indicado en la configuración de los mismos. Para ello se necesitaba instalar el programa y las librerías [24]. También se decidió instalar el cliente para poder monitorizar en el mismo sistema que los mensajes publicados por los sensores se recibían correctamente en el servidor.

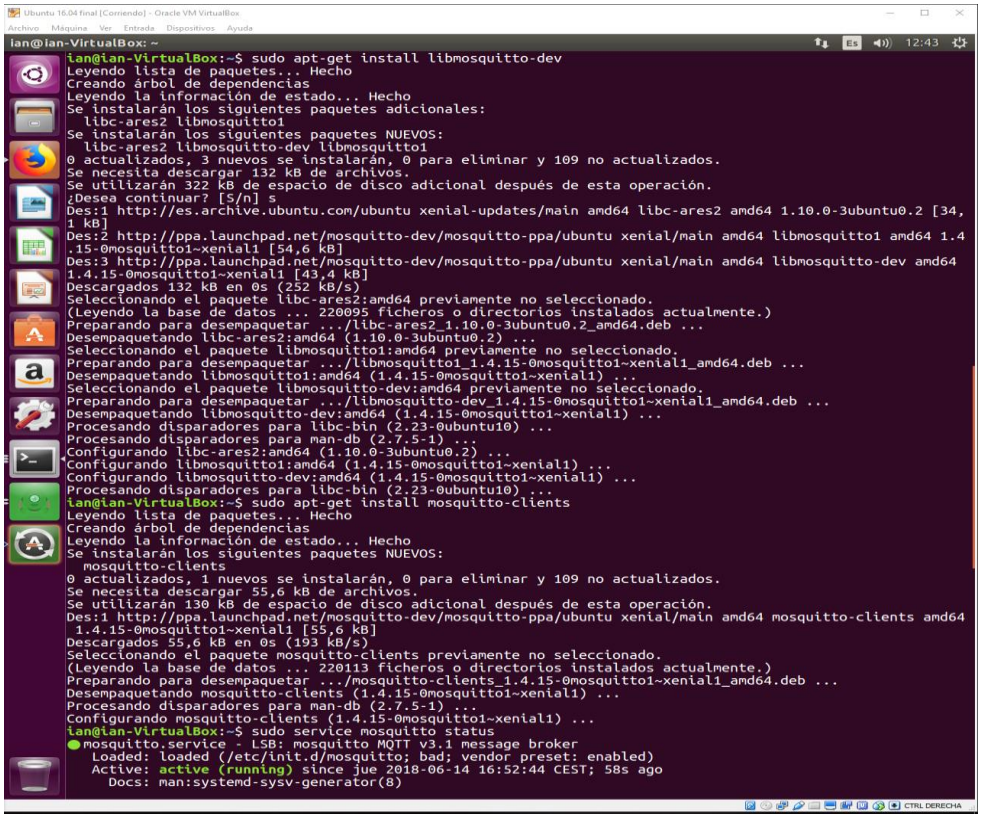

Figura 16. Resultado al instalar el servidor MQTT *mosquitto*

Después de activar el servidor MQTT y una vez implementado el sensor, se fue a la configuración de la máquina virtual y en el menú "red" se configuró la opción de "reenvió de puertos". Esto es necesario ya que el servidor se encuentra en el espacio virtual y es necesario configurar el gestor de máquinas virtuales para que actúe como NAT. Para ello se agregó la IP que tenía la máquina virtual y se la hizo corresponder con la del ordenador que actuaba como anfitrión. El puerto utilizado es el puerto por defecto de MQTT, el 1883. El resultado de esta configuración se puede apreciar en la Figura 17

Figura 17. Reenvió de puertos a la máquina virtual

Para finalizar se accedió a *node-red*, en dicho programa se incluyó un nodo MQTT y una función. El nodo MQTT al cual se llamó LASS permitía recoger los datos de todos los sensores que en el MQTT bróker del servidor contuvieran dicho *topic*. Por otro lado, la función implementaba un código en JavaScript que discriminaba los datos en función del ID del sensor.

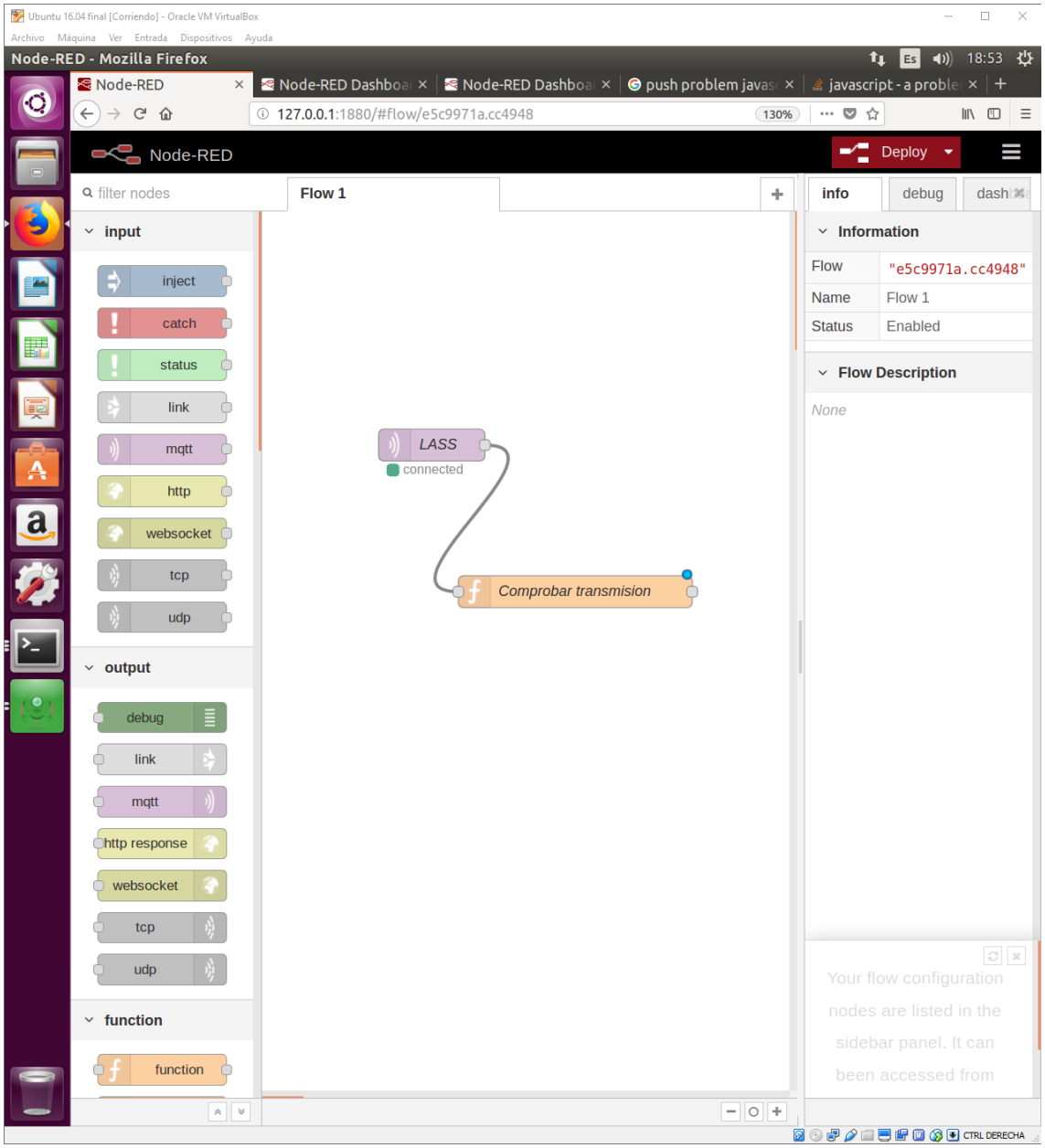

Figura 18. Vista final de la Monitorización en Node-Red.

3.4. DASHBOARD DE MONITORIZACIÓN

Un *dashboard* o interfaz gráfica es una representación gráfica de una colección de datos cuya función consiste en la visualización de dichos datos con el objetivo de

favorecer la toma de decisiones, así como transformar dichos datos en información útil orientada a conseguir los objetivos deseados.

Se comenzó por instalar un *dashboard* vacío en *node-red*, para lo cual se hizo uso de la terminal de comandos de Ubuntu. Después se fue a node-red; se clico en "Manage Palete", en la pestaña "install" se buscó e instalo "node-red-dashboard", mostrando las herramientas del *dashboard*, así como una interfaz gráfica en blanco [25].

En este aparatado del trabajo, se decidieron separar los datos que se mostraban en tiempo real y los datos históricos que se habían almacenado.

3.4.1. Datos en Tiempo Real

Al obtener el string del ID seleccionado, se procedió a separar los valores de longitud, latitud, temperatura, humedad, $PM_{2.5}$ y PM_{10}.

Para lograrlo se hizo uso de funciones, cada una de las cuales tenía un valor por objetivo. Para seleccionar el valor correcto, se cogía el nombre del atributo y se colocaba un puntero. Posteriormente se colocaba otro puntero al finalizar el valor al que hacía referencia el atributo. Para terminar, se seleccionaba lo que había entre ambos punteros y se dejaba únicamente el valor. Los atributos eran los siguientes:

- "gps_lat" para la latitud
- "gps_lon" para la longitud
- "s_t0" para la temperatura
- "s_h0" para la humedad
- "s_d0" para $PM_{2.5}$
- "s_d1" para PM_{10}

Tras ello se procedió a crear una etiqueta con el nombre del sensor y dentro de la etiqueta, un grupo con el nombre "Datos en Tiempo Real".

Para finalizar se conectaron las funciones a las herramientas del *dashboard*, Para la longitud y la latitud se utilizó un texto y para el resto se utilizaron medidores. A todos ellos se le añadió al grupo en "Datos en Tiempo Real".

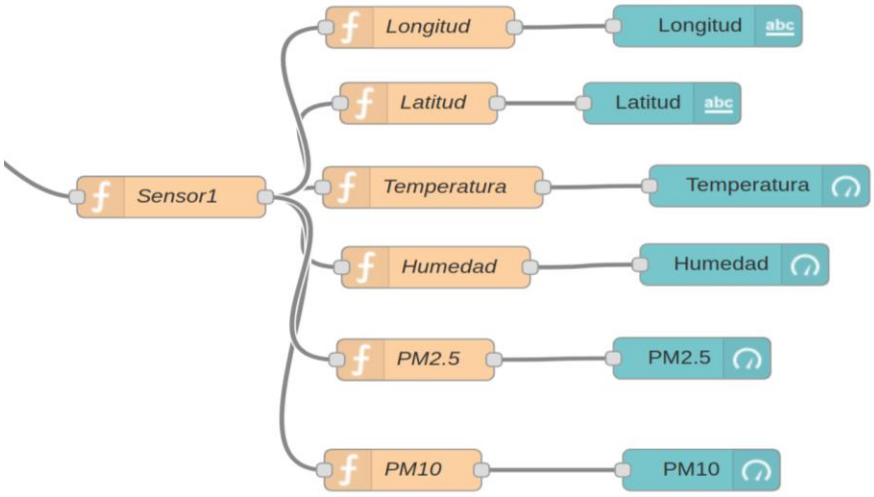

Figura 19. Resultado en node-red para los datos en Tiempo Real

3.4.2. Datos Históricos

En este apartado se querían crear gráficas, por lo que al ser necesario almacenar los datos, se hizo uso de *mongodb*. Para instalarlo se utilizó la terminal de comandos [26].

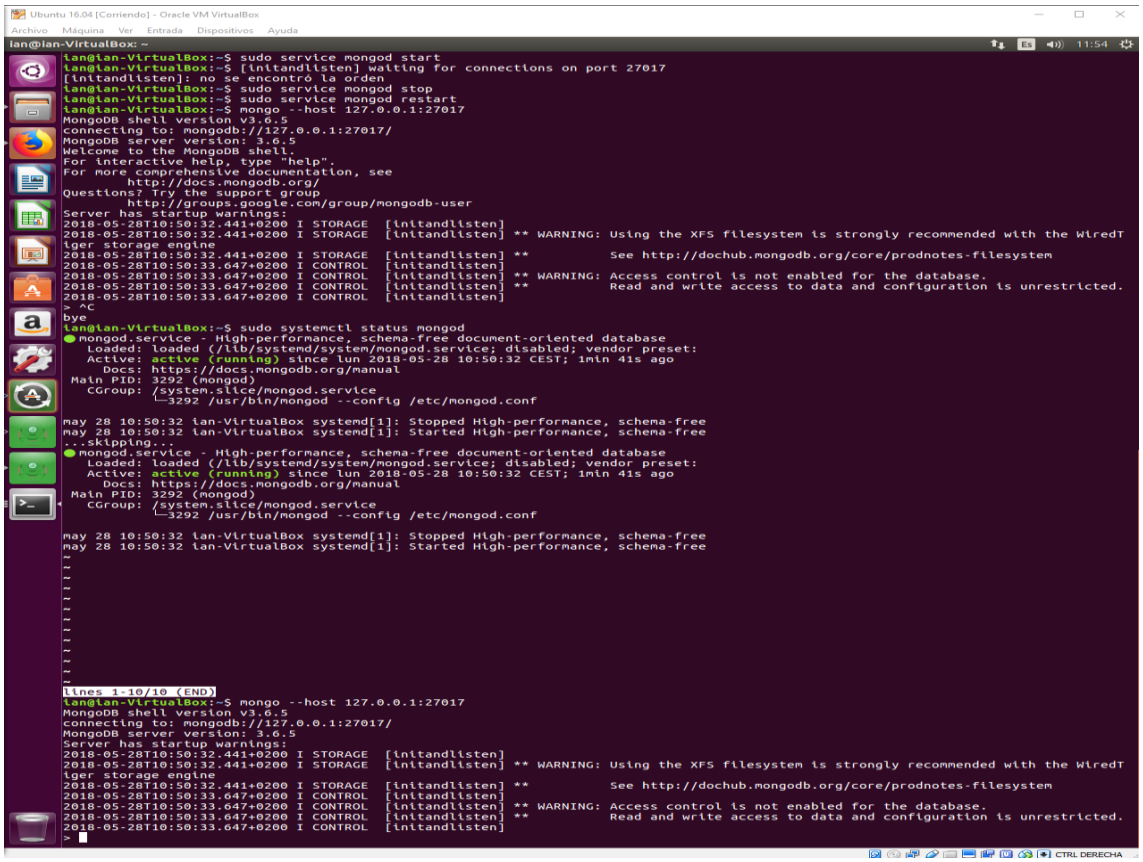

Figura 20. Resultado de la instalación de mongod

Tras comprobar su correcto funcionamiento, se fue a *node-red*, en "Manage Palete", en la pestaña "install" se buscó e instalo "node-red-mongodb2" [27]. De esta manera el nodo que se conecta al servidor *mongodb* de *node-red* se mostró en la paleta.

Asimismo, se descargó el programa *robo3t*, con el objetivo de crear bases de datos *mongodb* de manera más sencilla e intuitiva, además de visualizar el contenido de dichas bases de datos [28].

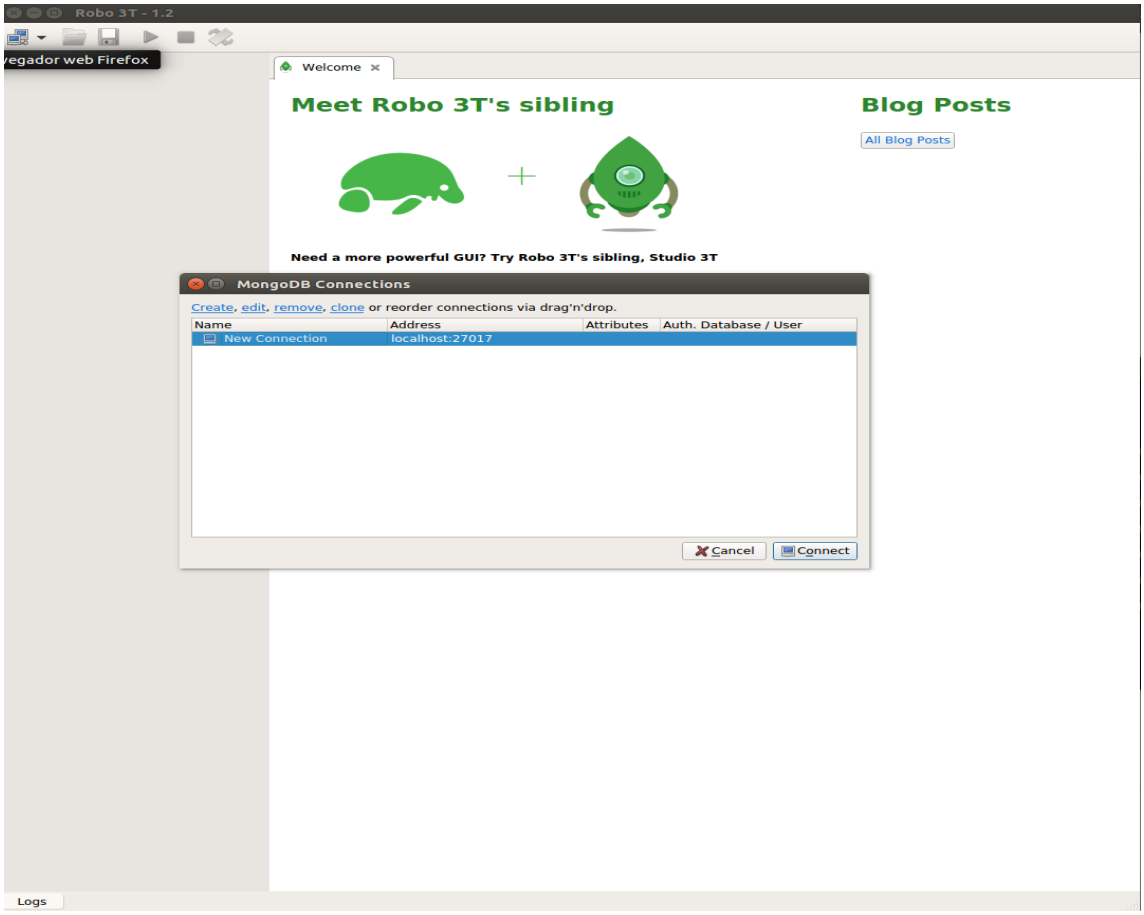

Figura 21. Ventana inicial de robo3t

Con el programa *robo3t* se creó una base de datos a la cual se llamó "sensor1" y dentro de ella una colección a la cual se llamó del mismo modo.

Para almacenar los datos en una base de datos se fue a *node-red*. En dicho programa, se comenzó por cambiar el formato de LASS a JSON mediante una función, la cual estaría unida a la función que distingue las IDs. Dicha función separa los valores de la misma manera que en el *dashboard* en tiempo real, además se agregó un *datetime* y se pasó al formato JSON mediante el uso de una variable.

Los datos en JSON se pasaron a un nodo *mongod* que enviaría los datos directamente a la colección "sensor1".

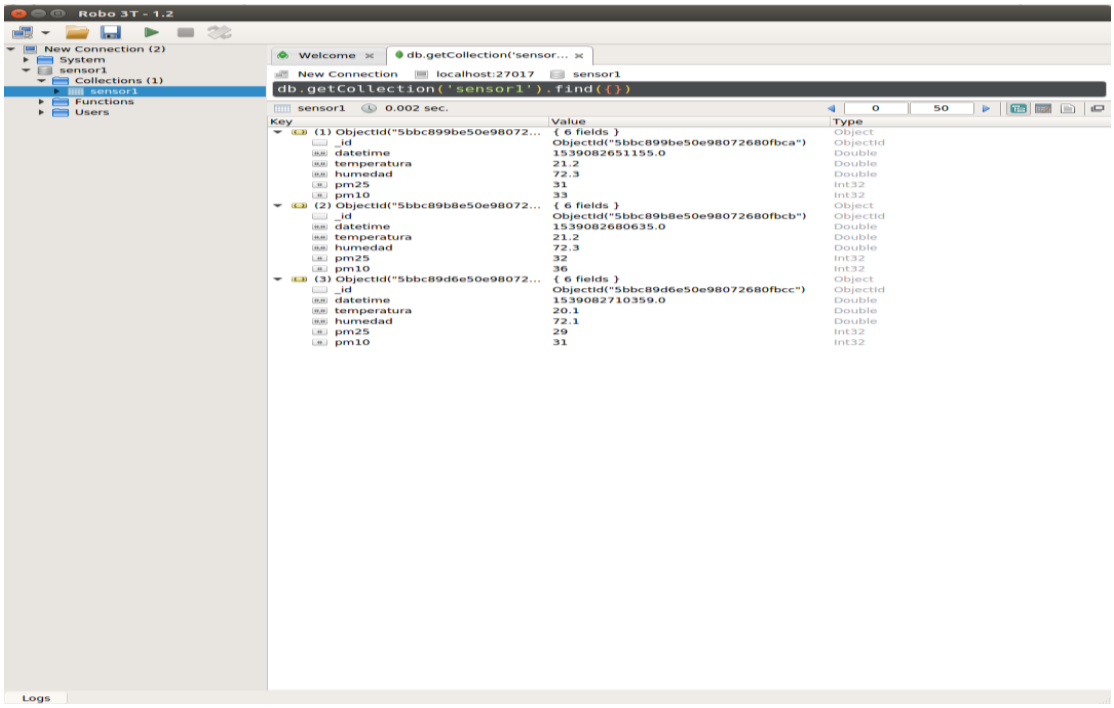

Figura 22. Colección de datos visualizada en robo3t

Ahora se procedió a sacar los datos guardados en la base de datos, para ello se empezó por colocar un "inject node" que se iniciara cada 0.1 segundos. Se conectó a una función que se nombró "enviar datos", la cual señalaba la colección, así como la operación y en el que el mensaje y la proyección eran nulos. Dicha función se unió a un nodo *mongodb*, el cual se encargaba de extraer los datos de la colección.

Tras el *mongodb-node* se colocaron funciones, cuyo propósito era extraer el valor datetime, así como el valor clave que se iba a utilizar en la gráfica (temperatura, humedad, PM2.5 y PM10). Asimismo, se pusieron en el eje X y el eje Y de tal manera de que un nodo chart de *dashboard* pudiera interpretarlo.

Para finalizar se unió la función a un chart node para poder visualizar los datos en una gráfica en el *dashboard*.

Tras comprobar su correcto funcionamiento, se apreció que una vez las gráficas se desplegaban dejaban de coger datos de la base de datos, Para ello se usó un *botton-node* de la paleta de *dashboard* para que se actualizarán los valores de las gráficas cuando se pulsara. Dicho nodo estaba unido a la función "enviar datos".

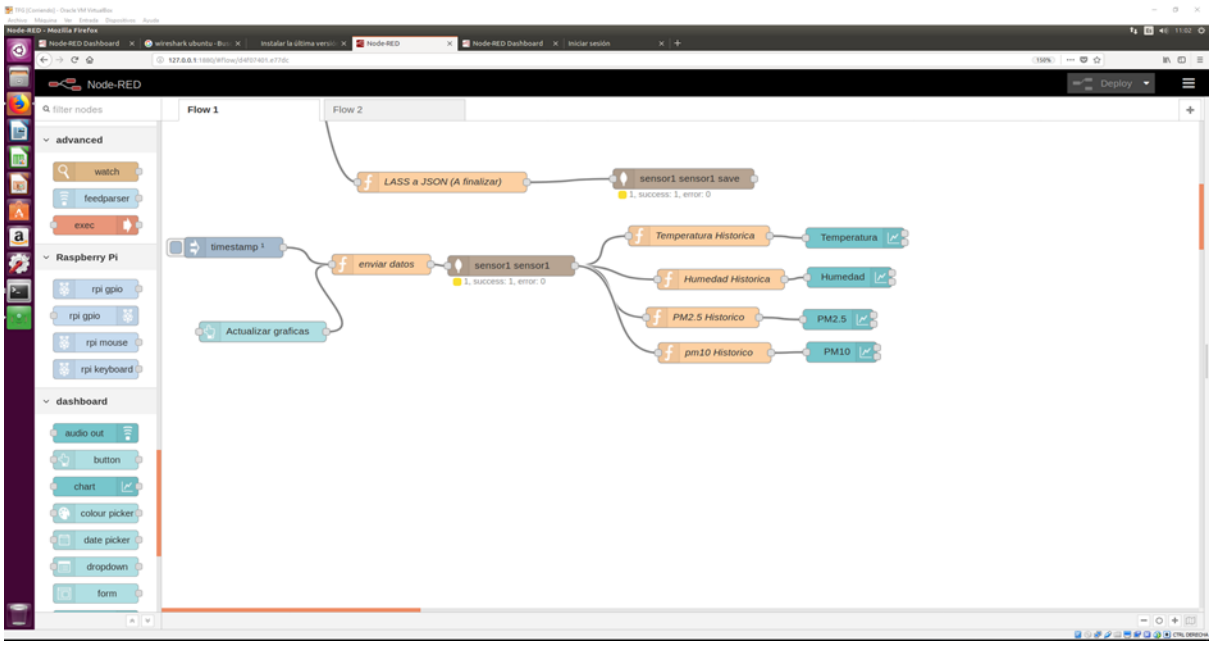

Figura 23. Resultado en node-red del apartado de Datos Históricos

3.5. COMPROBACIÓN DE LA ESCALABILIDAD

Para comprobar la escalabilidad, se hizo uso de un segundo sensor al cual se denominó “sensor2” y de un sensor virtual al cual se denominó “virtual1”.

Para el “sensor2”, la implementación del sensor fue la misma del “sensor1”. En cuanto al sistema de monitorización, partiendo de la base dejada por el anterior sensor, solo se tuvo que añadir un nodo de identificación del ID, en el cual, el único cambio era el propio ID. Además, en el apartado del *dashboard* histórico se añadió una colección denominada “sensor2” a la base de datos “sensor1” y se cambiaron los nodos *mongodb*. Asimismo, se añadió un nuevo etiquetado(“sensor2”) y grupos del *dashboard*; el resto de nodos se hicieron de la misma manera que en el apartado anterior.

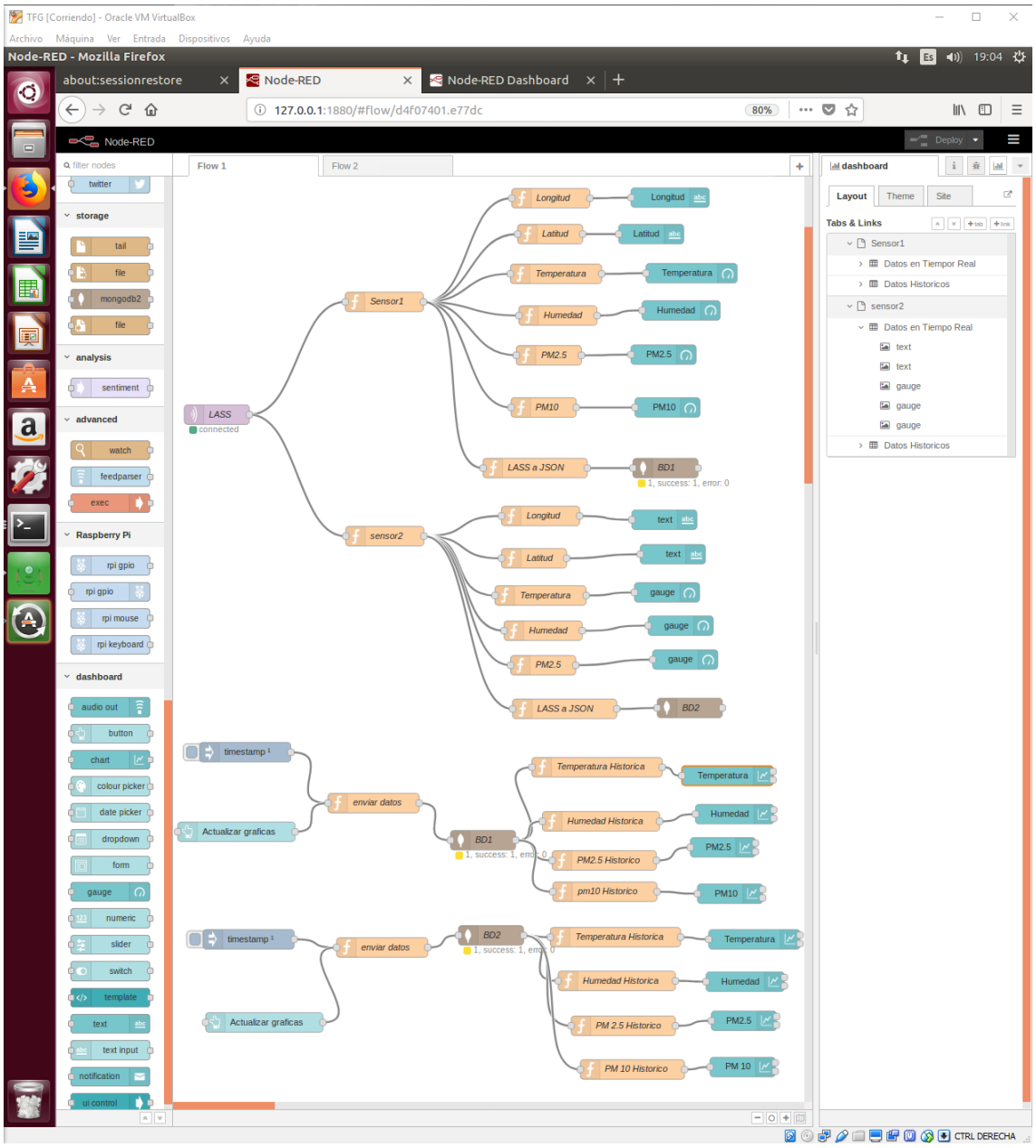

Figura 24. Resultado en node-red añadiendo el dashboard correspondiente "sensor2"

Para el sensor "virtual1" se comenzó por poner un nodo *inject* cada 1 segundo unido a un nodo *HTTP-Request* a la URL http://datos.santander.es/api/datos/sensores_smart_env_monitoring/279.json, lo que provocaba que cada 1 segundo se hiciera una petición a la URL para coger los datos de la misma. Después se convertía el JSON a un objeto JSON con el nodo JSON.

El siguiente paso era formar una función, la cual se encargaría de seleccionar los datos concretos dentro de la URL, añadir los que faltaban seleccionando un numero aleatorio entre un intervalo y añadir todos los valores utilizando el formato LASS. Dicha función se llamó "virtual1".

Para finalizar el sensor virtual se unió la función a un nodo MQTT cuyo *topic* se puso el común a ambos sensores. Tanto la monitorización como el *dashboard* se hicieron como en el caso del "sensor2".

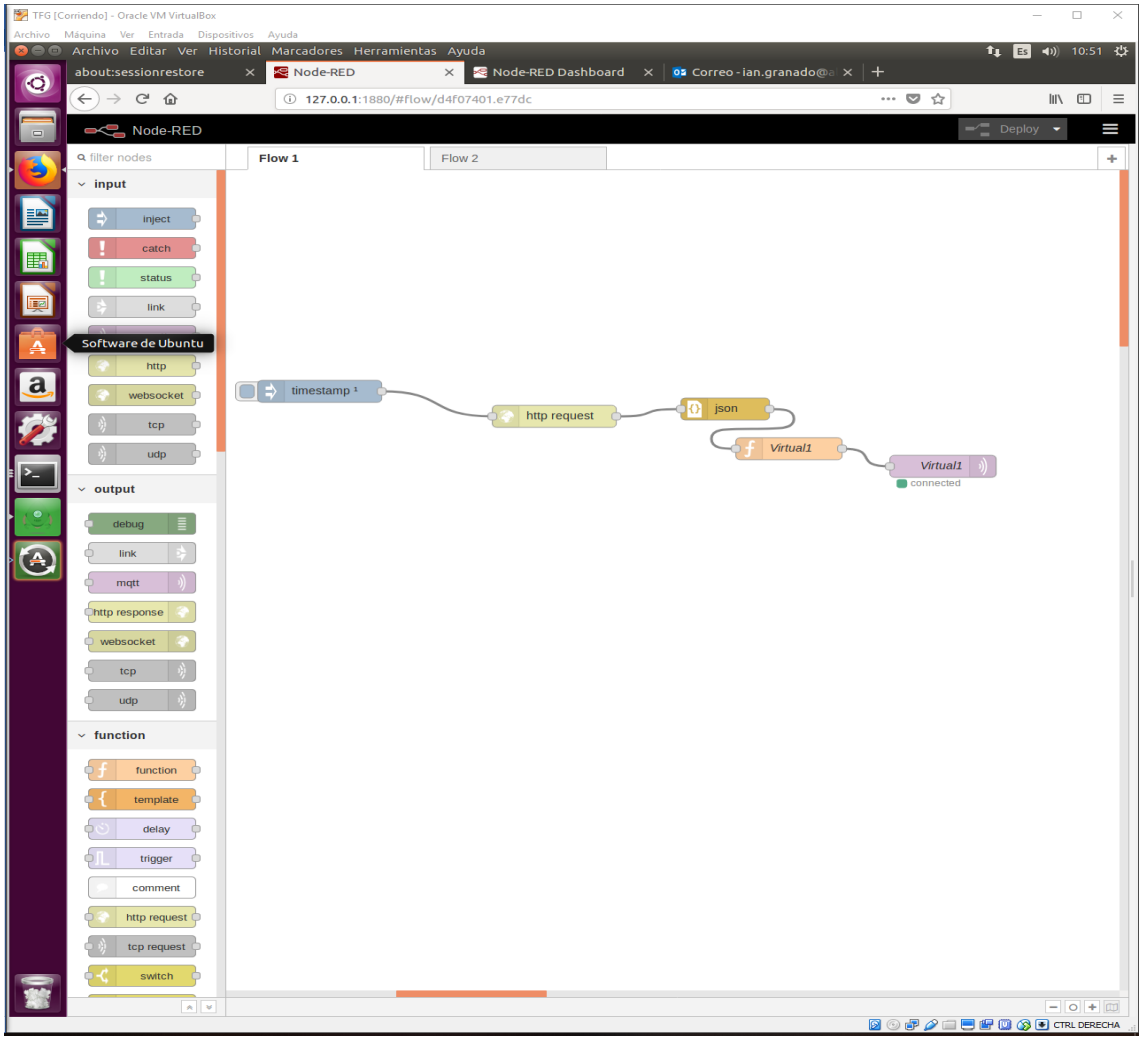

Figura 25. Visualización del conjunto de nodos que forman virtual1 en node-red

Capítulo 4: Integración en SmartSantander

En este apartado, el principal objetivo consistió en integrar las observaciones que se recibían de los sensores de humedad, temperatura, $PM_{2.5}$ y PM_{10} a la plataforma de SmartSantander. En este sentido, esta plataforma integra una gran cantidad de sensores repartidos por toda la ciudad y resulta mucho más conveniente para poder establecer correlaciones en aspectos de calidad medioambiental. Esta tarea se abordó en dos fases bien diferenciadas. En primer lugar, fue necesario estudiar y comprender los interfaces y los modelos de datos utilizados por la plataforma de SmartSantander. Estos interfaces son completamente distintos y también la codificación de la información es diferente. Una vez adquirido el conocimiento sobre cómo y con qué formato inyectar la información en la plataforma de SmartSantander, se pasó a la segunda fase que consistió

en transformar los datos tal y como se recibían de los sensores al modelo de datos utilizado en SmartSantander para después inyectarlos en sus servidores.

4.1. TRANSFORMACIÓN DE LOS MODELOS DE DATOS

Se comenzó por examinar los formatos antes de la susodicha transformación.
El formato LASS consiste en un string de atributos y valores. Cada atributo se encuentra relacionado con un valor al cual se une colocando entre ambos un "=". Cada uno de los conjuntos de atributo y valor se encuentra separado del resto por una línea vertical ("|"). En el siguiente ejemplo se puede apreciar un ejemplo de dicho formato:

```
|ver_format=3|FAKE_GPS=0|app=PM25|ver_app=0.8.3|device_id=sensor1|tick
=41182|
date=2080-01-
05|time=23:59:47|device=LinkItONE|s_0=0.00|s_1=100.00|s_2=1.0||s_3=0.0
0|s_4=1.00|
s_d0=11.00|s_t0=0.00|s_h0=0.00|s_d1=12.00|gps_lat=43.215423|gps_lon=3.
136075|gps_fix=0|gps_num=0|gps_alt=47
```

El modelo de datos de SmarSantander consiste en un objeto JSON que, a su vez, contiene una serie de objetos que se detallan a continuación:

- La "URN" vincula los datos obtenidos de la observación con el recurso que los ha proporcionado. En caso de que los datos provengan de un recurso no identificado o erróneo, automáticamente se descarta. Una URN se puede dividir en varios subdominios.
- El "timestamp", por lo general es producido por el dispositivo. En caso de no ser generado por el dispositivo, se genera automáticamente una vez que los datos acceden a la plataforma central de SmartSantander. El formato que utiliza es ISO 8601, una ventaja que presenta consiste en que su notación facilita la migración entre distintas plataformas.
- La "location" debe contener el formato GeoJSON. Es posible incluir recursos que no hagan referencia a un punto único especifico en el mapa.
- Dentro de "measurements" se recogen los diferentes tipos de observaciones, las cuales se denominan "phenomenon", asi como la unidad de medida "uom" y el valor del mismo "value". Los "phenomenon"y "uom" hacen uso de un

diccionario unificado que describe las diferentes clases medidas por el sensor. En caso de aparecer un nuevo fenómeno o unidad de medidad, para su inclusión es necesario que el administrador la acepte, tras lo cual será posible recoger los siguientes datos. El "value" se refiere al valor proveniente del sensor real, pero puede hacer referencia a la medición en bruto o a un procesamiento posterior, sin embargo, siempre se encuentra definido por el "uom"

Posteriormente se puede apreciar un ejemplo de dicho modelo:

```
{
      "urn": "urn:x-iot:testing:ian-granado:sensor",
      "observations":[
            {
                  "urn": "urn:x-iot:testing:ian-granado:sensor",
                  "timestamp": s,
                  "location":
      {

      "coordinates:[parseFloat(val1),parseFloat(val2)]
      "type":"Point"
      },
      "measurements":
      [{
      "phenomenon":"temperatura:ambient",
      "value": parseFloat(val3),
      "uom":"degreeCelsius"
      },{
      "phenomenon":"relativeHumidity",
      "value": parseFloat(val4),
      "uom":"percent"
      },{
      "phenomenon":"chemicalAgentAtmosphericConcentration:PM2.5",
      "value": parseFloat(val5),
      "uom":"microgramPerCubicMetre"
      },{
      "phenomenon":"chemicalAgentAtmosphericConcentration:PM10",
      "value": parseFloat(val6),
      "uom":" microgramPerCubicMetre"
}]}]}
```

Tras examinar los modelos, se procedió realizar la transformación, la cual se haría con *node-red*. Para ello, se hizo uso de una "función". Dentro de dicha función, se empezó por separar los atributos y valores que se iban a utilizar del resto, los cuales se encontraban en el formato LASS. Se separaron utilizando un puntero antes del atributo

y otro puntero después del valor. Posteriormente se cogió la zona entre ambos punteros y se reemplazó el atributo y el igual por un carácter nulo, dando como resultado el valor. Además de los valores provenientes de LASS, era necesario añadir un parámetro de tiempo; así pues, se empleó el tiempo del ordenador y tras seleccionar la fecha y hora se tuvo que pasar al formato ISO 8061.

Para terminar la transformación se procedió a colocar los valores y atributos de la manera en la que fuera comprensible para la plataforma SmartSantander, teniendo una URN propia y conociendo tanto los fenómenos como las unidades de medidas que acepta la plataforma.

En este apartado antes de enviar los datos a SmartSantander, se utilizó como paso previo la URL https://sensor.getsandbox.com para comprobar la correcta transformación de los datos. Este servicio se basa en un entorno en la nube que permite la generación de APIs para pruebas. En este caso, necesitábamos un *endpoint* que recibiera las peticiones HTTP POST que se generarían desde nuestro sistema. Este *endpoint*, emulaba el interfaz funcional con el que cuenta la plataforma SmartSantander por lo que comprobando que el sistema funcionaba correctamente en el entorno emulado, el siguiente paso sería el sustituirlo por la plataforma real de SmartSantander.

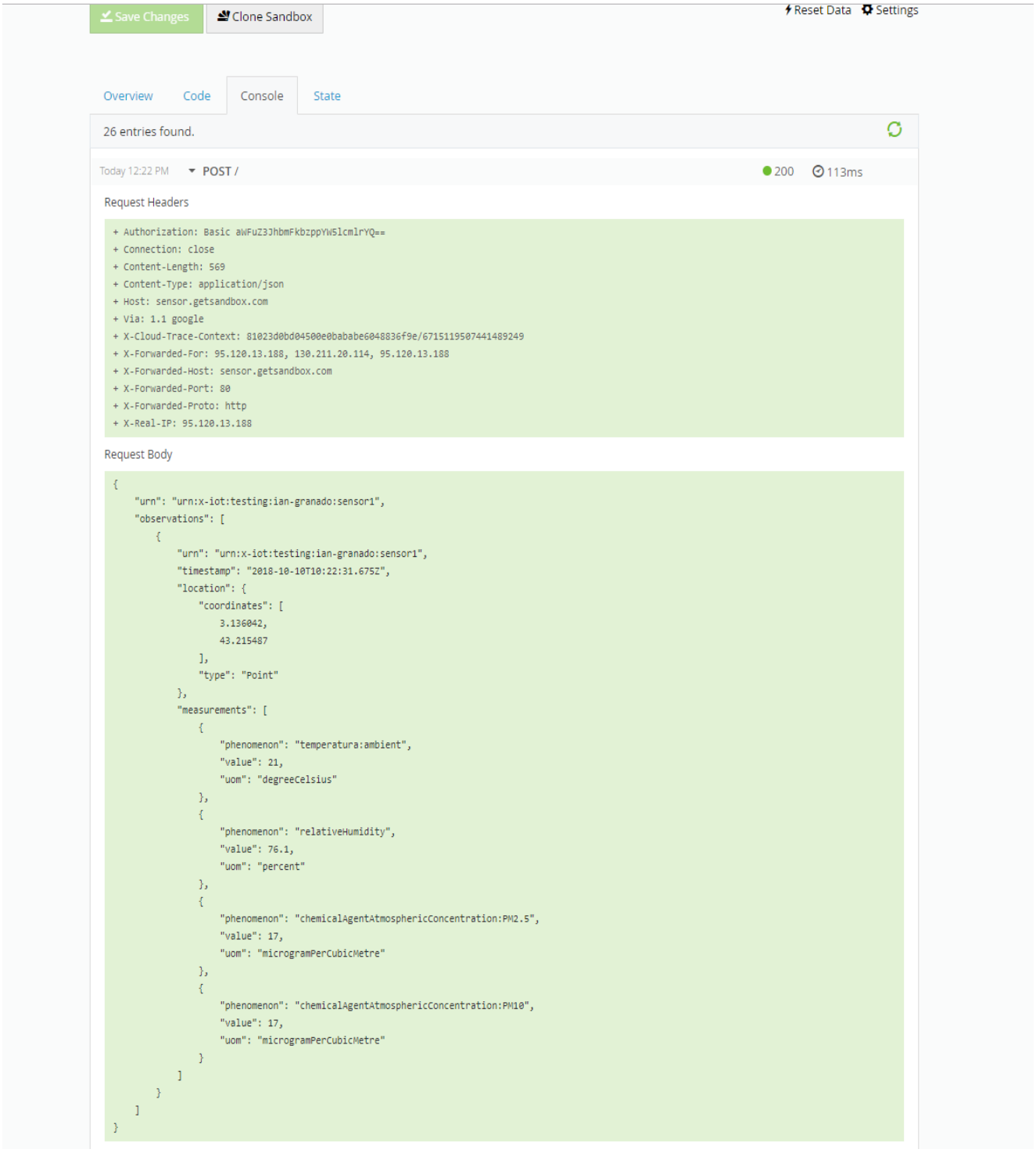

Figura 26. Formato SmartSantander en getsandox.com.

4.2. INTEGRACIÓN EN EL API SMARTSANTANDER

Debido a que la plataforma SmartSantander hace uso de una API REST, la cual es una interfaz entre sistemas http que se usa para obtener datos o generar operaciones en una gran variedad de datos (JSON incluido), se hizo uso de un *HTTP-Request-node*.

Dicho nodo utilizaba el método "POST", así como una URL propia a la que se enviaban los datos. Asimismo, se utilizaba una seguridad basada en un usuario y contraseña. Además, debido a problemas del servidor con el certificado se tuvo que ir quitar la opción en la que el servidor verificaba los certificados para evitar el error. Tras esto el servidor respondía con un 200 lo que significaba que los datos se habían recibido correctamente.

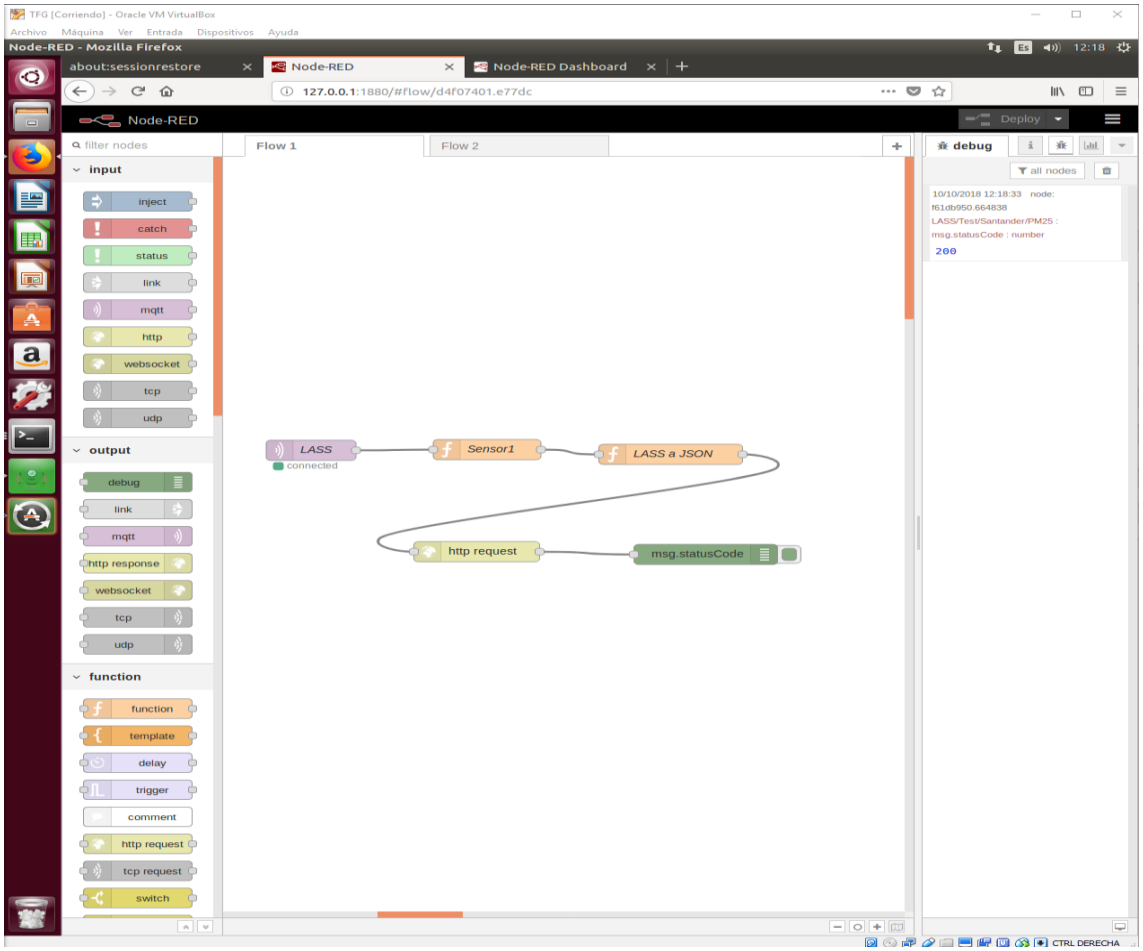

Figura 27. Resultado final de la integración en Node-Red.

Capítulo 5: Conclusiones y Líneas Futuras

En este apartado se analiza la consecución de los objetivos que se plantearon al comenzar este TFG. Además, se hace un resumen somero de las posibles tareas que se podrían abordar para extender el trabajo realizado. Estas tareas bien podrían convertirse en nuevos TFG.

5.1. CONCLUSIONES

Como principal conclusión es importante destacar que se han cumplido con los principales objetivos que se plantearon al comenzar este TFG. Se ha llevado a cabo el ensamblaje del hardware de una placa sensora con capacidad de monitorización de uno

de los principales factores de contaminación atmosférica que afecta a nuestras ciudades, las partículas micrométricas.

Asimismo, se ha adaptado y configurado el software que se ejecuta en dicho sensor. En este sentido, se ha integrado un sistema basado en el protocolo MQTT. La elección de este protocolo se debe a que sus características son muy adecuadas para entornos de redes de sensores como el que subyace en este TFG. Para ello, se ha instalado y configurado toda la arquitectura, incluyendo el sensor MQTT y las correspondientes bases de datos, que permite no solo la recepción de las observaciones generadas por los sensores ensamblados sino su almacenamiento para su posterior procesado.

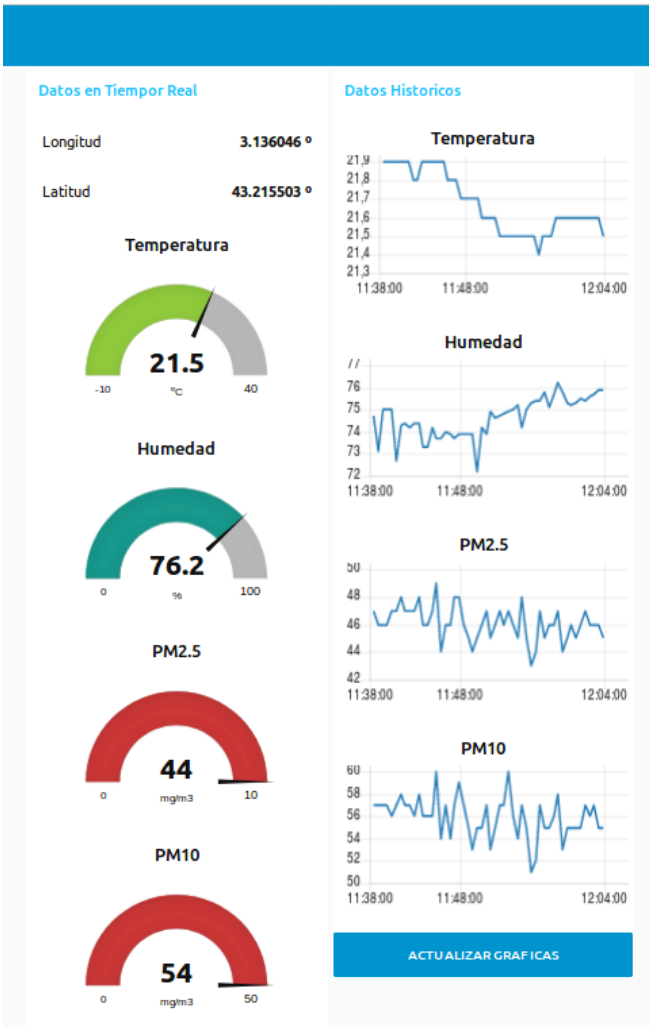

Figura 28. Dashboard operativo

En este sentido, otro de los objetivos alcanzados ha sido, precisamente, la implementación de un panel de control, o *dashboard*, en el cual se puede hacer un seguimiento tanto de los últimos valores observados (pseudo tiempo real) como del

histórico de observaciones que se han recibido de los distintos sensores. Para probar la escalabilidad del sistema implementado, además de inyectar los valores de los dos sensores físicos ensamblados, se ha emulado la existencia de una serie de sensores virtuales para lo cual se ha accedido a las medidas disponibles en el portal de datos abiertos del Ayuntamiento de Santander.

Finalmente, con objeto de incrementar la usabilidad del desarrollo realizado en este TFG, se ha integrado todo el sistema en la plataforma de SmartSantander. Para ello, se ha generado un inyector que recoge los datos que llegan al servidor MQTT y después de adaptarlos al modelo de datos que maneja la plataforma SmartSantander, los envía a la misma.

5.2. LÍNEAS FUTURAS

Una forma de extender el alcance de este trabajo hubiera sido incrementar el número de variables ambientales a observar. El sensor que se ha ensamblado en este TFG pretende ser un dispositivo abierto de bajo coste y fácil montaje que cualquiera pudiera construir. El objetivo último es que su uso se extendiera y se creara una red de sensores pertenecientes a diferentes personas, administraciones, negocios, etc. y que, de manera desinteresada, contribuyeran a generar un conocimiento más detallado acerca de la calidad del aire en las ciudades. Incorporar otros sensores, como contaminación por gases como CO_2, NO_2 o O_3, radiación solar, o ruido, y adecuar tanto el hardware como el software utilizado permitiría que los mapas que se pudieran generar a partir de la información recogida pudieran ser más variados.

Asimismo, los objetivos que se perseguían en este TFG se reducían a generar una prueba de concepto del sistema a implementar. Por ello, el comportamiento del sistema en periodos largos de tiempo no se ha probado ni evaluado. En este sentido, una de las formas a través de la cual se podrían extender los objetivos de este TFG sería sería poner los sensores en un funcionamiento práctico durante periodos largos e integrar definitivamente una red extensa de sensores en la plataforma SmartSantander para que su información pudiera ser utilizada de manera real y efectiva en aplicaciones útiles a los ciudadanos e instituciones. De esta forma también se podría comprobar la durabilidad de los sensores, así como su eficiencia.

Referencias

[1] Informe anual de la ONU, 2014
https://esa.un.org/unpd/wup/Publications/Files/WUP2014-Report.pdf

[2] Contaminación del aire OMS http://www.who.int/es/news-room/detail/02-05-2018-9-out-of-10-people-worldwide-breathe-polluted-air-but-more-countries-are-taking-action

[3] Smart Cities: eficiencia energética y sostenibilidad en un mismo lugar https://top-ten.cl/article/smart-cities-eficiencia-energetica-y-sostenibilidad-en-un-mismo-lugar

[4] El agua en las smart cities, una apuesta de futuro
https://www.tecnoaqua.es/descargar_documento/Reportaje-agua-smart-cities-apuesta-futuro-tecnoaqua-es.pdf

[5] La contaminación acústica en las ciudades https://www.urbiotica.com/soluciones-inteligentes-3/monitorizacion-del-ruido/

[6] SmartSantander: The meeting point between Future Internet research and experimentation and the smart cities https://ieeexplore.ieee.org/abstract/document/6095264

[7] SmartSantander: Experimentation and service provision in the smart city https://ieeexplore.ieee.org/abstract/document/6618633

[8] LASS https://paper.dropbox.com/doc/LASS-in-English-fpY3pDI25lD4GqfJ7ztSG

[9] LASS https://github.com/LinkItONEDevGroup/LASS

[10] MQTT http://mqtt.org/

[11] MQTT http://www.tst-sistemas.es/mqtt/

[12] Aprende el protocolo MQTT paso a paso http://www.ermesh.com/aprender-protocolo-mqtt-parte-1/

[13] PRIMEROS PASOS CON MQTT https://ricveal.com/blog/primeros-pasos-mqtt/

[14] Node-Red https://nodered.org/about/

[15] LinkIt ONE http://wiki.seeedstudio.com/LinkIt_ONE/

[16] LinkIt™ ONE https://labs.mediatek.com/en/platform/linkit-one

[17] Plantower PMS3003 Air Quality Sensor http://aqicn.org/sensor/pms3003/es/

[18] Grove - Temperature&Humidity Sensor Pro https://www.seeedstudio.com/Grove-Temperature-Humidity-Sensor-Pro-AM230-p-838.html

[19] LASS Field Try #1: Device Instructions https://paper.dropbox.com/doc/LASS-Field-Try-1-Device-Instructions-tarjClSu2W54PvCIrkreh

[20] LASS.ino y configuration.h https://github.com/LinkItONEDevGroup/LASS/tree/master/Device_LinkItOne/LASS

[21] Cómo crear una máquina virtual con VirtualBox https://www.softzone.es/2014/10/30/como-crear-una-maquina-virtual-con-virtualbox-paso-paso/

[22] How to Connect Your Internet of Things with Node-RED https://www.digitalocean.com/community/tutorials/how-to-connect-your-internet-of-things-with-node-red-on-ubuntu-16-04

[23] Hola Mundo IoT https://sg.com.mx/revista/51/hola-mundo-iot

[24] Instalacion de Mosquitto Broker MQTT http://pdacontroles.com/instalacion-de-mosquitto-broker-mqtt-en/

[25] Getting Started with Node-RED Dashboard https://randomnerdtutorials.com/getting-started-with-node-red-dashboard/

[26] How to Install and Configure MongoDB
https://www.howtoforge.com/tutorial/install-mongodb-on-ubuntu/
[27] Power Prototyping with MongoDB and Node-RED
https://www.compose.com/articles/power-prototyping-with-mongodb-and-node-red-2/
[28] Instalando Robo3t http://benjagarrido.com/instalando-robomongo-ubuntu-16-04/

Anexo: Procedimiento de instalación del entorno de desarrollo

COMANDOS DE INSTALACIÓN NODE.JS

```
$ sudo apt install nodejs-legacy
$ sudo apt-get install –y nodejs
$ node –v
```

COMANDOS DE INSTALACIÓN NPM

```
$ sudo apt-get install npm
$ npm –v
```

COMANDOS DE INSTALACIÓN NODE-RED

```
$ sudo npm install –g –unsafe-perm node-red node-red-admin
$ sudo ufw allow 1880
```

COMANDOS DE INSTALACIÓN MONGODB

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --
recv 68818C72E52529D4
$ sudo echo "deb http://repo.mongodb.org/apt/ubuntu
bionic/mongodb-org/4.0 multiverse" | sudo tee
/etc/apt/sources.list.d/mongodb-org-4.0.list
$ sudo apt-get update
$ sudo apt-get install -y mongodb-org
$ sudo service mongod start
$ sudo systemctl status mongod
```

COMANDOS DE INSTALACIÓN DE DASHBOARD

```
##sudo apt install git
$ git clone https://github.com/node-red/node-red-dashboard.git
$ cd node-red-dashboard
$ npm install
```

COMANDOS DE INSTALACIÓN ROBO3T

```
$ cd Descargas
$ tar -xvzf robo3t-1.2.1-linux-x86_64-3e50a65.tar.gz
$ sudo mkdir /usr/local/bin/robomongo
$ sudo mv  robo3t-1.2.1-linux-x86_64-3e50a65/*
/usr/local/bin/robomongo
```

COMANDOS DE INSTALACIÓN SERVIDOR MQTT

```
$ sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa
$ sudo apt-get update
$ sudo apt-get install mosquitto
$ sudo apt-get install libmosquitto-dev
$ sudo apt-get install mosquitto-clients
$ sudo service mosquitto status
```