Monitorización de los niveles de polución ambiental mediante sensores móviles

TRABAJO FIN DE MÁSTER

Autor: Willian Jesús Zamora Mero
Director: Dr. Carlos Tavares Calafate

Septiembre de 2015

Departamento de Informática de Sistemas y Computadores
Grupo de Redes de Computadores

Agradecimientos

A mi tutor, Carlos Tavares Calafate, por ser quien propuso el tema y por haber sabido supervisar con éxito la culminación de este trabajo.

Al Gobierno Nacional del Ecuador a través del SENESCYT por la ayuda brindada a través de su programa de Becas, y a la Universidad Laica Eloy Alfaro de Manabí.

Dar gracias a mi familia, y en especial a mi hija Isabella, por saber comprender mi ausencia durante esta etapa de mi vida. Sin ella este trabajo aun estaría por empezar…

Resumen

Las zonas urbanas con alta densidad de población se caracterizan por unos niveles de contaminantes en el aire más elevados que en otras zonas, lo que es actualmente motivo de preocupación tanto para los ciudadanos como para las autoridades gubernamentales. En este proyecto se propone usar diferentes sensores ambientales para obtener mapas de la polución ambiental en Valencia. En el estudio se propone que los sensores sean instalados en un coche, moto o bicicleta para poder así cubrir el área a estudiar mediante una solución móvil, la cual, además de los sensores, incluye también un gateway basado en Raspberry Pi y una aplicación móvil para Android. La solución propuesta requiere también un tratamiento posterior de los datos para obtener mapas de contaminación de forma visual. Para eso se ha creado un servidor capaz de recibir los datos generados y procesarlos, ofreciendo una interfaz web capaz de facilitar información detallada sobre los niveles de polución detectados mediante diferentes tipos de gráficas incluyendo mapas de calor.

Abstract

Urban areas with a high population density are characterized by having higher pollutant levels in the air than in other areas, which has become one of the main concerns for both citizens and governmental authorities. In this project we propose using environment sensors to obtain environment pollution maps for the city of Valencia. The proposed sensors can be installed in a car, motorcycle or a bike to cover the studied area through a mobile solution; in addition to the sensor devices, it includes a Raspberry Pi based gateway and an Android application. The proposed solution also requires the treatment of gathered data to create visual contamination maps. With that purpose we created a server capable of receiving and processing the data obtained, offering a web interface able to provide detailed information about the levels of pollution detected through different types of graphics, including heat maps.

TABLA DE CONTENIDO

Contenido de Figuras

Contenido de Tablas

Capítulo I: Introducción

En los últimos años el crecimiento poblacional y económico de las ciudades, el desarrollo industrial y el aumento de vehículos en circulación ha provocado que se libere una gran cantidad de partículas que contaminan al aire. Diferentes estudios [1] [2] [3] señalan que ciudades con alta densidad de población se caracterizan por unos niveles de contaminantes en el aire más elevados que en otras zonas, lo que es actualmente motivo de preocupación tanto para los ciudadanos como las autoridades. En este sentido, destacar que niveles de contaminación ambiental considerados como moderados e inclusos bajos se asocian con efectos nocivos para la salud [1].

En relación con lo anterior, existen soluciones tecnológicas que ofrecen información en tiempo real sobre el estado de contaminación ambiental, pero estas soluciones normalmente pasan por ubicar dispositivos sensores en puntos estáticos muy específicos, no ofreciendo una cobertura completa y continua en el tiempo, por lo que la población en general desconoce la situación ambiental de su entorno.

Por otro lado, el avance de las tecnologías de la información y comunicación ha hecho que los dispositivos móviles tengan una demanda creciente, sea para uso personal o empresarial. Dispositivos móviles como smartphones y tabletas, que a día de hoy tienen elevadas capacidades de cómputo, apoyan el día a día de las personas, y tareas como agendar una cita, realizar una videoconferencia o obtener geolocalización son habituales para estos dispositivos. Su bajo coste y el aumento de sensores instalados, como WiFi, Bluetooth, NFC, GSM, WCDMA, GPS, etc., sirven de incentivo para que algunas aplicaciones sean desarrolladas con fines específicos.

Este Trabajo Fin de Máster, se centra en unificar tecnologías móviles con los sensores ambientales existentes, de tal forma que se pueda mostrar información de contaminación en tiempo real y en un punto determinado, por lo que la solución obtenida debe generar mapas de niveles contaminación.

La propuesta es el desarrollo de un sistema monitorización ambiental con sensores móviles. Para ello se hace uso de un sensor ambiental de la empresa Libelium y una Raspberry Pi que se integra en una arquitectura de servicios cliente-servidor a través de un dispositivo móvil como smartphones o tabletas con sistema operativo Android.

1.1 Objetivos general y específicos

El objetivo general es desarrollar un sistema de monitorización medioambiental basado en sensores móviles. Este sistema permite a un teléfono o tableta móvil con sistema operativo Android transmitir en tiempo real los datos capturados por un sensor ambiental de la empresa Libelium usando un dispositivo Raspberry Pi como elemento de unión entre ambos. Estos datos serán procesados por dicho dispositivo móvil y mostrados gráficamente, además de ser enviados a un sistema web que tratará los datos recabados con el fin de obtener mapas de contaminación para la ciudad de forma visual. El esquema a desarrollar se muestra en la figura 1.1.

Para alcanzar el objetivo general se definen los siguientes objetivos específicos:

- Desarrollar una interfaz de comunicación que permita leer los valores de contaminación entre el dispositivo Raspberry Pi y el Sensor Libelium.
- Desarrollar un servicio que permita integrar la Raspberry Pi y el terminal Android a través de Bluetooth.
- Crear una interfaz gráfica en Android que permita leer y mostrar datos en tiempo real de los valores de contaminación.
- Representar valores de contaminación a través de gráficos y mapas en Android.
- Diseñar la aplicación web que procese la información obtenida y la presente mediante Mapas de Calor.

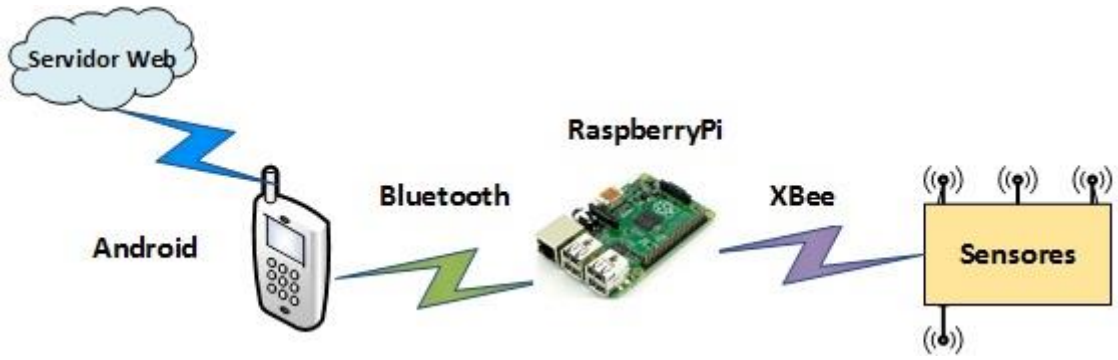


Figura 1. 1 .-Esquema a desarrollar.

1.2 Motivación

La elección de un sistema móvil, nació al cursar las asignaturas “Sistemas basado en Redes Móviles” y “Sistemas Distribuidos y Empotrados”, ambas integrantes del plan de estudios del Máster en Ingeniería de Computadores y Redes. En ambas asignaturas se hicieron prácticas que enriquecieron mis expectativas sobre esta área de conocimiento. En relación con lo anterior, se propusieron temas por parte de los docentes de este Máster, y fue en el Grupo de Investigación en Redes de Computadores que centré mis expectativas futuras para este proceso investigativo, como es el Trabajo Fin de Máster, y al inicio de mi estudio doctoral.

Sobre esa base, la idea de unificar diferentes tecnologías como Raspberry Pi, Sensor Libelium y Android, con el fin de obtener y evaluar los niveles de contaminación medio ambientales en la ciudad de Valencia, fundamentaron a esta decisión.

Otro punto de interés es que este tema es una investigación de campo, con evaluación en sitio y adicionalmente sirve como medio de concienciación al ciudadano sobre el daño que se está causando al medio ambiente, haciéndolo responsable de su ciudad.

1.3 Estructura del documento

Capítulo I - Introducción: En este capítulo se describe el contexto en el cual se desarrolla este Trabajo Fin de Máster. En él se describe la problemática, los objetivos generales y los objetivos específicos a seguir. Finalmente se presenta la motivación del autor con respecto al tema.

Capítulo II - Antecedentes: En este capítulo se analiza el objeto de estudio, se identifica lo que se conoce del tema y sus efectos, y se finaliza presentando los estudios más relevantes realizados hasta la fecha.

Capítulo III - Tecnología empleada: En este capítulo se describen los dos principales componentes tecnológicos que forman parte de la solución planteada: el sistema móvil y el sistema web. Para cada uno se ha hecho una descripción de su tecnología. El capítulo finaliza con la tecnología empleada para la comunicación entre dispositivos.

Capítulo IV - Análisis del sistema: En este capítulo se describen los requisitos funcionales y no funcionales del sistema propuesto; se utilizan diagramas casos de usos para definir sus principales características.

Capítulo V - Diseño del Sistema: En este capítulo define la arquitectura en la que se basa la implementación, y adicionalmente muestra la interfaz del sistema propuesto.

Capítulo VI - Pruebas: En este capítulo se muestran las pruebas realizadas a la solución desarrollada, evidenciando los resultados obtenidos en la aplicación móvil y en el sistema web.

Capítulo VII - Conclusiones: En este capítulo se describen los principales logros alcanzados y los trabajos futuros.

Capítulo II: Antecedentes

En este capítulo se hace una revisión bibliográfica de los trabajos más afines a este Trabajo Fin de Máster. Se empieza por describir algunos contaminantes atmosféricos y sus efectos en la salud humana. A continuación se contextualiza la información para la ciudad de Valencia, visualizando algunos de sus datos atmosféricos. Adicionalmente se describe un índice de calidad del aire para uno de los contaminantes más relevantes para la salud, como es el ozono. Finalmente, se expondrán algunos trabajos tecnológicos realizados en este contexto de estudio.

2.1 Contaminación Atmosférica

En la actualidad nuestro planeta sufre las consecuencias de la contaminación atmosférica, y el exceso de gases contaminantes provoca cambios en las condiciones climáticas de los diferentes continentes. En este contexto, la disminución de la capa de ozono es uno de los factores que más perjudican a la naturaleza y los seres humanos. De hecho, diferentes trabajos [1] [2] [3] señalan al cambio climático como la mayor amenaza de este siglo para la salud a nivel global. Se entiende por contaminación atmosférica toda alteración producida por las actividades humanas, y por el crecimiento industrial en ausencia de políticas ambientales, incluyendo gases emitidos por combustibles fósiles, desechos químicos, calefacción doméstica, entre otros.

Este cambio climático ha obligado a que la mayor parte de países establezcan normas y regulaciones para minimizar el impacto del hombre en el medio ambiente. Uno de ellos es el Protocolo de Kioto [4], cuyo objetivo es reducir las emisiones de seis gases de efecto invernadero que causan el calentamiento global: dióxido de carbono (CO_2), gas metano (CH_4) y óxido nitroso (N_2O), y otros tres gases industriales fluorados: hidrofluorocarburos (HFC), perfluorocarbonos (PFC) y hexafluoruro de azufre (SF6).

En Europa, desde el siglo pasado se ha establecido un marco legislativo [5] relativo a los problemas de contaminación ambiental. SOER [6], en su informe de marzo del 2015, señala que este marco legislativo es el más exhaustivo del mundo. En el mismo informe describe la reducción de la contaminación como una mejora sustancial en la calidad de las aguas y el aire de Europa. Por contra, la pérdida de funciones edáficas, la degradación del suelo y el cambio climático constituyen problemas de primer orden que ponen en riesgo los flujos de bienes y servicios medioambientales que sustentan la producción económica y el bienestar europeos.

Así mismo, en Europa se han creado organismos que apoyan estas iniciativas, como:

- AEMA Agencias Europea de Medio Ambiente: ofrece información sólida e independiente sobre el medio ambiente, y actualmente tiene 33 países miembros. Su sitio web es: http://www.eea.europa.eu/es/about-us/who.

- CITEAIR (información Común sobre el aire en Europa): proyecto co-financiado por los Programas INTERREG IIIC[1] y IVC de la unión Europa que empezó en marzo del 2004 y otras agencias. El sitio web http://www.airqualitynow.eu/es/about_home.php muestra la calidad del aire en Europa.

En España se han realizado estudios en el centro Español de Contaminación Atmosférica y Salud (EMECAS) donde se constató el efecto a corto plazo de la contaminación atmosférica sobre la mortalidad. Adicionalmente existen iniciativas como el Sistema CALIOPE[2] del departamento de Ciencias de la Tierra del Barcelona – Centro Nacional de Supercomputación, que tiene como uno de los temas principales de investigación los modelos de calidad del aire.

En la provincia de Valencia, cuya extensión es de 10.763 km^2, se encuentra La Red Valenciana de Vigilancia y Control de la Contaminación Atmosférica (RVVCCA), creado bajo Decreto 161/2003. Esta red hace un seguimiento e informa al público de los contaminantes atmosféricos en las principales áreas urbanas e industriales de la comunidad Valenciana. En la figura 2.1 se observan los 51 puntos de control de esta red.

Este proyecto se centra en la ciudad de Valencia, cuya extensión territorial es de 134,6 Km^2, y según el Instituto Nacional de Estadística (2014) tiene una población de 786.424 personas. RVVCCA señala que tiene 8 puntos de control para la ciudad [7].

[1] INTERREG IVC es una extensión del IIIC que proporciona fondos para la cooperación **interregional en Europa objetivo Comunidad Europea** y financiado a través del Fondo Europeo de Desarrollo Regional (FEDER).
[2] http://www.bsc.es/caliope/es/equipo

Figura 2. 1.-Puntos de Control RVVCCA.

2.1.1 Principales contaminantes

Según SOER [6] la calidad del aire ha mejorado considerablemente en los últimos tiempos. Sin embargo la actual calidad del aire todavía afecta a la salud de la población [8], y en muchas ciudades europeas, la calidad de aire es una preocupación, siendo por lo tanto supervisada continuamente.

La circulación vehicular urbana es una de las principales fuentes de contaminación atmosférica, y el informe del Plan Nacional de Mejora de la Calidad del Aire [9] concuerda con esta aseveración indicando que el tráfico urbano genera en las ciudades españolas la mitad de la contaminación por partículas de suspensión, y por lo tanto un gran impacto en la salud. Ante este problema se han creado puntos estacionarios de monitorización ambiental para diferentes contaminantes. Entre los contaminantes principales tenemos:

- **Material particulado (PM10/2.5**).- El material particulado varía extensamente en su composición física y química, fuente y el tamaño de partícula. Se define PM10 como la fracción de las partículas en aire de tamaño inferior a 10 µm, y partículas PM2.5 las de

tamaño menor a 2.5 µm. Ambas son de gran interés, ya que debido a su pequeño tamaño pueden penetrar profundamente en los pulmones, con potencial peligro para la salud. Destacar que no son fáciles de inhalar las partículas mayores, que además son retiradas de manera relativamente eficiente del aire por la sedimentación. La fuente principal de emisiones en las ciudades europeas es el tráfico rodado, en particular por los vehículos diésel. Destacar también que los valores límites definidos son sobrepasados en las ciudades europeas con bastante frecuencia.

- **Óxidos de nitrógeno (NO_x).-** Los óxidos de nitrógeno (NO_x) describen una mezcla de óxido nítrico (NO) y dióxido de nitrógeno (NO_2). Son gases inorgánicos formados por la combinación de oxígeno con el nitrógeno del aire. El NO es producido en cantidades mucho mayores que el NO_2, pero se oxida a NO_2 en la atmósfera. El NO_2 causa efectos perjudiciales en los bronquios. Las concentraciones de dióxido de nitrógeno con frecuencia se acercan, y por veces superan, los niveles límite de calidad del aire en muchas ciudades europeas. Los NO_x son emitidos cuando el combustible está siendo quemado, p.ej. en el transporte, procesos industriales y generación de energía eléctrica.

- **Ozono (O_3).-** El ozono troposférico (a nivel terrestre), a diferencia de otros contaminantes, no es emitido directamente a la atmósfera, sino que es un contaminante secundario producido por la reacción entre el dióxido de nitrógeno (NO_2), los hidrocarburos y la luz solar. La luz solar proporciona la energía de activación de la formación de ozono; por consiguiente, los niveles más altos se tienen lugar fundamentalmente en verano, y en las horas centrales del día.

- **Hidrocarburos (HC) y compuestos orgánicos volátiles (VOC).-** Los HC pertenecen a un grupo más grande de sustancias químicas conocidas como compuestos orgánicos volátiles (VOC). Los HC son compuestos exclusivamente de hidrógeno y carbón, mientras que los VOC pueden contener otros elementos. Se producen por la combustión incompleta de hidrocarburos, y también por su evaporación. Al existir cientos de compuestos diferentes, los HC y VOC muestran una amplia gama de propiedades. Unos, como el benceno, son cancerígenos; algunos son tóxicos, mientras que otros son inofensivos para la salud.

- **Dióxido de azufre (SO_2).-** Los combustibles fósiles contienen rastros de compuestos de azufre, produciéndose SO_2 cuando estos combustibles son quemados. La mayoría del SO_2 emitido al aire proviene de la generación de energía eléctrica. La contribución de las

fuentes de transporte es pequeña (exceptuando el transporte marítimo). La exposición al SO_2 puede dañar la salud por su acción sobre los bronquios. El ácido sulfúrico generado en las reacciones atmosféricas de SO_2 es el componente principal de la lluvia ácida. Las partículas de sulfato de amonio son las partículas secundarias más abundantes encontradas en el aire.

- **Monóxido de Carbono (CO).-** Es un gas inodoro, insípido e incoloro producido por la combustión incompleta de materiales que contienen carbón, incluyendo la mayor parte de los combustibles empleados en el transporte. El CO es tóxico, actuando por reacción con la hemoglobina, reduciendo su capacidad para el transporte de oxígeno en la sangre. Incluso en grandes centros urbanos, las concentraciones de CO raras veces exceden los límites establecidos.

Por otro lado, la Agencia de Protección Ambiental (EPA) de los Estados Unidos, en sus estudios sobre el ozono, ofrece un índice de calidad del aire [10], definiendo un rango de valores de bueno a peligroso. Este índice asigna colores para estos rangos.

Otro índice de calidad del aire es el de la Unión Europea [11], el cual clasifica a los contaminantes atmosférico en 5 niveles: Muy bajo, bajo, medio, alto y muy alto. En este índice, valores muy altos reflejan niveles de alerta para el ser humano. El sitio web de Contaminación del aire de Europa muestra una clasificación en función de la protección de salud [12] (ver figura 2.2).

Índice de Calidad del Aire (ICA)	Calidad del Aire	Proteja su Salud
0 - 50	Buena	No se anticipan impactos a la salud cuando la calidad del aire se encuentra en este intervalo.
51 -100	Moderada	Las personas extraordinariamente sensitivas deben considerar limitación de los esfuerzos físicos excesivos y prolongados al aire libre.
101-150	Dañina a la Salud de los Grupos Sensitivos	Los niños y adultos activos, y personas con enfermedades respiratorias tales como el asma, deben evitar los esfuerzos físicos excesivos y prolongados al aire libre.
151-200	Dañina a la Salud	Los niños y adultos activos, y personas con enfermedades respiratorias tales como el asma, deben evitar los esfuerzos excesivos prolongados al aire libre; las demás personas, especialmente los niños, deben limitar los esfuerzos físicos excesivos y prolongados al aire libre.
201-300	Muy Dañina a la Salud	Los niños y adultos activos, y personas con enfermedades respiratorias tales como el asma, deben evitar todos los esfuerzos excesivos al aire libre; las demás personas, especialmente los niños, deben limitar los esfuerzos físicos excesivos al aire libre.
300+	Arriesgado	

Figura 2. 2.-Índice de calidad del Aire y efectos en la Salud

2.1.2 Efectos sobre la salud

El sitio web air quality [13] señala ciertas enfermedades que pueden ser provocadas por la mala calidad del aire exterior.

- **Partículas (PM10, PM2.5).-** Las partículas finas pueden penetrar profundamente en los pulmones, pudiendo causar inflamación y un empeoramiento de los síntomas en pacientes con enfermedades de corazón y pulmonares. Además, su efecto se agrava si incluyen compuestos compuestos cancerígenos que puedan ser adsorbidos en la superficie de los pulmones.

- **Ozono (O_3).-** El Ozono irrita las vías aéreas de los pulmones, aumentando los síntomas del asma y de las enfermedades pulmonares.

- **Monóxido de carbono (CO)**.- Este gas impide el transporte normal de oxígeno por la sangre. Esto puede conducir a una reducción significativa del suministro de oxígeno al corazón, en particular en pacientes que sufren problemas cardíacos.

- **Plomo y metales pesados**.- Incluso pequeñas cantidades de plomo pueden ser perjudiciales, sobre todo para los niños. Además, el plomo inhalado por la madre puede interferir con la salud del niño. La exposición también ha sido vinculada a interferencias en la función mental, la función visual, la memoria, la capacidad de atención y el daño neurológico en niños.

- **Microagentes contaminantes tóxicos orgánicos**.- Son producidos por la combustión incompleta de combustibles o residuos. Comprenden una gama compleja de sustancias químicas, algunas de las cuales son emitidas en cantidades muy pequeñas. Son sumamente tóxicas o cancerígenas. Los compuestos en esta categoría incluyen: PAHs (Hidrocarburos Aromáticos Policíclicos), PCBs (bifenilos policlorados), dioxinas, etc. Pueden causar una amplia gama de efectos, como cáncer, reducción de la inmunidad, trastornos del sistema nervioso e interferencias con el desarrollo infantil. No hay ninguna dosis "umbral" ya que hasta cantidades reducidas pueden causar daño.

2.1.3 Estaciones de monitorización ambiental terrestres

Las estaciones de monitorización ambiental terrestres son espacios físicos ubicados en puntos estratégicos, destinados a medir la contaminación ambiental y la calidad del aire de su entorno.

Por lo general, la mayoría de estaciones están integradas con otras, formando una red de monitorización de ámbito local, nacional o continental. La mayoría de estaciones usan tecnologías con un sistema de calibración que minimiza los errores de lectura. Ejemplos de estaciones terrestres se muestran en la figura 2.3 (ubicación) y 2.4 (estación física).

Figura 2. 3.-Puntos de estaciones terrestres en Valencia. Fuente CALIOPE.

Figura 2. 4.-Estación meteorológica y de contaminación atmosférica de la Universidad Politécnica de Valencia.

Estas estaciones son muy costosas y ocupan un espacio significativo para su instalación, y los equipos por lo general son sensores de gama alta ya que los sensores de gama baja y bajo coste tienen una gran fluctuación entre mediciones, por lo que su uso, no es frecuente para estas instalaciones. Las estaciones de medioambientes terrestres obtienen mediciones y sus valores por lo general se extrapolan para zonas no muestreadas [14].

2.2 Estudios realizados

Se ha observado que la contaminación atmosférica es una preocupación global. De igual forma, se identificaron algunos de los contaminantes atmosféricos, y sus posibles consecuencias para la salud humana. Dentro del contexto Europa/España, se han consultado bases de datos de instituciones preocupadas por la contaminación atmosférica y la calidad del aire de la región, y finalmente se constató que la información emitida es extraída de las estaciones de monitorización terrestre indicadas anteriormente.

Sobre esa base, y con el fin de minimizar los costes y obtener valores de contaminación en puntos específicos, se han hecho estudios alternativos [15] y [16] donde se ubica un sensor móvil de bajo coste en autobuses (ver figura 2.5). En el primer caso, se mide el ozono con un buen grado de precisión, validando sus resultados con patrones de ozono espaciales, disponibles para otras ciudades. En el segundo caso se utilizan sensores de marca Waspmote para medir la contaminación ambiental en la ciudad de Belgrado, Serbia. Ambos estudios presentan mapas de contaminación, aunque ninguno de ellos ofrece una solución gráfica en tiempo real de datos de contaminación.

En [17] se utiliza un microcontrolador, varios sensores de contaminación del aire, un modem GPRS y un sistema de posicionamiento global (GPS) integrado en el móvil, que utiliza la red pública de telefonía para transmitir los datos al Servidor (ver figura 2.6).

En [18] (ver figura 2.7), se muestra una solución móvil conectada a un sensor mediante de una interfaz RS232. Esta aplicación presenta en tiempo real valores de contaminación de ozono y utiliza el sistema operativo Android.

Figura 2. 5.-Sensor Wasmote autobús Belgrado.

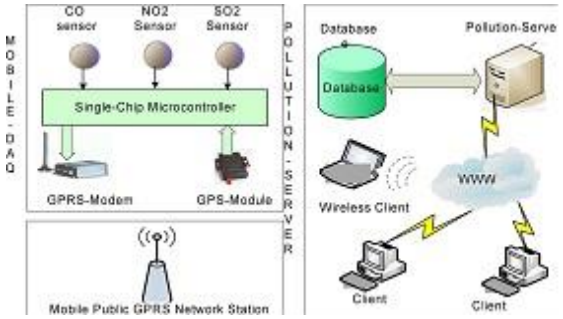


Figura 2. 6.- Monitorización a través de un Sensor Móvil GPRS.

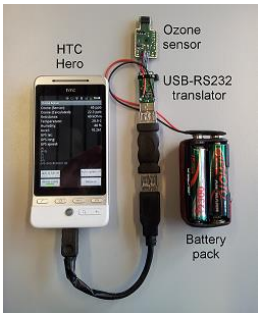


Figura 2. 7.-Monitorización por un Smartphone.

Adicionalmente, se han venido desarrollando algunas aplicaciones móviles que muestran información sobre la calidad del aire, así como otras aplicaciones que sirven de información educativa sobre la contaminación ambiental, entre otras. Ante esto se han hecho revisiones de algunas aplicaciones móviles de la plataforma Google Play. Estas aplicaciones están conectadas a una o varias estaciones terrestres, y en algunas de estas App la información presentada no siempre es en tiempo real. Además, ninguna de ellas toma datos directamente, usando todas ellas información proveniente de estaciones terrestres.

Aplicaciones móviles disponibles:

OzoneMap – Android

Proporciona un mapa de los niveles de ozono en tiempo real que permite a los residentes de la región de Houston, Texas, tomar decisiones informadas sobre la protección de su salud mediante la limitación de la exposición personal a niveles insalubres de ozono a nivel del suelo (ver figura 2.8).

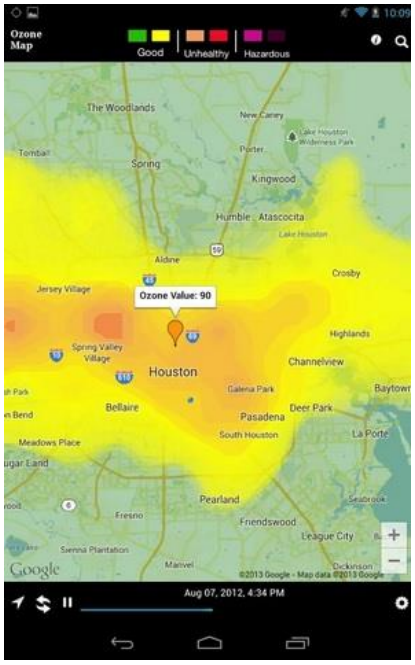


Figura 2. 8.-Aplicación móvil OzoneMap.

Esta aplicación es de tipo estacionaria y fue desarrollada por el Laboratorio de Fisiología de la Computación de la Universidad de Houston [19].

Air4ASEAN - Android

Esta aplicación (ver figura 2.9), muestra datos recientes sobre contaminación del aire en los países de la ASEAN, con más de 150 estaciones, en las redes de monitorización de calidad del aire y que incluye en la actualidad a Brunei, Malasia, Tailandia y Singapur. La información del mapa del ASMC incluye neblina, índices de concentraciones de contaminación, y valores de PM10 y PM2.5. Los usuarios pueden ver mapas, buscar, ordenar y agregar sitios de monitorización como favoritos.

Figura 2. 9.-Aplicación móvil Air4ASEAN.

Esta app usa datos de estaciones fijas y ha sido desarrollada por el Departamento de control de Contaminación Ambiental de Thailandia con fecha 13 de enero del 2015 [20].

Air Quality China - Android

Esta aplicación (ver figura 2.10), muestra el índice de calidad del aire de China, y es una aplicación móvil gratuita para las principales ciudades de este país. Incorpora un widget que se actualiza automáticamente cada 30 minutos.

Esta aplicación es de tipo estacionaria [21] .

Figura 2. 10.- Aplicación móvil Air Quality China.

airCheck – Android

Con AirCheck (ver figura 2.11), puedes consultar el índice de calidad del aire en Suiza y su principales ciudades. Muestra mapas y planos de las estaciones de monitorización, informa acerca de los efectos sobre la salud y de aquellos aspectos de los cuales uno debe ser consciente en caso de alta contaminación atmosférica. Esta aplicación muestra los niveles de PM10, partículas finas, NO_2, dióxido de nitrógeno, O_3, ozono y niebla [22].

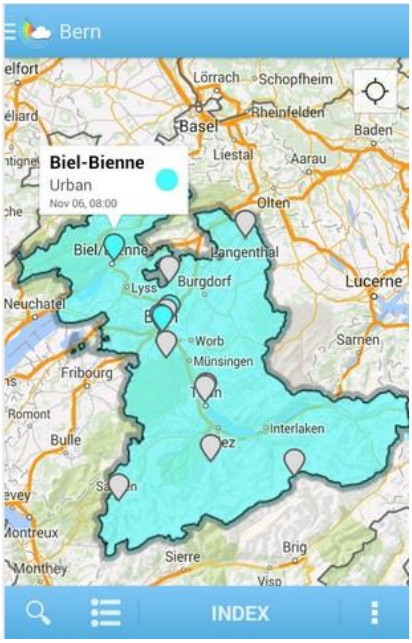


Figura 2. 11.-Aplicación móvil airCheck.

CALIOPE: Calidad del Aire - Android

Esta aplicación (ver figura 2.12), permite visualizar el pronóstico de la calidad del aire en España, además de los mapas de concentración para las próximas 12 horas de los principales contaminantes atmosféricos: ozono (O_3), dióxido de nitrógeno (NO_2), dióxido de azufre (SO_2) y material particulado (PM10 y PM2.5).

La aplicación utiliza el GPS de su dispositivo móvil para establecer su ubicación y consultar en línea las bases de datos del Sistema CALIOPE, mostrando el pronóstico de la calidad del aire en las estaciones más cercanas, a través de cinco categorías: buena, admisible, deficiente, mala y muy mala.

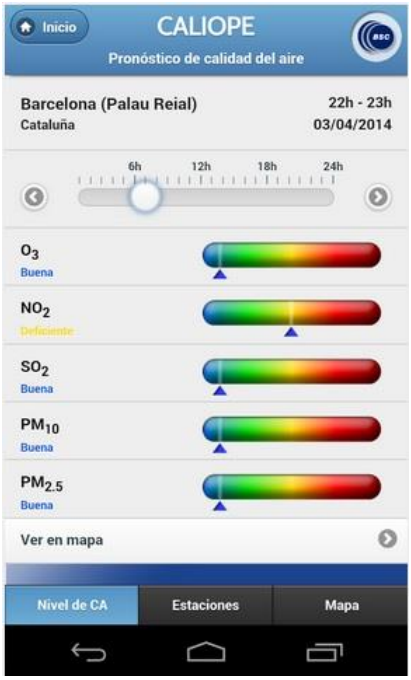


Figura 2. 12.- Aplicación móvil CALIOPE

Capítulo III: Tecnología empleada

Para el desarrollo de este Trabajo Fin de Máster es necesario describir las diferentes tecnologías implicadas, lo cual facilita la comprensión de las ideas tratadas en los capítulos siguientes. El sistema está compuesto por una arquitectura que integra varios componentes de hardware y software, de los cuales destacan el sistema móvil de monitorización ambiental y el servidor web. Estos componentes se integran mediante un gateway entre el dispositivo sensor y Android.

En esta sección se describe la arquitectura empleada para el sistema de manera general y específica.

3.1 Arquitectura empleada

En la figura 3.1 se muestra la arquitectura de hardware y software empleada para la solución propuesta. En la figura se destacan 4 componentes: el Sensor Waspmote Libelium, el Raspberry Pi, el dispositivo Android, y el Servidor web. Los diferentes componentes forman una cadena, siendo conectados por diferentes tecnologías de comunicación.

El primer componente es el Sensor Waspmote Libelium, el cual realiza la función de captura de datos medioambiental a través de sus diferentes sensores incorporados. En este Trabajo Fin de Master se trabajó con los siguientes sensores: CO_2, Ozono, contaminantes del aire y temperatura. Adicionalmente, mediante su GPS incorporado, ha sido posible obtener datos de posición, (latitud y longitud). La información leída de su placa de sensores es procesada a través su sistema Arduino. Esta información es tratada como un mensaje de caracteres. En particular, la información procesada puede ser leída por otros dispositivos a través de conexiones serie. Para esta propuesta se usó una conexión serie ZIGBEE para su interacción con el dispositivo Raspberry Pi.

El segundo componente es el Raspberry Pi, el cual tiene la función de gateway entre el componente Waspmote Libelium y el dispositivo Android. Raspberry Pi utiliza Python como entorno de desarrollo incorporado y se basa en el sistema operativo Raspbian. A través de Python se configura la Raspberry Pi para que sirva como cliente desde la perspectiva del componente Waspmote Libelium, y como servidor de comunicaciones desde la perspectiva del dispositivo Android. El servicio de comunicación que ofrece el Raspberry Pi se materializa a través de conexiones serie hacia los dispositivos anteriormente nombrados. El mensaje suministrado al enviar y recibir datos consiste en una cadena de caracteres simple. El mensaje que recibe y envía el Raspberry Pi incluye datos de: ozono, CO_2, contaminantes de aire,

temperatura, latitud, longitud, consumo de batería y la fecha y hora de captura. En cuanto a su portabilidad, el dispositivo Raspberry Pi se combina con una batería externa como fuente de alimentación.

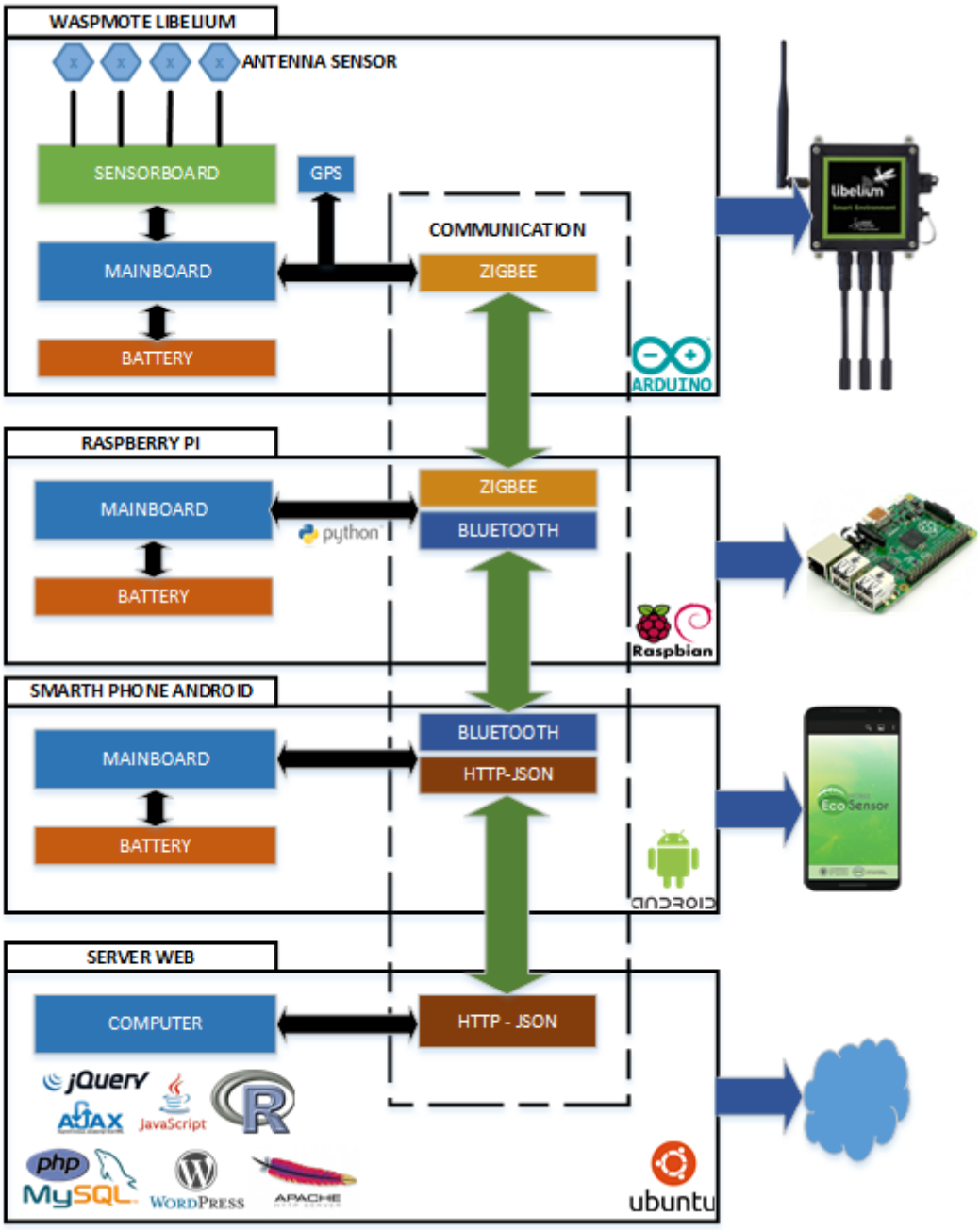


Figura 3. 1.- Arquitectura de hardware y software del sistema propuesto

El tercer componente es el dispositivo Android, en el cual se instala la aplicación de monitorización ambiental móvil. La aplicación ha sido desarrollada en Android Studio, y permite mostrar en tiempo real los datos obtenidos por el dispositivo gateway Raspberry Pi. El enlace entre el Gateway y el dispositivo Android, se realiza mediante una conexión Bluetooth.

El dispositivo Android permite almacenar los datos capturados en su memoria. Una vez almacenados podrán ser enviados al Servidor Web cuando el usuario lo desee mediante una petición de tipo POST usando JSON. Esta petición incluirá datos como ozono, CO_2, contaminantes de aire, temperatura, latitud, longitud, fecha y hora de captura y el identificador del usuario que envía la petición.

El último componente de la arquitectura es el Servidor Web. El Servidor Web procesa la información enviada por el dispositivo Android a través de un servicio web instalado en el servidor. El objetivo del servidor es que la información sea procesada y mostrada mediante diferentes tipos de gráficas. El servidor web combina varias tecnologías software con el fin de que la información sea mostrada al usuario mediante una interfaz sencilla y simple de utilizar.

Entre las tecnologías de software empleadas por el sistema se incluyen:

- MySQL.- Servidor de base de datos que permite almacenar los datos emitidos desde el dispositivo Android.
- WordPress.- Sistema gestor de contenidos, permite definir la interfaz del sitio web desarrollado. WorPress ha sido implementado sobre Apache como servidor web.
- PHP.- Lenguaje de desarrollo, permite programar características particulares para el sitio web propuesto, y adicionalmente ayuda a implementar el servicio web que procesa los datos emitidos.
- Javascript.- Lenguaje de programación interpretado, usado del lado del cliente, permite que las páginas web sean dinámicas por lo que se consigue una interfaz de usuario mejorada.
- JQuery y Ajax.- Son técnicas y librerías que usan JavaScript con el fin de realizar peticiones HTTP (POST o GET), sin necesidad de recargar la página web.
- R.- Software libre usado ampliamente en el campo de la investigación. Esta herramienta es importante porque nos permite procesar los datos almacenados en la base de datos MySQL y generar diferentes gráficos, entre los cuales tenemos: heatmap, kriging, plot y boxplot. Una de las características más relevantes es que su código puede ser ejecutado mediantes llamadas al Shell scripts en Linux (Para más información refiérase al apartado 3.5.5).

3.2 Sensor Waspmote Plug And Sense

Es un dispositivo desarrollado por la empresa Española Libelium, considerado de gama baja y bajo coste. Entre sus características, posee un tamaño aceptable y es fácil de usar (ver figura 3.2). Tiene 6 puertos para diferentes tipos de sensores (ver figura 3.3), y adicionalmente permite configurar interfaces de comunicación como: ZigBee Pro, Wifi, Bluetooth de baja energía, GPRS, GPRS+GPS, y 3G, entre otras. Está basado en Arduino, y en su sitio web se ofrecen actualizaciones y librerías para gestionar las comunicaciones y la transmisión de datos entre dispositivos.

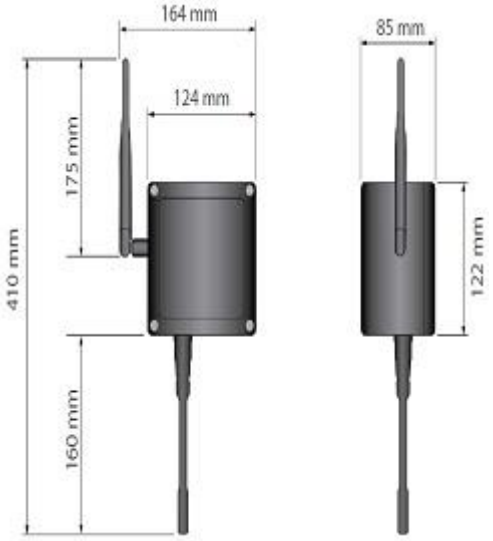


Figura 3. 2.-Tamaño Sensor Waspmote.

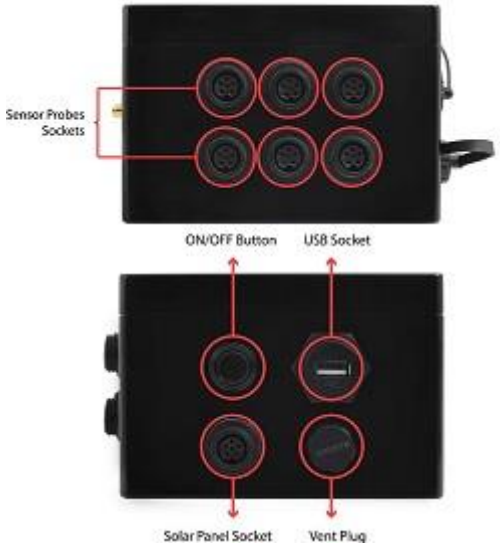


Figura 3. 3.-Socket Sensor Waspmote.

La familia de Waspmote Plup and Sense ofrece diferentes modelos de equipos de acuerdo a su función, destacando entre ellos:

- Smart Environment.- Diseñado para medir los parámetros ambientales como temperatura, humedad, presión atmosférica y algunos tipos de gases. Las principales aplicaciones de este Waspmote Plug & Sense son la medición de la contaminación en ciudades, granjas y criaderos, el control de procesos químicos e industriales, incendios forestales, etc.
- Smart Environment Pro.- Empleado para ambientes inteligentes, es una evolución del sensor anterior. Permite medir la contaminación y la calidad del aire en ambientes industriales o agrícolas con elevados requisitos de precisión, fiabilidad y rango de medición. Los sensores vienen calibrados de fábrica.
- Smart Security.- Su aplicación es el control de accesos, detección de presencia de líquidos, y apertura de puertas y ventanas.

- Smart Water.- Mide más de 6 parámetros de la calidad del agua, incluyendo los más relevantes como el oxígeno disuelto, reducción de oxidación, Ph, conductividad y temperatura.
- Smart Metering.- Su principal función es la de medir energía, consumo de agua y fugas, gestión de líquidos, y automatización industrial, entre otros.
- Smart Cities.- Permite medir varios parámetros para las ciudades inteligentes como nivel de acústica, calidad del aire, etc.
- Smart Parking.- Permite detectar plazas de aparcamiento disponibles, colocando el nodo bajo el pavimento.
- Smart Agriculture.- Permite medir varios parámetros relacionados con la agricultura.

3.2.1 Sensor Libelium Smart Environment

Este es el equipo usado para este proyecto (ver figura 3.4). El sensor Libelium Smart Environment está diseñado para monitorizar parámetros ambientales como la temperatura, humedad, presión atmosférica y algunos otros tipos de gases mediante los siguientes tipos de sensores:

- Temperatura
- Metano - CH_4
- LPG
- Air pollutants 1: $C_6H_5CH_3$, H_2S, CH_3CH_2OH, NH_3, H_2
- Air pollutants 2: C_4H_{10}, CH_3CH_2OH, H_2, CO, CH_4
- Derivados de Alcohol: CH_3CH_2OH, H_2, C_4H_{10}, CO, CH_4
- Humedad
- Presión atmosférica
- Dióxido de Carbono - CO_2
- Dióxido de nitrógeno - NO_2
- Ozono - O_3
- Hidrocarbonos - VOC
- Oxígeno - O_2
- Monóxido de Carbono – CO

Para este proyecto se van a tomar datos de los sensores de temperatura, Ozono (O_3), Dióxido de Carbono (CO_2), y Pollutants 2 (Contaminantes del aire). A continuación se ofrecen más detalles de estos sensores.

Figura 3. 4.- Sensor Libelium Smart Environment

3.2.2 Sensor de Temperatura

El sensor MCP9700A es un sensor analógico que convierte un valor de temperatura en una tensión analógica proporcional. El rango de salida de voltajes es entre 100 mV (-40°) y 1,75 V (125 °C), lo que resulta en una variación de 10 mV/°C, con 500 mV de salida para 0°C. En el Anexo A, tabla 4, se muestran más detalles técnicos de este sensor.

3.2.3 Sensor Ozono (O_3)

El MICS-2610 es un sensor resistivo que permite medir la variación de la concentración de O_3 entre 10 y 1000 ppb (partículas por billón). Su resistencia varía entre 11kΩ y 2MΩ aproximadamente. A diferencia de los MICS-2710, este sensor se alimenta a través de un regulador de voltaje de 2.5V, con un consumo de aproximadamente 34mA. La resistencia del sensor en el aire, así como su sensibilidad, pueden variar bastante, por lo que se recomienda calibrar cada uno de ellos antes de su uso. En el Anexo A, tabla 5, se muestran más detalles técnicos de este sensor.

3.2.4 Sensor de Dióxido de Carbono (CO_2)

El sensor TGS4161 entrega una salida de voltaje proporcional a la concentración de CO_2 en la atmósfera. Muestra un valor entre 220 y 490mV para una concentración de 350 ppm (partículas por millón) aproximadamente la concentración normal de CO_2 en el aire, el cual disminuye a medida que aumenta la cantidad de dicho gas. Diferentes sensores pueden mostrar una gran variabilidad en los valores iniciales de tensión en 350 ppm y en su sensibilidad, por lo que se recomienda calibrar cada sensor antes de incluirlo en la aplicación. En el Anexo A, tabla 6, se muestran más detalles técnicos de este sensor.

3.2.5 Sensor Air Pollutants 2

El sensor TGS2600 muestra sensibilidad a la variación de la concentración de numerosos gases que no se encuentran generalmente en la composición de la atmósfera y que se consideran

contaminantes. Entre estos tenemos principalmente el, etanol (CH_3CH_2OH), el isobutano (C_4H_{10}) y, con menor respuesta, monóxido de carbono (CO) y metano (CH_4). Este sensor también es sensible a las variaciones en la concentración de hidrógeno (H_2). La resistencia del sensor en el aire puede variar entre 10 y 90kΩ, con una relación de sensibilidad entre 0,3 y 0,6 para una concentración de H_2 de 10 ppm. Debido a esta variabilidad se recomienda calibrar cada uno de los sensores antes de su uso en una aplicación final. En el Anexo A, tabla 7, se muestran más detalles técnicos de este sensor.

3.3 Raspberry Pi

El Raspberry Pi es un ordenador en miniatura ligero y potente. Tiene un procesador ARM, utilizable para aplicaciones de PC de sobremesa. Fue creado por la Fundación Raspberry Pi, organización benéfica registrada para fines educativos y con sede en Reino Unido.

En su historia, Raspberry Pi ha tenido diferentes modelos de hardware y software. Para este trabajo se usó el Raspberry Pi en su modelo B (ver figura 3.5), el cual es más robusto y tiene mejor rendimiento que los anteriores. Entre sus características destacan: Memoria RAM (512 MB), una potencia de 3.5 watts (700 mA), y arquitectura RISC con microprocesador ARM 11. El sistema operativo que utiliza es el Raspbian basado en Linux Debian.

Figura 3. 5.- Raspberry Pi B.

Una de las características del Sistema operativo Raspbian es el lenguaje de programación Python, que viene preinstalado. Python es un lenguaje de programación interpretado por lo que no es necesario compilar su código, y además es gratuito. La solución que se plantea en este trabajo utiliza el lenguaje Python para las funciones de lectura de datos del sensor y entrega de datos al dispositivo Android.

3.4 Android

Android es un Sistema Operativo basado en el kernel de Linux, y fue originalmente ideado para dispositivos móviles con pantalla táctil. En la actualidad se puede encontrar en diferentes equipos como televisores, relojes, automóviles, etc.

Inicialmente fue desarrollado por Android Inc., compañía fundada en Palo Alto (California, EE.UU) en Octubre del año 2003 por Andy Rubin, Rich Miner, Nick Sears and Chris White, hasta que en el año 2005 Google lo respaldara económicamente y lo comprara.

El 5 de noviembre de 2007 se fundó un consorcio de varias compañías denominado Open Handset Alliance, y entre las que encuentran, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel, Intel, LG, Marvell Technology Group, Motorola, y T-Mobile; el 9 de diciembre de 2008, 15 nuevos miembros se unieron al proyecto, incluyendo ARM Holdings, Atheros Communications, Sony Ericsson, Huawei, Toshiba, ZTE, entre otros; con el fin de desarrollar estándares abiertos para dispositivos móviles, y coincidiendo con la formación de la Open Handset Alliance, se presentó el primer producto, Android, construido sobre la versión 2.6 de Linux.

La estructura del SO Android se compone de aplicaciones que se ejecutan en un framework Java, permitiendo desarrollar aplicaciones a través de una variante de este lenguaje de programación, para lo cual se necesitan los siguientes componentes de software:

1. El kit de desarrollo de Java estándar (Java Development Kit ó JDK).
2. El kit de desarrollo de Android (Android SDK).
3. Un entorno de desarrollo, como puede ser Eclipse o Android Studio.

Para este Trabajo Fin de Máster se ha utilizado el entorno de desarrollo Android Studio en su Versión 1.2.1.1, del 5 de Mayo de 2015.

3.4.1 Arquitectura

Android sigue una arquitectura formada por varias capas, las cuales tienen como finalidad facilitar el desarrollo de las aplicaciones, proporcionando los medios necesarios para que el desarrollador no tenga que programar a nivel de componentes físicos. Es importante conocer que la arquitectura de Android sigue una arquitectura en pila (*stack* en inglés), lo que permite

que las funciones de una capa utilicen elementos de las capas inferiores de forma transparente [23]. A continuación de describe brevemente cada una de las capas (Ver Figura 3.6).

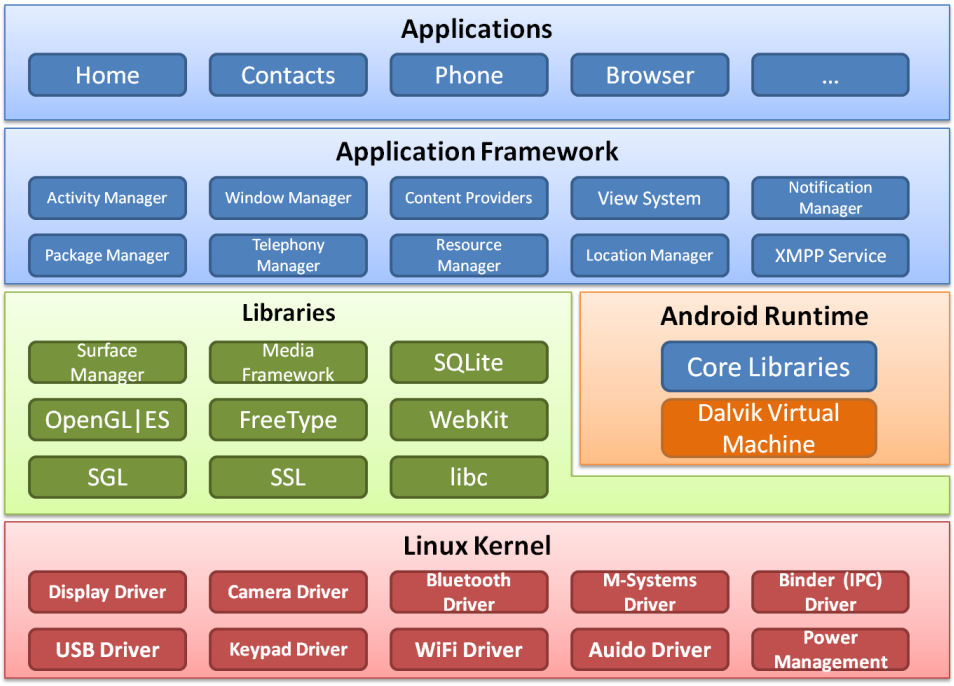


Figura 3. 6.- Arquitectura del Sistema Operativo Android.

- **Aplicación:** En este nivel se incluyen las aplicaciones Android estándar, así como aquellas que el usuario vaya agregando posteriormente. Todas estas aplicaciones utilizan los servicios, las API y librerías de los niveles inferiores.
- **Framework de Aplicaciones:** Este nivel proporciona una plataforma de desarrollo libre para aplicaciones con gran riqueza e innovaciones; esta capa se diseñó para simplificar la reutilización de componentes.
- **Librerías:** Este nivel incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema: System C library, bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite, entre otras.
- **Runtime de Android:** En este nivel se incluyen la librería del kernel y la máquina virtual Dalvik, la cual se usa para ejecutar los programas de Java.
- **Linux Kernel:** Este nivel está formado por el sistema operativo Linux versión 2.6, proporcionando servicios como seguridad, manejo de la memoria, multiproceso, pila de protocolos y soporte de drivers para dispositivos.

3.4.2 Ciclo de vida

Las interfaces de usuario basadas en ventanas que se ocultan poseen mucha aceptación en los ordenadores personales, pero para dispositivos con la pantalla reducida no son adecuadas. Android utiliza una interfaz que generalmente ocupa toda la pantalla del dispositivo. Así la interfaz de una aplicación estará formada por un conjunto de pantallas que permiten la interacción con el usuario. Cada una de estas pantallas será una instancia de una actividad.

Cuando el usuario ejecuta una aplicación, ésta se carga y se sitúa en primer plano, ocupando toda la pantalla. Desde esa aplicación abierta se podrán lanzar nuevas aplicaciones, pasar a otras pantallas de la misma aplicación o recuperar las que no son visibles. Las solicitud para lanzar nuevas actividades se llaman *Intents* (intentos, solicitudes). Un *Intent* es el responsable de que se lance la primera actividad de una aplicación, y nuevos *intents* serán responsables de ir lanzando las nuevas pantallas que sean necesarias. Todos estos programas y pantallas serán guardados en una pila a través del gestor de actividades del sistema. De esta manera el usuario podrá navegar por la pila y acceder a las pantallas abiertas con anterioridad a través del botón "Retornar" del dispositivo.

Una actividad Android puede encontrarse en diferentes estados:
Activa: Hace referencia a cuando una actividad ocupa el primer plano, la pantalla está visible y tiene la atención e interacción del usuario.
Pausada: cuando ha perdido la atención del usuario pero todavía es parcialmente visible, es decir; ocurre cuando se abre un diálogo encima de la pantalla.
Parada: cuando la actividad no es visible.

El usuario no tiene control sobre el estado actual de su aplicación, pues es controlado por el sistema, es decir, una actividad puede invocar su metodo *finish()* en cualquier momento, pero cuando escaseen los recursos y haya finalizado el proceso de una actividad, se acabará primero con aquellas que esten paradas, luego con las pausadas, y en alguna situación crítica alguna actividad que se encuentre activa. Cuando una actividad termina, ésta debe guardar su estado para que se pueda recuperar cuando sea lanzada de nuevo. Por consiguiente, el usuario recibirá notificaciones cuando se realicen cambios de estado, y la plataforma invocará determinados métodos de la actividad para que ésta pueda realizar las operaciones oportunas. Estos metodos son los siguientes [24]:

- **onCreate():** Se llama cuando se crea por primera vez la actividad. Aquí es donde se puede: crear vistas, enlazar datos de listas, etc. Este método también proporciona un paquete que contiene el estado previamente congelado de la actividad, si había uno.
- **onRestart:** Llamado después de que una actividad se ha detenido, y antes de haberse puesto en marcha de nuevo.
- **onStart:** Llamado cuando la actividad se está convirtiendo en visible para el usuario.
- **onResume():** Se llama cuando la actividad inicia la interacción con el usuario. Por lo general es usado para lanzar animaciones y música.
- **onPause():** Se llama cuando la actividad está a punto de ser enviada a segundo plano, normalmente porque otra actividad es lanzada. Es el lugar para detener animaciones y música.
- **onStop():** Se llama cuando la actividad ya no es visible para el usuario, porque otra actividad se ha reanudado. Esto puede ocurrir ya sea porque se está iniciando una nueva actividad, porque una ya existente tien preferencia, o porque éste está siendo destruido.
- **onDestroy():** Este método se ejecuta antes de que la actividad sea destruida.

A continuación se ilustra el ciclo de vida de una actividad (ver figura 3.7).

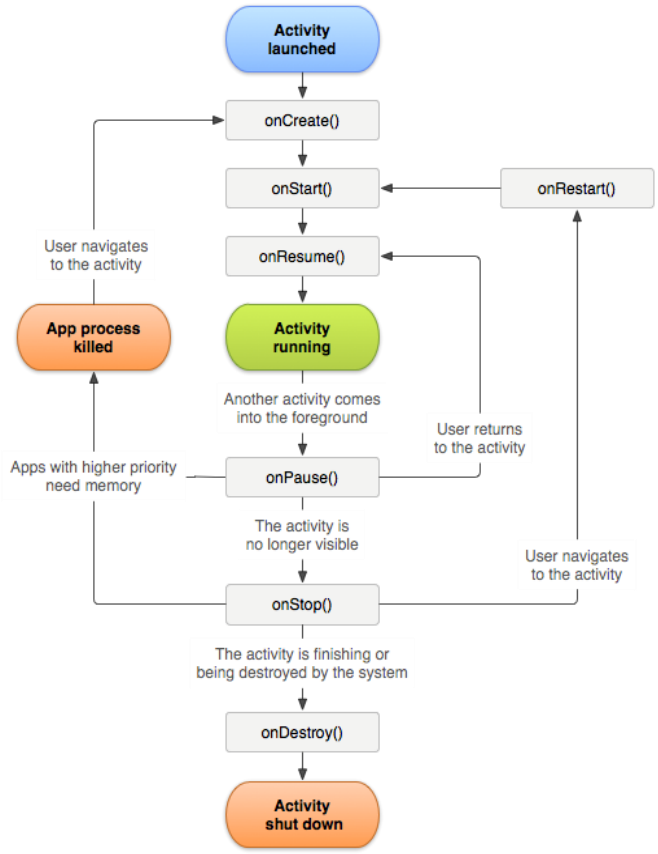


Figura 3. 7.- Ciclo de vida de una actividad Android.

3.4.3 SQLite

SQLite es un motor de base de datos Open Source. A diferencia del sistema de gestión de bases de datos cliente-servidor, SQLite no es un proceso independiente con el que la aplicación principal se comunica; en lugar de eso, la biblioteca SQLite se acopla con el programa pasando a incorporarlo, por lo que éste lee y escribe directamente a archivos de disco ordinarios. SQLite es la opción ideal para manipular datos dentro del Sistema Operativo Android.

3.5 Servidor Web

Un servidor web o servidor HTTP es un programa informático que se ejecuta del lado del servidor, es decir, se mantiene a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet, el cual es el encargado de contestar a estas peticiones, entregando como resultado una página web o información de acuerdo a lo solicitado.

Los Servidores Web establecen conexiones bidireccionales con el cliente. Para realizar la transmision de estos datos generalmente se usa el protocolo HTTP.

El cliente puede enviar información al servidor web mediante los siguientes métodos:

- **Método GET:** Sirve para obtener información del servidor como respuesta. Para esto se tiene que solicitar (request) algún dato, y el medio de envío es la URL. Este método es visible para el usuario.
- **Método POST:** Consiste en enviar información desde el cliente al servidor para que sea procesada. Dichos datos son procesados dando como respuesta alguna página con información; los datos enviados aparecen en el cuerpo del mensaje http, por lo que este método es invisible para el usuario.

Existen varios servidores web como Apache, Nginx y Microsoft IIS, entre otros. Para el caso de este Trabajo Fin de Máster se ha utilizado el servidor web Apache.

3.5.1 Apache

Este servidor web basado en http surgió para crear un recurso de libre acceso, robusto y de calidad comercial, y fue llevado a cabo por voluntarios de todo el mundo. En 1995 una primera versión del servidor resultó ser un gran éxito, y en 1999 miembros del grupo Apache formaron la “Apache Sofware Fundation”, que proporcionaría apoyo financiero, legal y organizacional para asegurar el futuro del proyecto, e impulsaría gran cantidad de proyectos de código libre [25].

Apache es de código abierto, modular y multiplataforma; el servidor consta de una sección llamada núcleo y diversos módulos que aportan mucha de la funcionalidad que se consideran básica para un servidor web. Algunos de los módulos más importantes son:

1. mod_ssl - Comunicaciones Seguras vía TLS.
2. mod_rewrite - Reescritura de direcciones.
3. mod_php - Páginas dinámicas en PHP.

3.5.2 PHP

PHP (Hypertext Preprocessor) es un lenguaje de código abierto, adecuado para el desarrollo web y que puede ser embebido en HTML. El lenguaje PHP [26] fue creado por Rasmus Lerdorf en 1995, como un simple conjunto de scripts PERL para el control de acceso a currículums on-line. El lenguaje fue creciendo en funcionalidad hasta permitir a los usuarios desarrollar aplicaciones Web dinámicas.

PHP permite conectarse a diferentes bases datos tales como MySQL, Postgress, Oracle y SQLite, entre otras. Para ser usado en el lado del cliente basta con un navegador web. En el lado del servidor es necesario tener instalado los módulos relativos a PHP.

3.5.3 WordPress

WordPress [27] nació por el deseo de un sistema web elegante, y su arquitectura fue construida en PHP y MySQL, y licenciado bajo la GPLv2 (o posterior). WordPress es un software reciente, pero sus raíces y desarrollo se remontan a 2001. Se trata de un producto estable, centrado en la experiencia del usuario y los estándares web. Hoy ha evolucionado para ser utilizado como sistema de gestión de contenido completo y mucho más, a través de los miles de plugins, widgets y temas disponibles.

3.5.4 MySql

Es un sistema de gestión de base datos relacionales, multihilo y multiusuario que es utilizado por muchos sitios web populares [28] tales como: Wikipedia, Facebook, Twitter e Youtube, entre otros. MySQL es muy utilizado en aplicaciones web, y su popularidad se debe a que se suele usar conjuntamente con PHP.

3.5.5 Lenguaje de Programación R

R [29] es un lenguaje de programación pensado para el análisis estadístico y representación gráfica de datos, siendo un software libre que se distribuye bajo licencia GNU GPL, además de ser un conjunto de herramientas integradas.

R integra variables, datos, resultados y funciones en un área de trabajo mediante objetos que llevan un nombre definido. Su información se encuentra estructurada en paquetes y librerías.

Adicionalmente existe RStudio, que es la interfaz gráfica para desarrollo en R. RStudio incluye una consola, editor de sintaxis que permite la ejecución de código, así como herramientas para creación de gráficos, depuración y la gestión del espacio de trabajo, entre otras.

En particular, para este Trabajo Fin de Máster, se ha hecho uso de varios paquetes para que R permita la creación de los gráficos en el Servidor Web. Los paquetes utilizados se muestran en la tabla 1.

Tabla 1.- Paquetes R instalados.

Paquetes	Descripción
sp	Datos espaciales, puntos, líneas, polígonos y tablas.
gstat	Modelado de variograma; Kriging simple, ordinario y universal, espacio temporal; simulación gaussiana, funciones de utilidad para trazado de mapas.
ggplot2	Gráficos en R, múltiples fuentes de datos, Multidimensional, etc.
raster	Lee, escribe, manipula, analiza y modela datos espaciales en cuadriculas. Implementa las funciones básicas y de alto nivel. Es compatible con el procesamiento de archivos de gran tamaño.
rasterVis	Método para mejorar la visualización y la iteración con los datos "raster".
maptools	Manipula y lee datos geográficos.
LSD	Crea un conjunto de colores para graficar, permite una gran cantidad de variaciones.
RgoogleMaps	Paquete con 2 propósitos: proporciona una cómoda interfaz de R para consultar del servidor Google para mapas estáticos, y (ii) utiliza el mapa como una imagen de fondo para superponer gráficos R. Esto requiere escalado coordinado.
RMySQL	Implementa interface para conectarse a Bases de datos MySQL y MariaDB
shapes	Rutinas para el análisis estadístico de formas.
dismo	Para división, aplicación y combinación de datos.
automap	Realiza una interpolación automática mediante la estimación de variogramas. Hace uso de llamadas “gstat”.
scales	Función para visualización graficas de mapas de datos, y proporciona métodos para determinar pausas automáticas y leyendas para los nombres en los ejes.

3.6 Tecnologías empleadas para la comunicación entre dispositivos

3.6.1 Libelium - Raspberry Pi

El Sensor Libelium Smart Environment tiene múltiples opciones de radio para comunicarse con otros sensores tales como: ZigBee, 802.15.4, Wifi, RF en 868Mhz y 900Mhz, 3G/GPRS y Bluetooth de baja energía en 2.4 GHz. En este trabajo se ha hecho uso de la tecnología ZigBee para leer datos del Sensor Libelium a través del Raspberry Pi.

ZigBee [30] es un estándar de comunicaciones inalámbricas de corta distancia y baja velocidad de datos, y en la actualidad es el estándar más aceptado para redes de sensores. Diseñado por la ZigBee Alliance, y está basado en el estándar IEEE 802.15.4. Este estándar tiene las siguientes características:

- Bajo consumo permitiendo usar equipos con batería.
- Bajo coste de dispositivos e instalación.
- Alcance corto (menor de 50 metros).
- Velocidad de transmisión menor que 250Kbps.
- Niveles de seguridad adecuados.

La lectura de datos entre Waspmote Libelium y el Raspberry Pi se realiza mediante una conexión serie. Su comportamiento es similar a la transferencia de datos entre Raspberry Pi y el dispositivo Android, por lo que se darán detalles más adelante. Su principal diferencia radica en que el Waspmote Libelium se comporta como servidor de comunicación y el Raspberry Pi se comporta como cliente. Hay que destacar que el sistema de procesamiento de datos en el Waspmote Libelium es Arduino.

3.6.2 Raspberry Pi – Dispositivo Android

El enlace entre el Raspberry Pi y el dispositivo Android se realiza mediante Bluetooth. Bluetooth es un protocolo de comunicación para Redes Inalámbricas de Área Personal (WPAN) que permite el envío de datos entre diferentes dispositivos mediante un enlace de radiofrecuencia en la banda de los 2.4 GHz.

Bluetooth [31] fue ratificado como estándar por el IEEE en el año 2002, y desde entonces ha tenido mejoras en sus prestaciones, partiendo de la versión 1.1 hasta la actualidad con su

versión 4.0 del año 2010. Esta última versión incluye características del Bluetooth clásico, del Bluetooth de alta velocidad, y protocolos de bajo consumo.

Unas de las áreas en la que Bluetooth es más utilizado es en las comunicaciones entre equipos móviles. Por lo general se utiliza en áreas reducidas con dos o más dispositivos. Para comunicarse, inicialmente debe haber un descubrimiento y emparejamiento entre dispositivos para luego empezar la transmisión. La figura 3.8 muestra como ejemplo un dispositivo Bluetooth con conector USB.

Figura 3. 8.- Dispositivo Bluetooth USB

La manera de comunicarse está basada en el SDK de Android, y usa una interfaz similar a los sockets TCP para comunicarse a través de Bluetooth. Un socket es una interfaz de programación de aplicaciones (API) bien conocida. Para comunicaciones vía Internet (y no solo), ofreciendo mecanismos para la entrega de paquetes de datos provenientes de una tarjeta de red a los procesos o hilos apropiados. Los sockets permiten implementar una arquitectura cliente-servidor, donde el inicio de la comunicación puede ser iniciada por un programa conocido como “cliente”, mientras el segundo programa espera las peticiones de este programa, el cual se conoce como servidor.

En un socket el envío y recepción de datos es producida siempre que una llamada realizada por el “cliente” es aceptada por el “servidor” y se ha completado con éxito. Una vez que el socket está establecido, puede ser utilizado hasta que la conexión falle debido a un error del enlace, o por terminación del usuario. Un ejemplo de conexión por sockets a través de TCP se muestra en la figura 3.9.

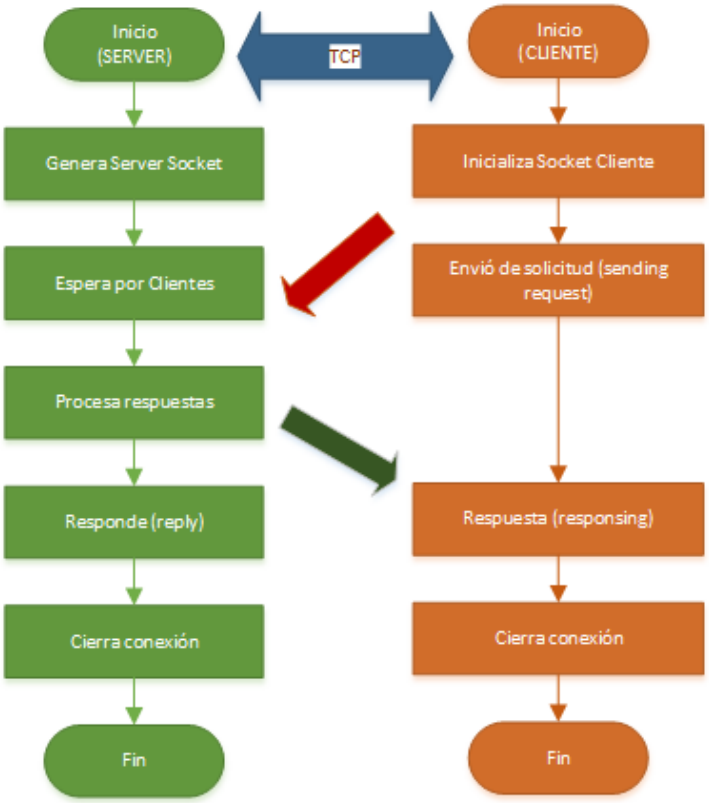


Figura 3. 9.- Transferencia TCP-SOCKET.

En este componente el que hace las funciones de servidor será el Dispositivo Raspberry Pi, y el cliente será el dispositivo móvil.

Las clases de la API Android usadas para realizar la comunicación del cliente fueron:

- BluetoothAdapter: Es el adaptador Bluetooth local. El "BluetoothAdapater" permite realizar tareas fundamentales como iniciar la detección de dispositivos, consultar los dispositivos emparejados, crear una instancia de "BluetoothDevice" utilizando una dirección MAC conocida, y crear un "BluetoothServerSocket" para escuchar las solicitudes de conexión de otros dispositivos.
- BluetoothDevice: Representa un dispositivo Bluetooth remoto. Permite crear una conexión con el dispositivo remoto, y consultar información como el nombre, dirección clase y estado del enlace. Las operaciones en esta clase se realizan en la dirección hardware del Bluetooth remoto, utilizando "BluetoothAdapter".
- BluetoothSocket: Representa el interface para un socket Bluetooth. Es el punto de conexión que permite a una aplicación el intercambio de datos con otro dispositivo Bluetooth mediante el uso de "InputStream" y "OutputStream".

Por otro lado se ha utilizado Python para el desarrollo del programa servidor, para lo cual se usó la librería PyBluez. PyBluez [32] es un módulo de extensión de Python escrito en C que proporciona acceso a los recursos del sistema Bluetooth y es orientado a objetos. PyBluez actualmente soporta 2 tipos de "BluetoothSocket": RFCOMM y L2CAP. El RFCOMM es el utilizado para esta transmisión, y se crea pasando RFCOMM como argumento al constructor "BluetoothSocket". Un ejemplo de código del lado servidor se muestra en la figura 3.10.

```
1  import bluetooth
2  server_sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )
3  port = 1
4  server_sock.bind(("",port))
5  server_sock.listen(1)
6  client_sock,address = server_sock.accept()
7  print "Accepted connection from ",address
8  data = client_sock.recv(1024)
9  print "received [%s]" % data
10 client_sock.close()
11 server_sock.close()
```

Figura 3. 10.- Ejemplo de código Python para la comunicación Socket Server.

El mensaje creado en el lado del servidor utiliza el formato de la figura 3.10.

id=	valor;	bat=	valor;	latitude=	valor;	longitude=	valor;	temperature=	valor;	ozone=	valor;	co2=	valor;	apollution=	valor;	fechayhora=	valor

Figura 3. 11.- Formato de mensaje Raspberry Pi – Android.

3.6.3 Dispositivo Android – Servidor Web

En este Trabajo Fin de Máster se ha hecho uso de los servicios web para la comunicación entre el dispositivo Android y el servidor web. Un servicio web es una solución que usa cualquier tecnología web(PHP, JAVA, .NET, etc.) para facilitar la interoperabilidad entre varios sistemas, independiente del lenguaje de programación o sistema operativo en que fueron desarrollados. Estos servicios web deben cumplir un formato estándar entendible por diferentes plataformas, siendo XML o JSON un buen ejemplo de ello.

En este trabajo se utilizó el formato JSON(JavaScript Object Notation). JSON es un formato de intercambio de datos liviano, y su sencillez al escribir un analizador sintáctico (parser) ha dado lugar a la generalización de su uso. Hoy día se presenta como una alternativa a XML en AJAX.

En este contexto, se unificó JSON con el sistema operativo Android (ver figura 3.12), realizando peticiones HTTP del tipo POST. Este método permite el envío de datos al servidor web.

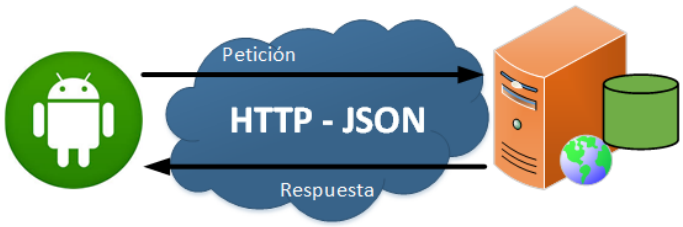


Figura 3. 12.- Transmisión de datos entre terminal Android - Servidor Web.

El formato del mensaje que se transmite entre el dispositivo Android y el Servidor Web tiene la siguiente estructura (ver figura 3.13).

```
{
    "detalle": [
        {
            "latitude": "39.470577",
            "longitude": "-0.336604",
            "temperature": "15",
            "batery": "90",
            "ozone": "28.27",
            "co2": "0.58413",
            "apollution": "0.618838",
            "fechayhora": "16/07/2015 15:05:15",
            "estado": "N",
            "idsqlite": "1"
        }
    ],
    "maestro": {
        "nombre": "Traza #1",
        "fechaini": "16/07/2015 15:05:15",
        "fechafin": "16/07/2015 15:06:15",
        "estado": "N",
        "idusuario": "1",
        "idsqlite": "1"
    }
}
```

Figura 3. 13.- Formato de mensaje JSON entre Android y Servidor Web.

Capítulo IV: Análisis del sistema

Una vez conocidas las tecnologías empleadas que dan soporte a la solución planteada para este Trabajo Fin de Máster, se procede a describir las características que tendrá la solución. Como señalan Canós et al [33], no existe una metodología universal para el desarrollo de software. La propuesta desarrollada se ha adaptado al contexto de este proyecto en tiempo de desarrollo, tipo de sistema, recursos tecnológicos, económicos y humanos.

Dicho esto, en este capítulo se describen las principales características de la solución presentada, las técnicas aplicadas, los casos de usos y comportamiento del sistema.

4.1 Descripción del sistema

El sistema desarrollado pretende ofrecer al usuario una solución en tiempo real de los datos generados por los diferentes sensores ambientales descritos en el capítulo III. Estos valores deben ser mostrados de tal forma que el usuario pueda comprender dicha información. De igual forma, el sistema debe tener la capacidad de ubicarse en cualquier punto geográfico que el usuario desee. Como punto a considerar, el sistema deberá empezar la captura de información desde que arranca hasta que el usuario finalice dicho proceso. Al finalizar se dará al usuario la opción de almacenar la información capturada.

Otra característica del sistema, es que el usuario pueda visualizar el recorrido a través de las herramientas de mapas de Google, de tal forma que se muestre la variación de los valores de dichos sensores. En relación con lo anterior, también debe tener la capacidad de mostrar gráficamente dicho resultados. Toda la información procesada debe ser almacenada localmente para futuras visualizaciones.

Adicionalmente, el sistema debe tener un repositorio central donde se reciba la información emitida por los diferentes sistemas lectores móviles. Este repositorio debe tener opciones para visualizar la información mediante gráficos como mapas de calor o diagramas de caja y bigotes.

Para lograr implementar dichas funcionalidades el sistema destaca dos componentes principales del software desarrollado, como son el cliente y el servidor web (ver figura 4.1). Hay que destacar que los medios de comunicación existentes entre estos componentes fueron descritos en el capítulo III.

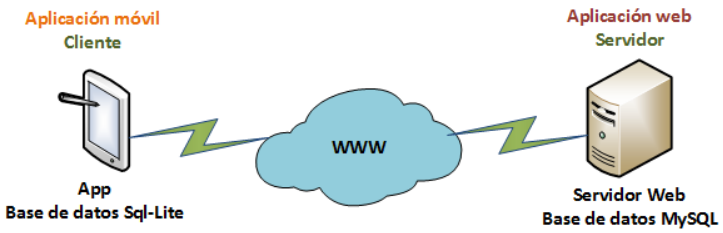


Figura 4. 1.- Componentes del sistema

El dispositivo cliente donde se ejecuta la aplicación móvil está basado en tecnología Android, y proporciona las herramientas necesarias para leer datos en tiempo real; y tiene también la capacidad de cómputo necesaria para mostrar gráficos, y almacenar la información recibida en su base de datos interna SQLite. Como se indicó en el capítulo III, la tecnología empleada para leer datos de contaminación de los sensores será Bluetooth.

El Servidor Web tiene la función de procesar los datos recibidos. Utiliza el software R para tratar la información almacenada en su base de datos MySQL, de tal forma que permita generar los diferentes gráficos descritos anteriormente. Estos gráficos serán visualizados a través de una interfaz web mediante una consulta individual y un informe general estadístico sobre los datos recabados. Para tener acceso a estas opciones se necesitará de un usuario que inicie sesión en el portal web creado.

4.2 Requisitos del Sistema

Es necesario detallar la funcionalidad de cada uno de estos componentes. En Ingeniería de Software se han descrito diferentes estilos para categorizar los requisitos de un sistema, y en nuestro caso se ha usado la clasificación de requisitos funcional y no funcional.

- **Los requisitos funcionales** son declaraciones de los servicios que proveerá el sistema, refiriéndose a las entradas, comportamientos y salidas. Ejemplos de estos son: los cálculos, transformación de datos, etc.

- **Los requisitos no funcionales** describen todos los requisitos que no afectan a un comportamiento específico del sistema. Ejemplo de estos son: coste, interfaz, etc.

4.2.1 Requisitos Funcionales

4.2.1.1. Aplicación móvil

- RF.1.1.1 El sistema debe iniciar el proceso de captura de datos con un solo clic al iniciar la aplicación.
- RF.1.1.2 El sistema debe permitir ejecutarse en segundo plano sin que interfiera en las tareas habituales del usuario.
- RF.1.1.3 El sistema debe permitir visualizar resultados en tiempo real mientras el proceso de captura esté activo.
- RF.1.1.4 Los resultados entregados por los sensores deben ser mostrados de manera cualitativa de tal forma que puedan ser comprensibles por el usuario.
- RF.1.1.5 El sistema debe permitir visualizar resultados de manera gráfica y utilizando tecnologías de Google Maps.
- RF.1.1.6 El sistema debe permitir finalizar el proceso de captura en cualquier momento.
- RF.1.1.7 El sistema debe permitir almacenar y consultar la información emitida por los sensores.
- RF.1.1.8 El sistema debe permitir enviar la información de la traza almacenada a un servidor Web.
- RF.1.1.9 El sistema debe permitir modificar las opciones de configuración.

4.2.1.2 Aplicación Web

- RF.1.2.1 El sistema debe mostrar información respecto a la aplicación desarrollada.
- RF.1.2.2 El sistema debe permitir recibir peticiones de la aplicación móvil.
- RF.1.2.3 El sistema debe permitir iniciar sesión.
- RF.1.2.4 El sistema debe permitir generar gráficos y almacenar dichos gráficos.
- RF.1.2.5 El sistema debe permitir mostrar un informe general de los sensores según lo recogido en la traza.
- RF.1.2.6 El sistema debe permitir mostrar un informe individual de sensores según lo recogido en la traza.

4.2.2 Requisitos no Funcionales

4.2.2.1 Aplicación móvil

- RN.2.1.1 La aplicación móvil usa el entorno de desarrollo Android Studio de Google, entre sus principales características tenemos:
    - Versión 1.2.1.1.
    - JRE: Java 1.7.0_75.
    - Entorno de gráfico, dependiente de: "com.github.PhilJay:MPAndroidChart:v2.1.0".
    - Entorno de mapas de Google, dependiente de: "com.google.android.gms:play-services:7.0.0".
- RN.2.1.2 La versión compilada puede instalarse en un dispositivo con sistema operativo Android versión 4.0.x. o superior.
- RN.2.1.3 El idioma de la aplicación móvil es el español. Las etiquetas de su código estarán disponibles para su traducción a otros idiomas.
- RN.2.1.4 La aplicación móvil debe comunicarse mediante un dispositivo Bluetooth.
- RN.2.1.5 La aplicación móvil debe tener acceso a Internet con el fin de enviar datos al servidor Web.
- RN.2.1.6 La aplicación móvil necesita como mínimo 30 MB de almacenamiento disponible para su instalación y funcionamiento.
- RN.2.1.7 La aplicación móvil debe mostrar una interfaz amigable, intuitiva y fácil de utilizar.

4.2.2.2 Aplicación Web

- RN.2.2.1 La aplicación web usa las siguientes características técnicas en su desarrollo:
    - WordPress plantilla "Fusion".
    - Apache 2.4.7.
    - PHP versión 5.5.9.
    - JSON encode y tecnología AJAX.
    - MySql versión 5.6.25 Comunity Server (GPL).
    - Lenguaje R.
- RN.2.2.2 La aplicación web debe aproximarse a los estándares de Usabilidad.
- RN.2.2.3 La aplicación web puede instalarse en servidores con sistema operativo Linux o Windows. Se recomienda un espacio de almacenamiento superior a 1GB con 1GB de memoria RAM.

4.3 Diagramas y Casos de Uso

En el desarrollo de software, uno de los métodos que ayuda a la representación de los requerimientos son los casos de uso. Los casos de uso son una técnica que describe el comportamiento de un sistema y sus especificaciones, involucrando al actor como usuario del sistema. Como se dijo anteriormente, el proyecto está conformado por 2 componentes: el primer componente es el sistema móvil, donde su actor principal es el usuario móvil. El segundo componente es el sistema web, que tiene como actores los usuarios web.

4.3.1 Caso de uso del Sistema Móvil

En la figura 4.2 se muestra el caso de uso del componente móvil. En la tabla 2 se muestra la relación con los requisitos funcionales, donde se evidencia el cumplimiento de las características descritas anteriormente.

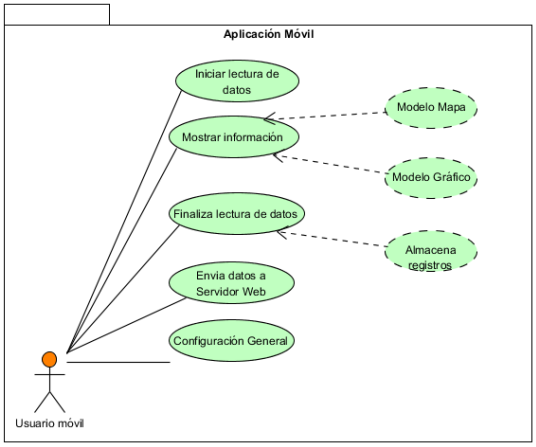


Figura 4. 2.-Diagrama de casos de uso de la aplicación móvil.

Tabla 2.- Relación casos de uso de la aplicación móvil - requisitos funcionales.

Casos de uso - Aplicación Móvil		
Identificador	Nombre	Requisitos cubiertos
C.1.1	Iniciar lectura de datos	RF.1.1.1, RF.1.1.2
C.1.2	Mostrar información	RF.1.1.3, RF.1.1.4
C.1.2.1	Modelo Mapa	RF.1.1.5
C.1.2.2	Modelo Gráfico	RF.1.1.5
C.1.3	Finaliza lectura de datos	RF.1.1.6
C.1.3.1	Almacena registros	RF.1.1.7
C.1.4	Enviar datos a Servidor Web	RF.1.1.8, RF.1.1.3, RF.1.1.4
C.1.5	Configuración general	RF.1.1.9

4.3.2 Caso de uso del Sistema Web

En la figura 4.3 se muestran los casos de uso del componente web, y en la tabla 3 se muestra la relación de éstos con los requisitos funcionales, evidenciando el cumplimiento de las características descritas anteriormente.

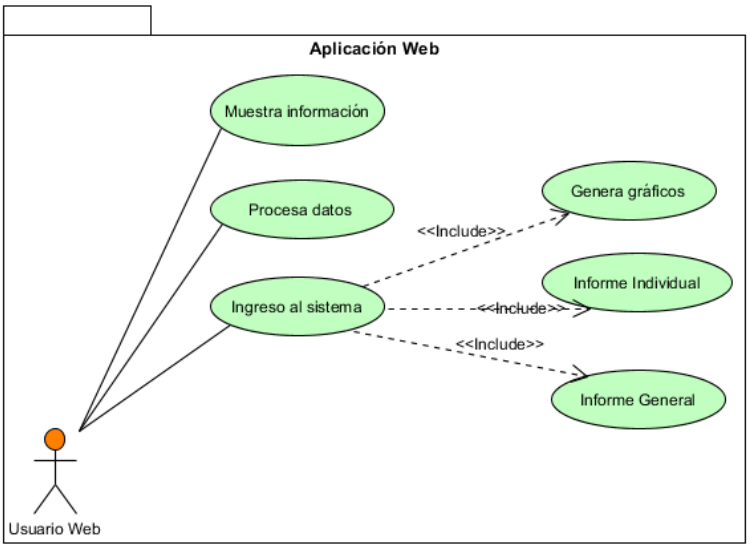


Figura 4. 3.- Casos de uso para la Aplicación Web.

Tabla 3.- Relación casos de uso de la Aplicación web – Requisitos funcionales.

Casos de uso - Aplicación Web		
Identificador	Nombre	Requisitos cubiertos
C.2.1	Muestra información	RF.1.2.1
C.2.2	Procesa datos	RF.1.2.2
C.2.3	Ingreso al sistema	RF.1.2.3
C.2.3.1	Genera gráficos	RF.1.2.3, RF.1.2.4
C.2.3.2	Informe general	RF.1.2.3, RF.1.2.5
C.2.3.3	Informe individual	RF.1.2.3, RF.1.2.6

4.4 Escenarios de casos de usos

Los escenarios en un caso de uso representan el flujo exitoso o secuencias de pasos que realiza el actor principal sobre el sistema. En los Anexos B y C se describen los diferentes escenarios para la aplicación móvil y la aplicación web, respectivamente.

Capítulo V: Diseño del Sistema

En este capítulo se describe arquitectura e interfaz de los dos componentes del sistema, aplicación móvil y aplicación web, permitiendo distinguir las funcionalidades de cada uno. Adicionalmente se usará un diagrama de clases UML para describir las características relevantes de cada componente del sistema.

5.1 Sistema Mobile Eco Sensor

En las figuras 5.1 y 5.2 se muestra la solución al sistema planteado. En la figura 5.1 se visualiza la aplicación móvil instalada en una VALENBISI, y en la figura 5.2 se muestra la solución web diseñada.

Figura 5. 1.- Ejemplo de Mobile Eco Sensor instalado en Valenbisi.

Figura 5. 2.- Interfaz web de Mobile Eco Sensor.

5.2 Arquitectura

5.2.1 Sistema Móvil

El sistema móvil ha sido desarrollado usando una arquitectura orientada a objetos, y su desarrollo se ha realizado en Android Studio usando Java JDK 1.7. En la figura 5.3 se muestra el diagrama de clases sobre el cual se basa su diseño.

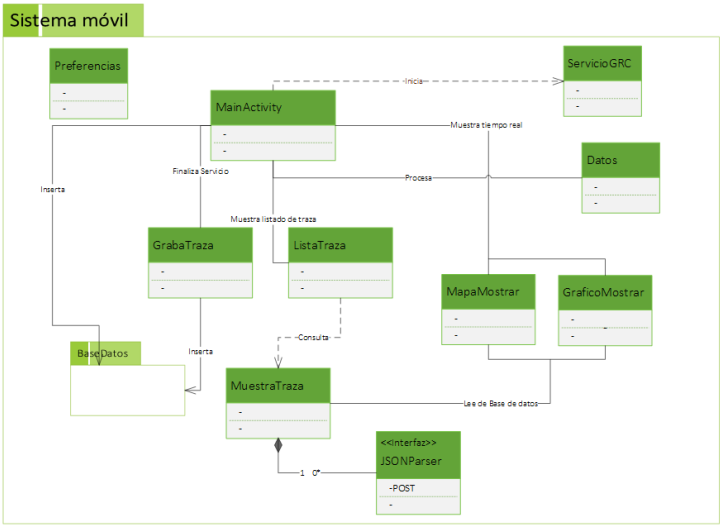


Figura 5. 3.- Diagrama de clases del sistema móvil.

A continuación se describen las características relevantes (atributos y métodos destacados) de algunas de las clases que se presentan en la figura anterior.

Clase MainActiviy: Es la actividad principal de la aplicación. Contiene los diferentes botones de accesos a las opciones del sistema. Permite iniciar y finalizar la captura de datos. Al proceso de iniciar y finalizar la captura de datos lo conocemos como generación de traza, la cual estará identificada por un id único. En el anexo D se muestran los atributos y métodos importantes de esta clase.

Clase ServicioGRC: Hereda características de "Service", y su función es solicitar datos a través de Bluetooth al servidor Raspberry Pi. Este servicio hace uso de las librerías internas de Android descritas en el capítulo de tecnologías empleadas. Cabe señalar que este servicio se ejecuta en background, por lo que no afecta al hilo principal de la aplicación. En el anexo D se muestran los atributos y métodos importantes de esta clase.

Clase Datos: Es la clase donde fluyen todos los mensajes que son leídos por la clase "ServiceGRC". Esta clase descompone el mensaje entregado por el dispositivo Raspberry Pi. En el anexo D se muestran los atributos y métodos importantes de esta clase.

Clase GrabaTraza: Esta clase permite visualizar el resumen de los datos capturados, mostrando su fecha de inicio y fecha de fin de captura de datos. Adicionalmente muestra valores máximos, mínimos y promedios de los valores obtenidos. En el anexo D se muestran los atributos y métodos importantes de esta clase.

Clase MuestraTraza: Esta clase tiene la misma funcionalidad que la anterior "GrabaTraza". Muestra un resumen de los datos capturados, así como valores máximos, mínimos y promedios. Tiene el botón enviar, el mismo que realiza la llamada a un servicio web para procesar e insertar los registros en la base de datos MySQL del servidor web. En el anexo D se muestran los atributos y métodos importantes de esta clase.

Clase JSONParser: Esta clase permite la interacción entre Android y el Servidor web a través de conexiones HTTP. En el anexo D se muestran los atributos y métodos importantes de esta clase.

Clase Database: Esta clase permite la inserción de los datos en la base de datos SQLite. Esta clase extiende la clase "SQLiteOpenHelper". En este desarrollo se usó una capa intermedia conocida como "Content Provider". Content Provider permite compartir datos entre aplicaciones y acceder a los mismos. Algunos métodos se describieron en la clase "GrabaTraza".

Para comprender esta clase, en la figura 5.4 se muestra el diagrama entidad relación, donde se observa una relación de 1 a muchos entre "msensores" y "dsensores". "msensores" contiene los datos de la traza, con un solo registro por traza. En "msensores" se almacenan registros como fecha y hora inicial y final, así como el identificador de usuario, entre otros. En "dsensores" guarda los datos capturados por el sensor como la temperatura, CO_2, ozono, fecha, latitud, longitud, y el registro de estado de la batería del equipo sensor. En este diseño el autor considera a "msensores" como registro maestro, y "dsensores" como registro que ofrece más detalles. Cabe señalar que en este diseño los campos de tipo float no son permitidos, por lo que en SQLite estos datos son de tipo "REAL". Finalmente se observa la entidad "asensores", que se refiere a una tabla de tipo de auxiliar que tiene la función de almacenar temporalmente los registros procesados y emitidos por el sensor. En el anexo D se muestran los atributos y métodos importantes de esta clase.

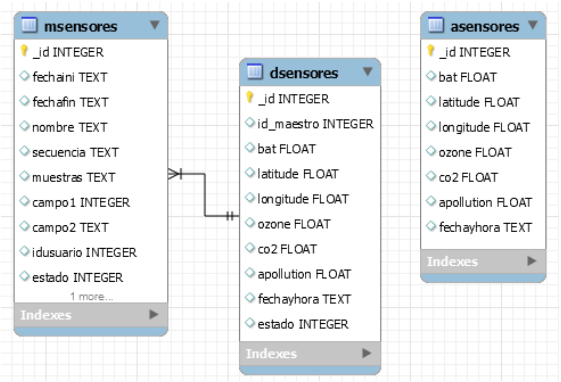


Figura 5. 4.- Modelo de base de datos del sistema móvil

Adicionalmente, en el diseño se destacan algunos permisos necesarios en el archivo manifiesto para Android Studio. Éstos son:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERACT_ACROSS_USERS_FULL"/>
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<uses-library android:name="com.google.android.maps" />

<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="key-android-google-maps" />
```

Las dependencias en el archivo de configuración build.gradle (módulo: app):

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.1.0'
    compile 'com.google.android.gms:play-services:7.0.0'
    compile 'com.github.PhilJay:MPAndroidChart:v2.1.0'
    compile 'com.nineoldandroids:library:2.4.+'
    compile files('libs/MPChart.jar')
    compile ('org.apache.httpcomponents:httpmime:4.3.5') {
        exclude group: 'org.apache.httpcomponents', module: 'httpclient'
    }
}
```

5.2.2 Sistema Web

El sistema web está diseñado mediante una plantilla de tipo WordPress, haciendo uso del lenguaje PHP, de una base de datos MySQL, de la tecnología AJAX, y del lenguaje R para el tratamiento de datos y obtención de gráficas. En la figura 5.5 se muestra el diagrama de clases sobre el cual se basa su diseño.

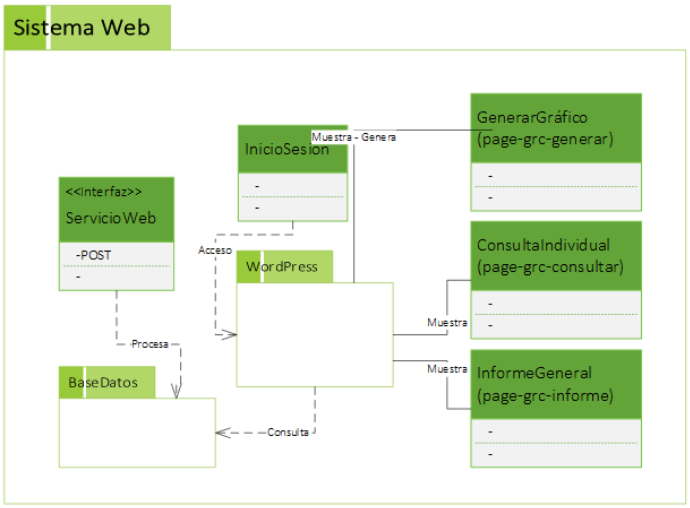


Figura 5. 5.- Diagrama de clases del sistema web.

WordPress, a través de su editor "wp-admin", permite al administrador de contenidos hacer uso de diferentes recursos como: crear páginas, categorías, subcategorías, menús, etc. Adicionalmente, WordPress puede agregar diferentes tipos de plugins y widgets, que permite dotar al sitio web de mayor funcionalidad. También se puede modificar las características de su diseño cambiando su fondo y el tema en general.

En este diseño se hizo uso de la página por defecto "Login Page", a partir de la cual se creó una página que permita iniciar sesión a los usuarios registrados previamente.

Con el fin de mostrar una página con una funcionalidad específica, WordPress permite crear plantillas diferentes a las páginas por defecto. Para este sitio web, hubo necesidad de crear páginas para generar gráficos, realizar consultas individuales y generar informes generales. Estas funcionalidades se crearon haciendo copias de los archivos por defecto de las plantillas predeterminada "page.php" (módulo del cuerpo de la página), "header.php" (módulo que contiene la cabecera del tema), y "footer.php" (módulo que contiene el pie de página del tema).

Los archivos creados tienen relación con los del diagrama de clases vistos en la figura anterior, y que son:

- page-grc-generar.php
- page-grc-informe.php
- page-grc-consultar.php

El archivo "**page-grc-genera.php**", hace uso de otros archivos PHP, como son header-grc.php y footer-grc.php. Esta página muestra los parámetros que selecciona el usuario para crear un gráfico. Estas características se describirán más adelante. Adicionalmente, esta clase invoca a

header-grc.php con el fin inicializar las particularidades de diseño (CSS) y su correspondientes funciones javascript.

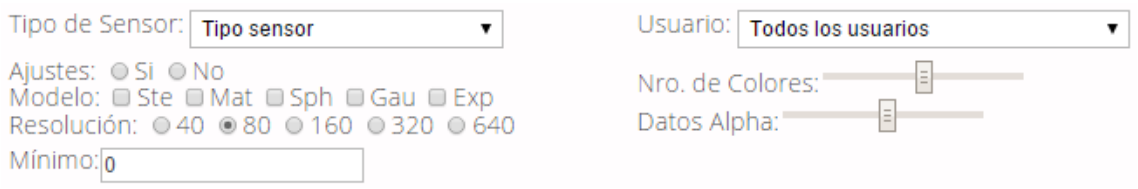


Figura 5. 6.- Opciones para generar gráficos.

Entre las funciones javascript utilizadas para esta página tenemos:

- $(document.ready(function())).- Esta función tiene 3 métodos, el primero (#cboTipoSensores.change) identifica el tipo de sensores que el usuario desea generar, y si el usuario selecciona un valor automáticamente llama a la función Llenar_Grid que muestra la información en forma de tabla. El segundo método (#cboUsuarios.change) realiza un filtro de acuerdo al usuario seleccionado. Este método refresca la tabla automáticamente. El último método (#grc-grid.on.click) muestra la fila seleccionada de la tabla con un color diferente al realizar click sobre ella.

- Función Llenar_Grid(idTipoSensores, idUsuarios).- Muestra en formato de tabla las trazas almacenadas en la base de datos. Estos valores pueden ser filtrados por el tipo de sensor, y por el usuario. Esta función hace uso de las librerías “jquery.dataTables.js” y “jquery.dataTables.css”

- Función GenerarGraficoClick(Codigo).- Genera los gráficos según el identificador seleccionado(traza). Esta función invoca a objetos tipo AJAX para que devuelva los gráficos generados. Para realizar esta acción se creó un script sobre R (ver Anexo E), el mismo que, al recibir los parámetros seleccionados, genera los diferentes tipos de gráficos (heatmap, kriring, plot y boxplot) según su id. Estos gráficos son almacenados en el servidor web en una carpeta predeterminada y para cada tipo de gráfico.

  El formato de los gráficos generados es el siguiente:

  - CO-ID# Imágenes para CO_2
  - OZO-ID# Imágenes para Ozono
  - AP-ID# Imágenes para contaminantes del aire
  - TE-ID# Imágenes para temperaturas

- Función (#pestana1, 2, 3, 4).- Realiza la función de tipo pestaña (tab). Usa el evento click con el fin de mostrar gráficos: heatmap, kriging, plot, boxplot del 1 al 4, respectivamente. Para realizar esta acción usa las librerías "tabstyle.css" y "jquery.js".

En el archivo "header-grc.php", se modificó el identificador del menú por "menu-grc". Esta acción se la realizó habiendo generado dicho menú desde el administrador de contenido de WordPress.

Con respecto a "footer-grc", lo que se hizo fue modificar su tamaño e incluir también el menú generado para esta plantilla; el comportamiento es igual para los modulos "page-grc-consultar.php" y "page-grc-informe".

Otra funcionalidad que tiene page-grc-generar.php es que llama al archivo grc-clasemaestra.php. Esta última tiene 3 métodos que son: conexion(), mostrarTipoSensores() y mostrarUsuarios(); la primera realiza la conexión a la base de datos, la segunda sirve para llenar el listado de los diferentes tipos de sensores, y la última llena el listado de los usuarios creados en el sitio web.

El archivo "**page-grc-consultar.php**", hace uso de otros archivos PHP, como son "header-grcconsultar.php" y "footer-grc.php". Esta página muestra los gráficos generados según el identificador de la traza, e invoca a "header-grcconsultar.php" con el fin de inicializar las particularidades de diseño (CSS) y su correspondientes funciones javascript.

Respeto a las funciones javascript utilizadas para esta página, éstas son iguales a las del archivo "header-grc.php", con la diferencia que el método "GenerarGraficoClick(Codigo)" no genera nuevos gráficos, solo realiza la acción de mostrar los gráficos en cada una de sus pestañas.

El archivo "**page-grc-informe.php**", hace uso de los archivos PHP, como son "header-grcinforme.php" y "footer-grc.php". Esta página muestra todos los gráficos generados de una traza específica. La diferencia con la página anterior radica en que estos gráficos no son mostrados en una pestaña. Esta clase invoca a "header-grcinforme.php" con el fin inicializar las particularidades de diseño (CSS) y su correspondientes funciones javascript.

Entre las funciones javascript utilizadas para esta página tenemos:

- Función "GenerarGraficoClick(Codigo)".- Muestra los gráficos según el identificador seleccionado (traza). Esta función invoca un objeto tipo AJAX a través del método POST al módulo "grc-returngrafico.php" que devuelve los gráficos generados.

- Adicionalmente incluye los métodos "(cboTipoSensores.change)", "(cboUsuarios.change)" y "(grc-grid.on.click)", así como "Llenar_Grid()". Estos métodos tienen comportamientos similares a los métodos descritos en el archivo "header-grc.php".

Finalmente, el modelo de la base de datos MySQL para el sitio web se muestra en la figura 5.7. Cabe indicar que, en este diagrama, solo se muestran las tablas que tienen relación con el sistema propuesto.

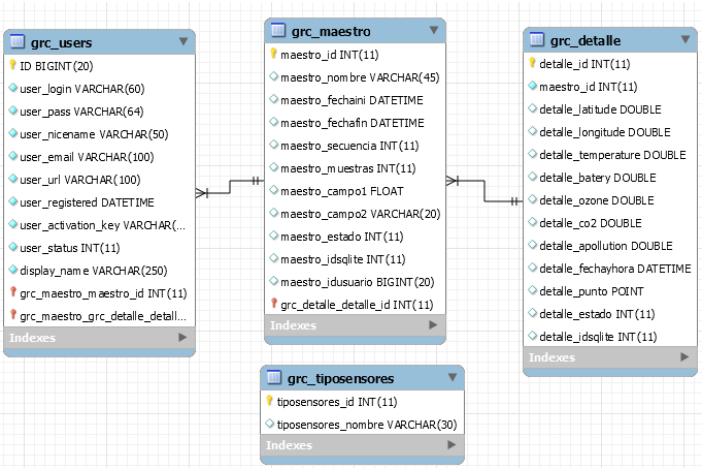


Figura 5. 7.- Modelo entidad-relación del sistema web.

5.3 Interfaz

La interfaz de los sistemas móvil y servidor tiene un aspecto similar, y la intención en general ha sido la de que esta interfaz sea simple y muy intuitiva, especialmente en lo que respeta al usuario móvil. De parte del administrador del sistema, el cual tendrá acceso al servidor web, se busca una interfaz que ofrezca un equilibrio entre simplicidad y funcionalidad, que será bastante más compleja en el caso del servidor, especialmente en lo que respeta a opciones en el tratamiento de datos.

5.3.1 Sistema Móvil

La pantalla principal de la aplicación móvil se muestra en la figura 5.8. La interfaz tiene 2 botones principales: iniciar (1) y parar (5). Al presionar el botón iniciar (1) el sistema empieza el servicio de lectura y captura de datos, lo cual habilita los botones gráficos (2) y mapas (3) que son opciones de consulta que se muestran en la figura 5.9; mientras se esté ejecutando la aplicación bajo esta opción, las consultas presentadas son en tiempo real. El botón (4) permite visualizar

un listado de las trazas grabadas en la base de datos. Esta acción solo se puede realizar si el servicio de captura de datos está suspendido. El botón (5) permite finalizar la captura de datos y habilita la actividad de guardar datos (ver figura 5.10). Finalmente la opción (0), que se encuentra en la parte superior de la barra de la pantalla principal, muestra y modifica las configuraciones principales del sistema.

Figura 5. 8.- Mobile Eco Sensor – Pantalla principal.

En la figura 5.9 se observa que el sistema ofrece dos opciones para mostrar gráficos. Cada opción presenta un listado mediante el cual el usuario móvil debe seleccionar qué tipo sensor desea visualizar en el gráfico. Los tipos de sensores disponibles para la opción (1) son CO_2, ozono, temperatura y contaminantes del aire.

La opción (1) muestra el recorrido (latitud y longitud) donde se encuentra el usuario realizando las lecturas, y el mapa muestra un trayecto que varía su color de acuerdo a la lectura entregada por el sensor. Dicha información se muestra en tiempo real.

La opción (2) presenta un listado donde se pueden seleccionar algunos de los sensores disponibles: CO_2, ozono, temperatura, contaminantes del aire y el indicador de batería. Al realizar la selección se muestran los gráficos de líneas que están en función del tiempo. Para los gráficos CO_2, ozono y contaminantes del aire se muestran los límites permitidos, los cuales varían entre bueno, admisible, malo y crítico.

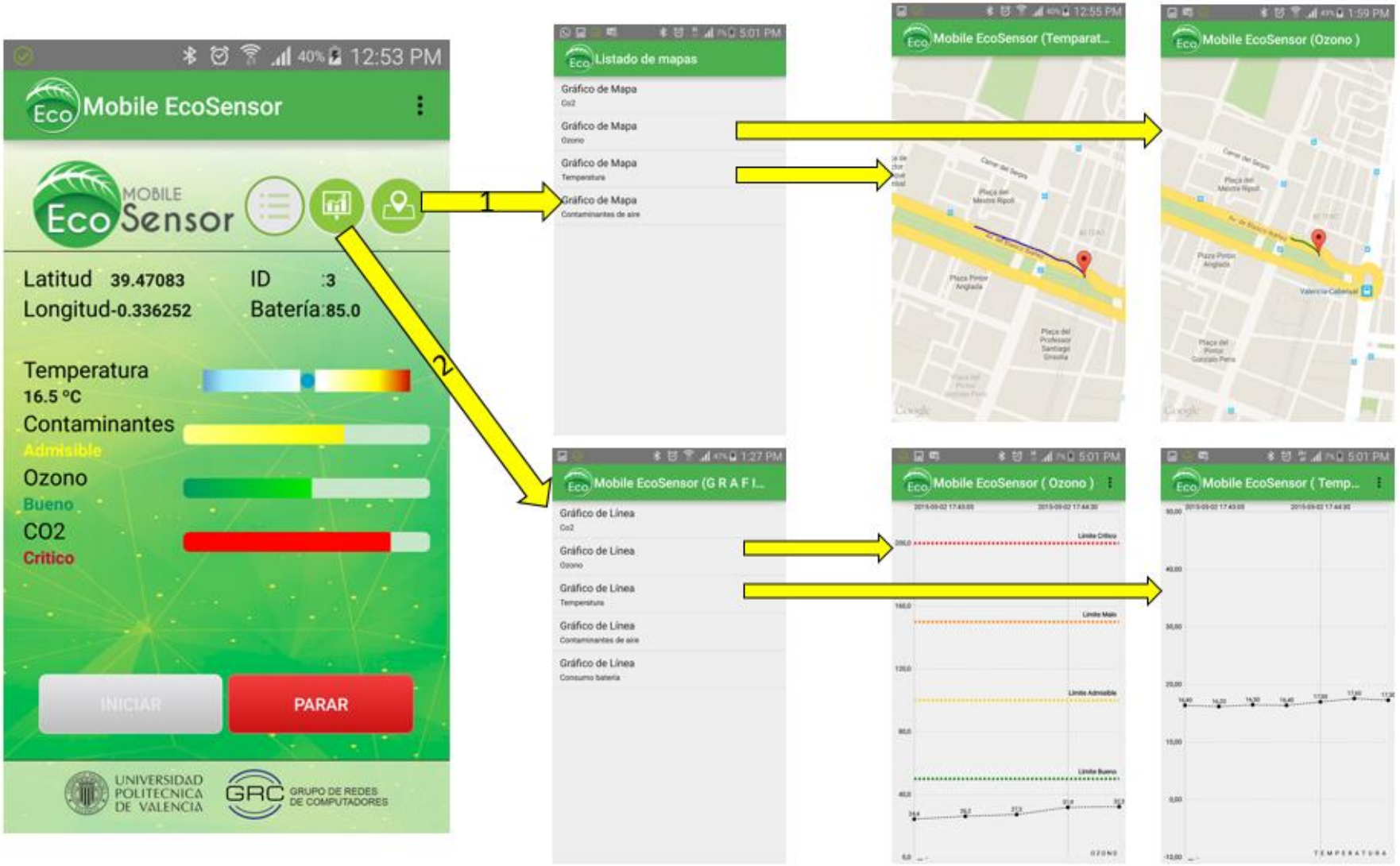


Figura 5. 9.- Mobile Eco Sensor – Consulta en tiempo Real.

En la figura 5.10, al presionar el botón “parar” (1), se presenta una actividad que permite al usuario grabar el recorrido efectuado. Esta actividad describe la fecha y hora inicial y final, el tiempo transcurrido, el total de muestras, y el resumen de cada sensor leído, mostrando su valor medio(avg), máximo(max) y mínimo(Min).

Figura 5. 10.-Mobile Eco Sensor - Grabar traza

En la figura 5.11 se muestra la opción que permite al usuario consultar un registro almacenado en la base de datos. El proceso consiste en seleccionar uno de los registros del listado, y si el usuario desea visualizar los gráficos deberá proceder como indicado anteriormente. Esta actividad también permite eliminar la traza de la base de datos y realizar el envío de la información al servidor web.

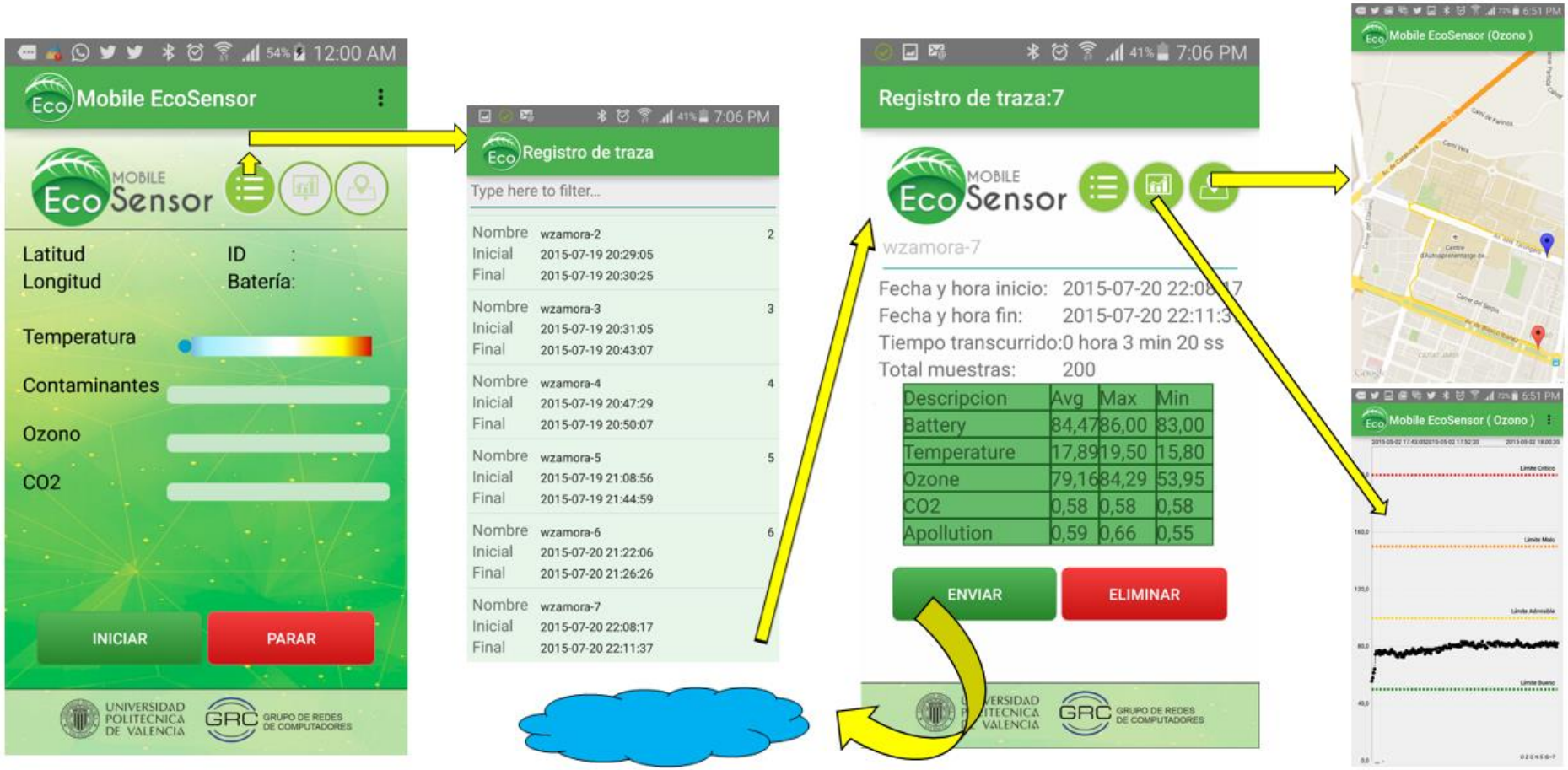


Figura 5. 11.- Mobile Eco Sensor – Consultar datos.

5.3.2 Sistema Web

El sitio web muestra una interfaz simple e intuitiva (ver figura 5.12). En su página principal se muestra una descripción de la solución planteada. Entre las opciones de menú se encuentra el enlace para iniciar sesión. Esta página permite acceder a un nuevo menú de opciones destinada a generar y visualizar los gráficos de contaminación respectivos. Esta acción está vinculada a cada una de las trazas almacenadas en su base de datos. Las opciones que se muestran después de iniciar sesión son: Generar gráfico, Informe individual e Informe general.

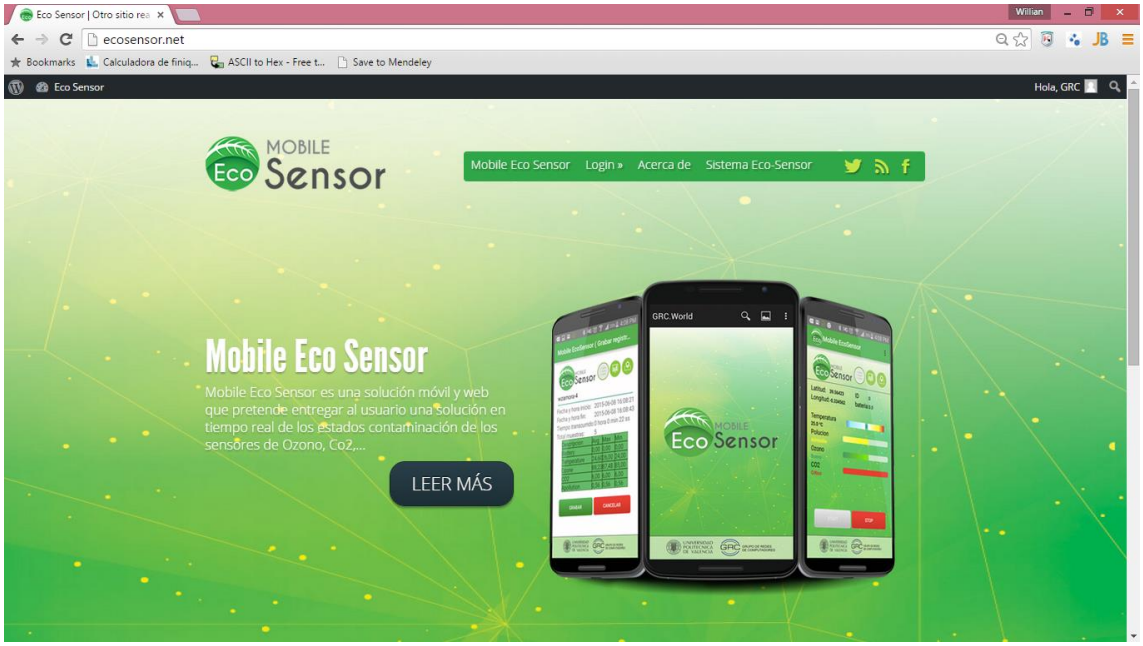


Figura 5. 12.- Página principal del sistema web.

La opción Generar gráfico permite crear gráficos relativos a una traza específica (ver figura 5.13). La página permite buscar filtrando por tipo de sensor y por el nombre de usuario que haya generado la traza. Esta página está diseñada para configurar el gráfico a mostrar en base a los parámetros de entrada. Estas configuraciones están asociadas a los gráficos del tipo Kriging, y HeatMap. Entre los parámetros permitidos se tiene:

- Ajustes.- Permite ajustar los valores del tipo de sensor.
- Modelo.- Permite el ajuste automático del variograma de acuerdo a los modelos seleccionados (STE, MAT, SPH, GAU, EXP). Si se seleccionan varios o todos, el sistema automáticamente elije el modelo que mejor se ajusta según los datos obtenidos.
- Resolución.- Permite dar calidad a la imagen, y se recomienda 80. Cuanto mayor sea el valor de la resolución, mayor será el tiempo de procesamiento al mostrar el gráfico.
- Mínimo.- Permite crear un filtro para los valores del sensor según un valor mínimo digitado. El valor por defecto es 0.

- Numero de Colores.- El número de colores a mostrar. Valor mínimo = 4, valor máximo = 12.
- Datos Alpha.- Toma un valor promedio de acuerdo a la fluctuación de datos. Permite disminuir la oscilación. Valor mínimo=1, valor máximo = 30.

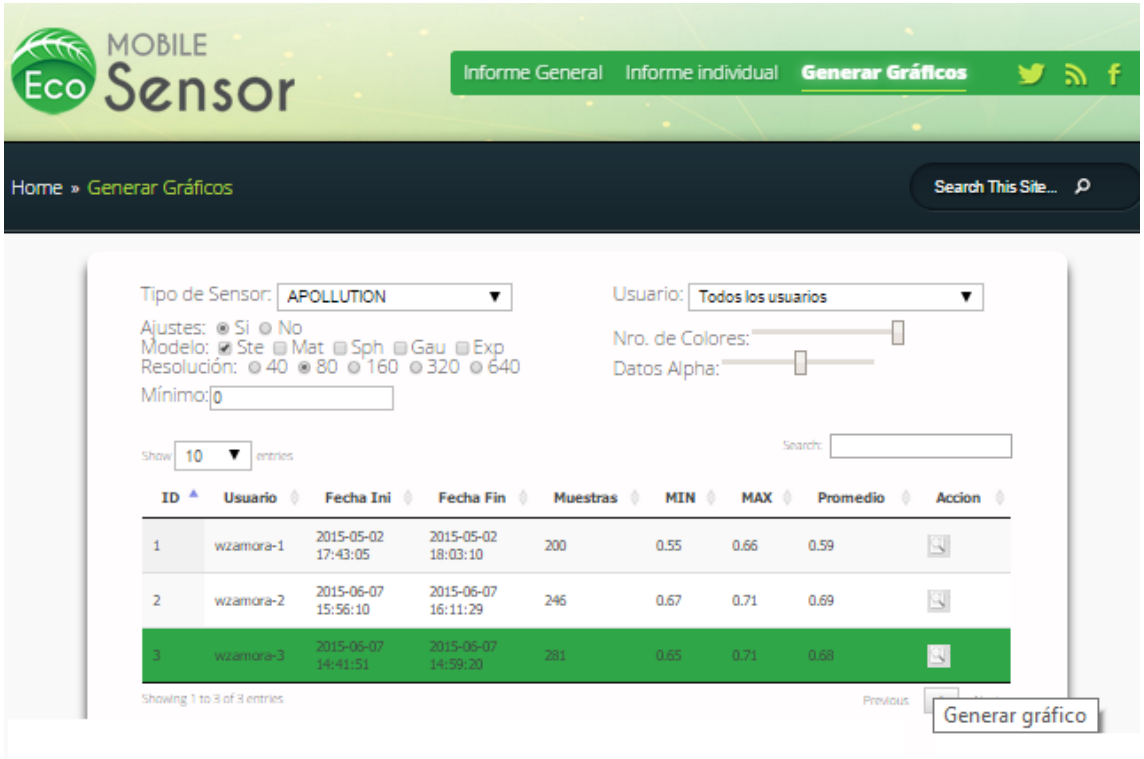


Figura 5. 13.- Página web - Generar Gráfico.

El informe individual se presenta a través de pestañas gráficas como: HeatMap, Kriging, Plot y BoxPlot de acuerdo a la traza seleccionada y al tipo de sensor, respectivamente. En la figura 5.14 se observa un ejemplo del gráfico HeatMap, de la traza con ID número 3. La variación de temperatura correspondiente a un recorrido de 281 muestras en el centro de la ciudad de Valencia.

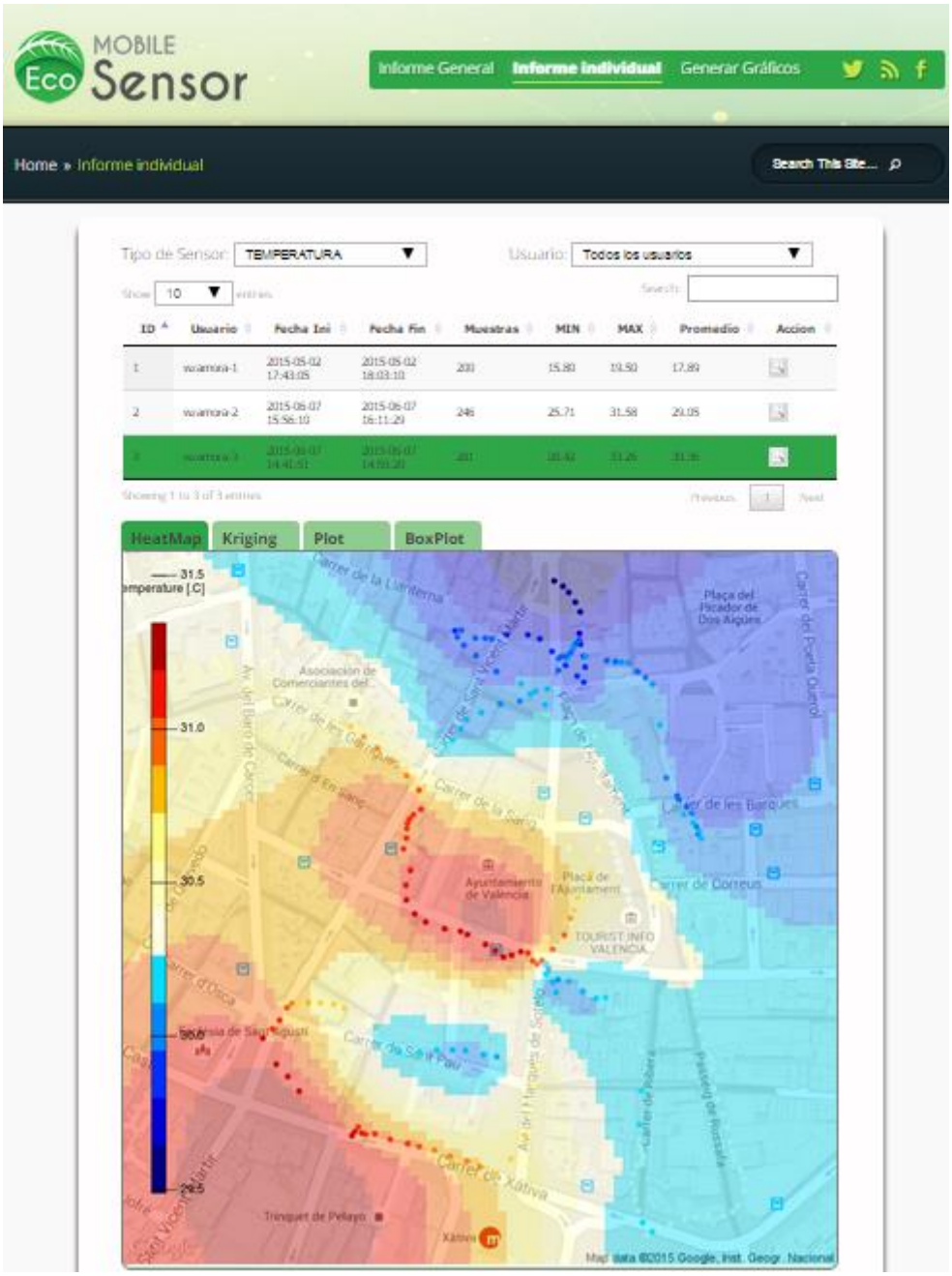


Figura 5. 14.- Página web - Informe Individual.

El informe General (ver figura 5.15) presenta un listado general del comportamiento de los sensores: CO_2, ozono, contaminantes del aire, y temperatura. Para mostrar los resultados es necesario seleccionar una traza. Para cada sensor se muestran gráficos tales como: HeatMap, Kriring, Plot y BoxPlot. Adicionalmente la página web da la posibilidad de que se muestre un tipo de sensor en particular.

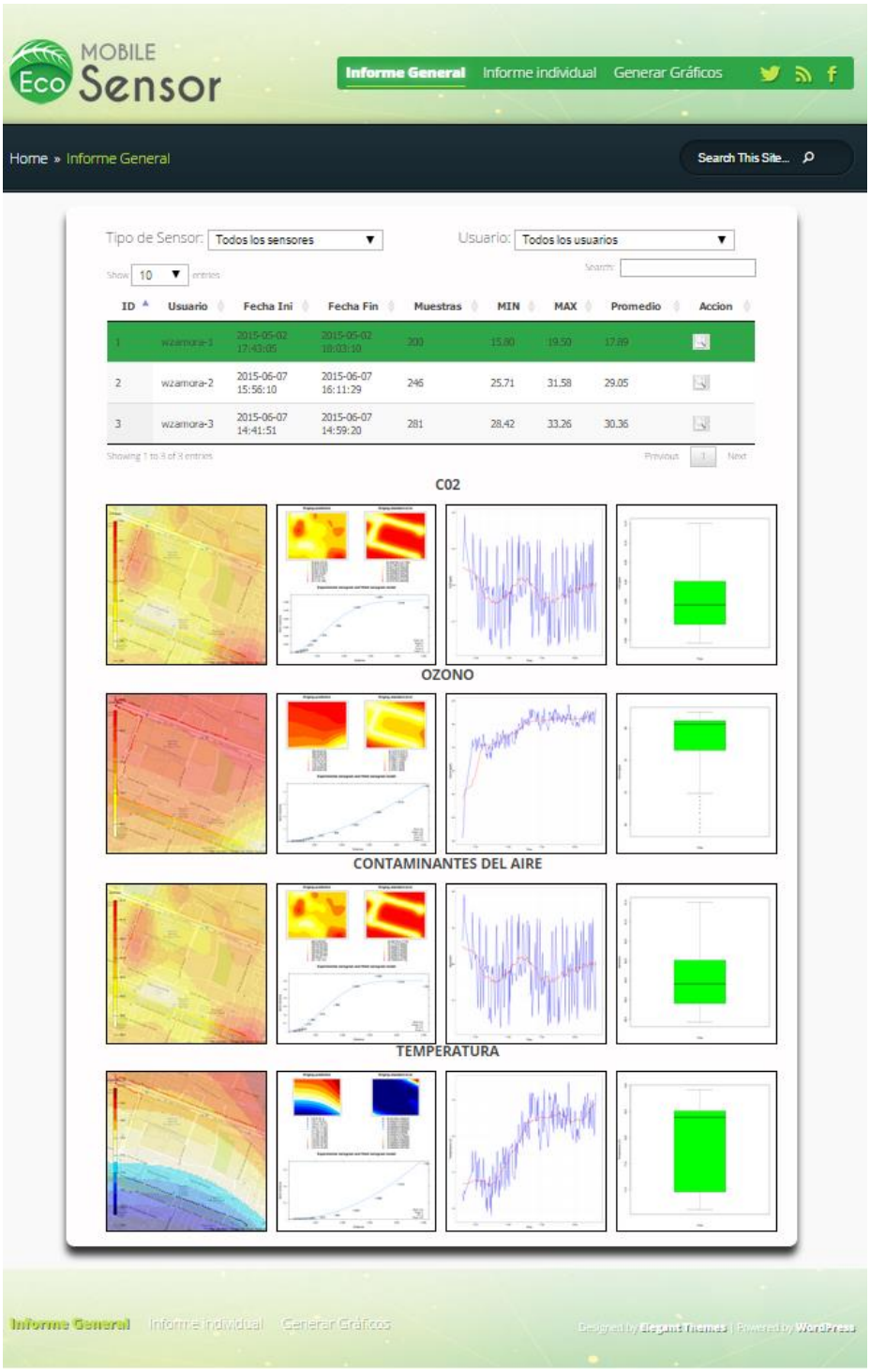


Figura 5. 15.- Página web - Informe General.

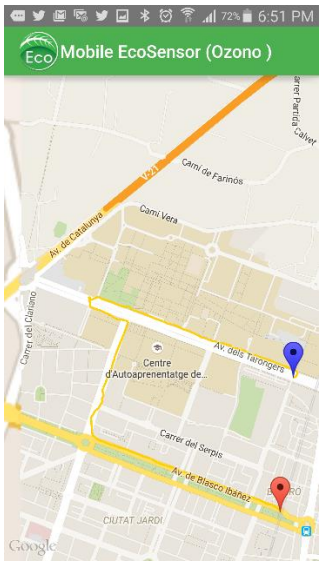
Capítulo VI: Pruebas

En este capítulo se detallan las pruebas realizadas en diferentes puntos de la ciudad de Valencia, y se presentan los resultados obtenidos. Se eligieron 3 trazas distintas que corresponden a trayectos representativos de entre las pruebas realizadas. A continuación se presentan los resultados para el aplicativo móvil y los datos mostrados por el sistema web. Destacar que los resultados obtenidos en cuanto a niveles de ozono son fiables ya que se hizo un trabajo previo de calibración de sensor [34].

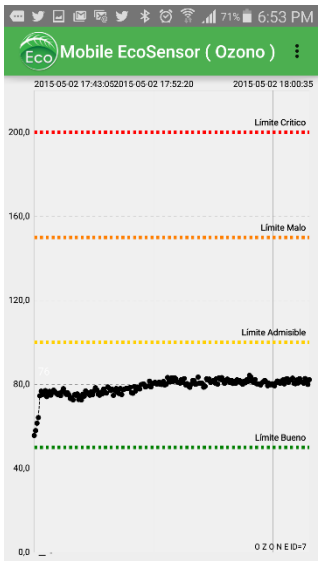
6.1 Pruebas del sistema móvil

Instalada la aplicación en un teléfono con sistema operativo Android, se procede a ubicar el sensor Libelium y la Raspberry Pi en una Valenbisi (ver figura 6.1 a). El sistema móvil tiene algunos parámetros de configuración que ya se encuentran preestablecidos. Unos de estos parámetros de configuración son los límites que van a tener cada sensor, que tienen carácter cualitativo, y se muestran en la pantalla al iniciar la captura de datos y al mostrar los gráficos (ver capítulo anterior). Estos valores están basados en índices de calidad del aire. En el desarrollo de la aplicación móvil se establecieron límites, para los valores máximos de 50, 100, 150 y 200, que en su interfaz lo muestra como bueno, admisible, malo, y crítico respectivamente, para cada sensor estudiado. Dicho límites no son aplicados al sensor de temperatura. Adicionalmente se muestra cada uno de estos valores con un color que ilustra el grado de peligrosidad asociado.

(a) Sistema en Valenbisi.



(b) Sistema móvil recorrido realizado.

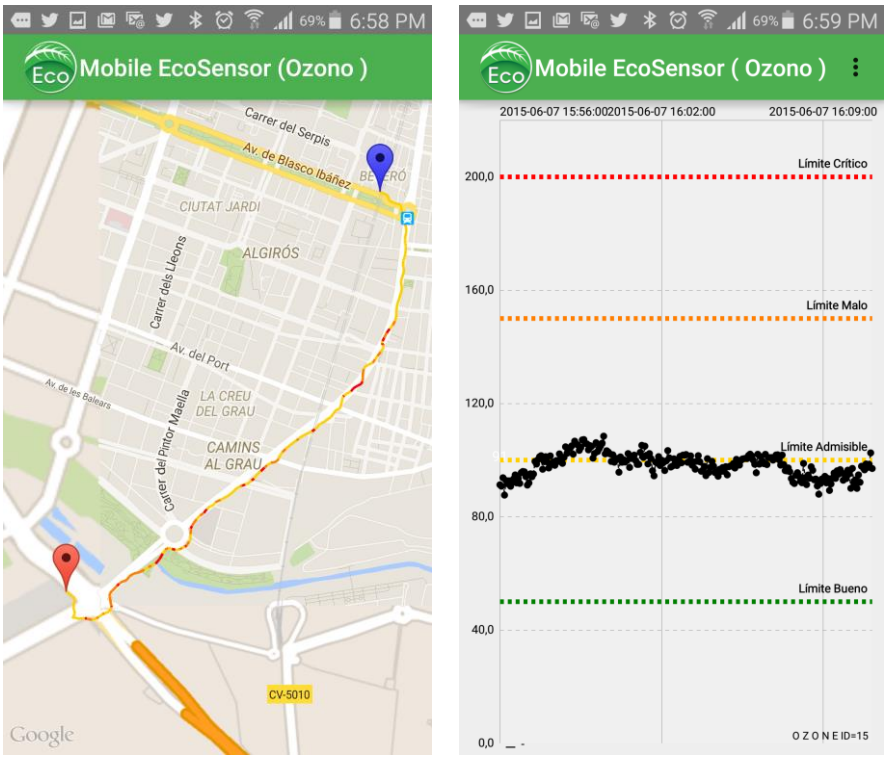


(c) Sistema móvil gráfico en función del tiempo.

Figura 6. 1.- Recorrido realizado # 1 e instalación de equipo.

Se eligieron diferentes rutas, y cada trayecto culminado, fue almacenado en la base de datos del dispositivo Android, para posteriormente ser enviado al servidor Web. En la figura 6.1.b, se muestra el recorrido #1 realizado desde Valencia-Cabanyal hasta las inmediaciones del campus de la Universidad Politécnica de Valencia, en la Av. Tarongers. El registro contabilizó un total de 200 muestras en un intervalo de 5 segundos para cada una de ellas. El recorrido se muestra con un color amarillo, lo que indica que el nivel de ozono es admisible para la salud humana. En la figura 6.1.c se evidencia que dicho valores están por encima del límite bueno (color verde). Adicionalmente la aplicación también evidencia los valores máximos, mínimos y promedio encontrados en el trayecto.

En la figura 6.2.a, se evidencia el recorrido #2 desde el Centro Comercial El SALER (punto Rojo) hasta la zona del Valencia-Cabanyal (punto azul). El total de las muestras tomadas para este recorrido fue de 246 con un intervalo de tiempo de 5 segundos. Dicho trayecto evidencia una intermitencia en los colores de valores entre admisible y malo (ver figura 6.2b). Estos resultados demuestran que a través de la aplicación móvil, ha sido posible visualizar el nivel de contaminación en un trayecto específico, de la ciudad de Valencia, y de manera móvil.



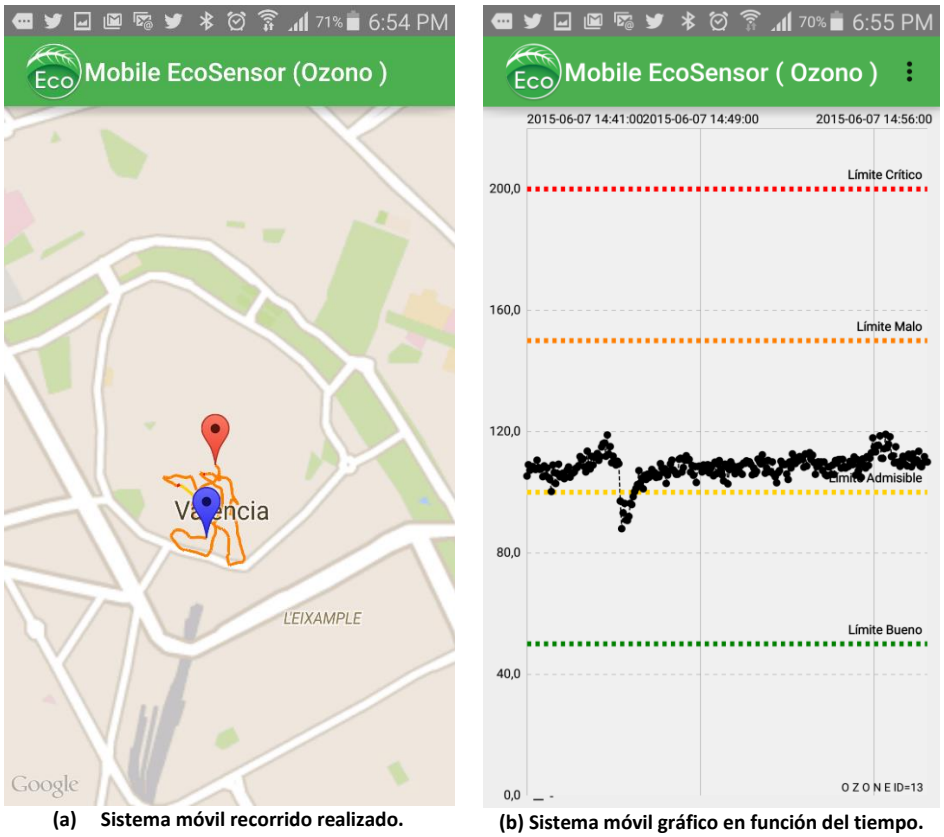
(a) Sistema móvil recorrido realizado.

(b) Sistema móvil gráfico en función del tiempo.

Figura 6. 2.- Sistema móvil recorrido realizado # 2 y gráfico.

Por último se hizo un recorrido aleatorio en una parte del centro de la ciudad de Valencia (ver figura 6.3 a y b), donde los valores están por encima de los máximos admitidos (color amarillo)

en casi todo el recorrido, señalando que la ciudad de Valencia evidencia ciertos puntos con niveles de contaminación alto, como se evidenció en la figura. El valor máximo en este trayecto es de 112.21 ppb, y el valor mínimo fue de 91.29 ppb. El promedio registrado fue de 108.62 ppb. El total de la muestras fue de 281, a igual intervalo que las anteriores.



(a) Sistema móvil recorrido realizado.

(b) Sistema móvil gráfico en función del tiempo.

Figura 6. 3.- Sistema móvil recorrido realizado # 3 y gráfico

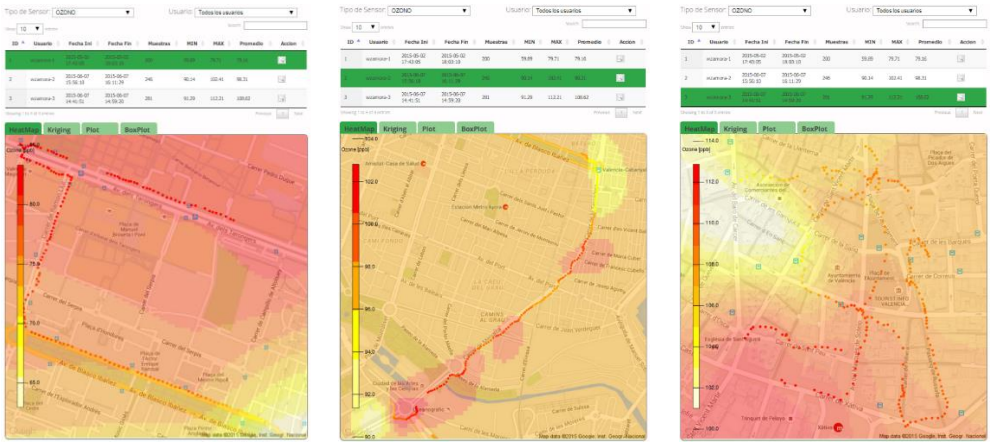
Adicionalmente se hizo la lectura de 2160 muestras a intervalos de 5 segundos, lo que representa 3 horas de captura de datos. Los datos capturados fueron enviados al servidor web sin ninguna novedad, y el medio de comunicación usado fue WiFi.

6.2 Pruebas del sistema web

El sistema web recibe los datos provenientes del sistema móvil. El sistema web genera los gráficos para las diferentes trazas a través de la opción “Generar Gráficos” del sitio web. Los resultados obtenidos se los puede visualizar en la misma opción, o través de los informes generales e individuales, respectivamente. Para estas pruebas el autor considera pertinente analizar los recorridos vistos en el apartado anterior, pero con el tratamiento de datos respectivo.

El estilo para este apartado será la revisión de las 3 trazas para una misma gráfica.

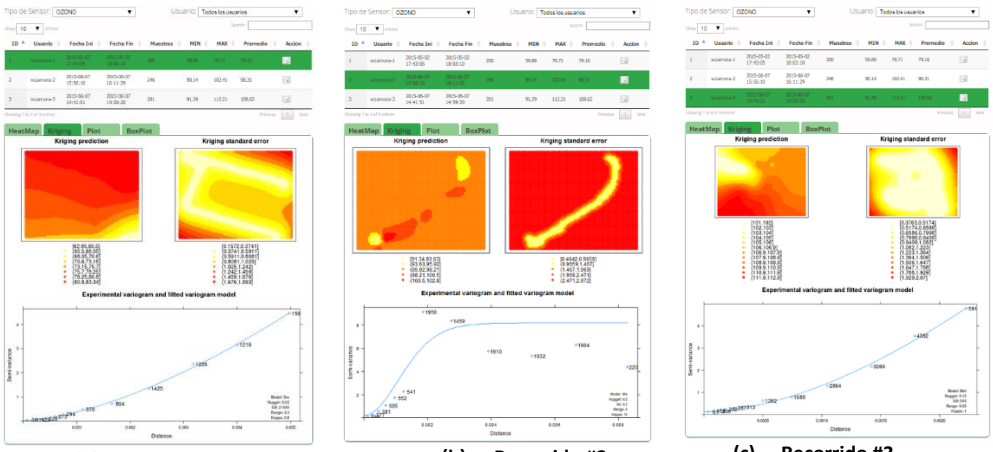
En las figuras 6.4 a b y c se muestran los diferentes recorridos descritos en la sección anterior. Para cada uno de ellos se evidencia su respectivo mapa de calor, en la cual se muestra la distribución de los datos del sensor dentro de un área específica, utilizando técnicas de kriging. En estos resultados el comportamiento del recorrido número #3 presenta un mayor índice de ozono, recorrido que corresponde a la parte céntrica de la ciudad de Valencia.



(a) Recorrido #1 (b) Recorrido #2 (c) Recorrido #3

Figura 6. 4.- Gráficos heatmap para los recorridos realizados.

En las figuras 6.5 a, b y c se muestran los valores calculados dentro del área recorrida pero sin el mapa de fondo. También se evidencia el error estándar sobre dicha área. En el recorrido #3 presenta mayor índices de contaminación de partículas de ozono sobre esta área.

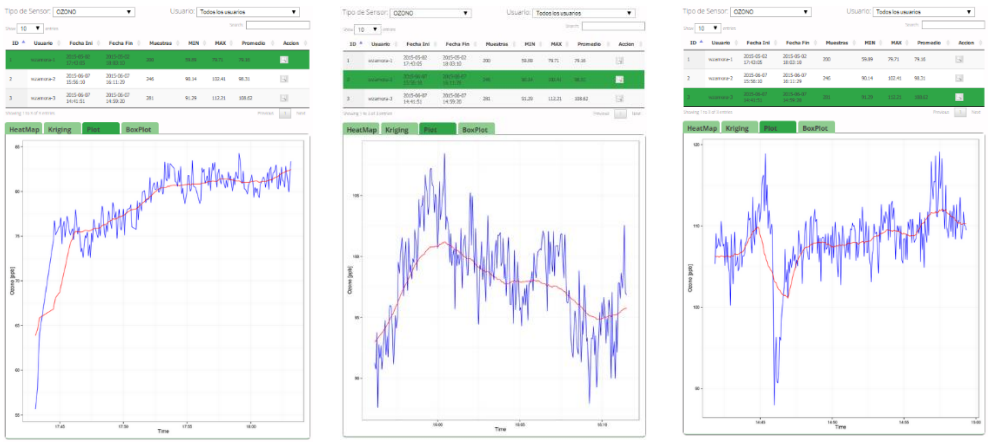


(a) Recorrido #1 (b) Recorrido #2 (c) Recorrido #3

Figura 6. 5.-Gráficos kriging para recorridos realizados.

En las figuras 6.6. a, b, y c, se muestran las medidas entregadas por el sensor en función del tiempo; adicionalmente se le agrega un filtro con la finalidad de agregar una línea con menor variación. En la figura 6.6c, se evidencia una mayor variación con respecto a los recorridos 2 y 1.

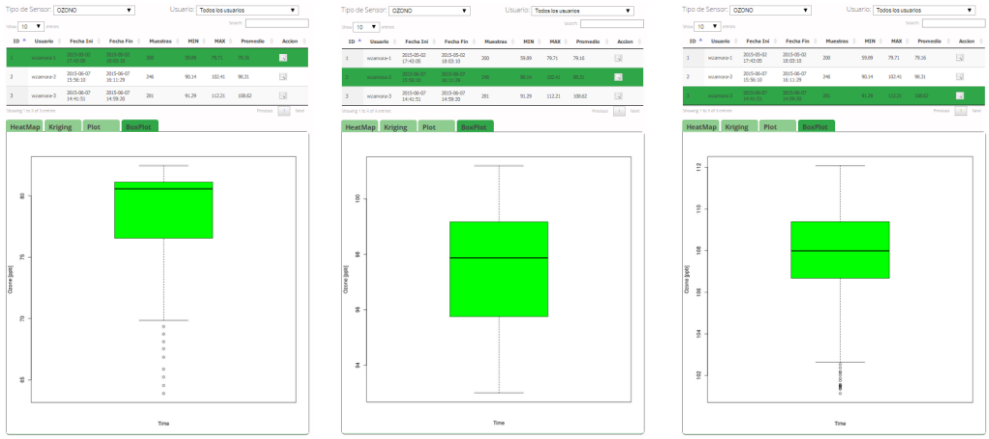
Estos gráficos, al ser comparados con los gráficos mostrados en la aplicación móvil, evidencian un comportamiento similar.



(a) Recorrido #1 (b) Recorrido #2 (c) Recorrido #3

Figura 6. 6.-Gráficos plot para recorridos realizados.

El boxplot o diagrama de caja es un gráfico basado en cuartiles que suministra información sobre los valores mínimo y máximo, cuartiles y mediana. Adicionalmente muestra la existencia de valores atípicos y la simetría de la distribución. En la figura 6.7, se evidencia que el recorrido #3 (ver figura 6.7.c) se caracteriza por unos niveles de polución más elevados con respecto a los otros. Se evidencia, también algunos valores atípicamente bajos para los recorridos 1 y 3.



(a) Recorrido #1 (b) Recorrido #2 (c) Recorrido #3

Figura 6. 7.-Gráficos boxplot para recorridos realizados.

Finalmente se pudo comprobar que las pruebas realizadas en las diferentes ubicaciones a través del aplicativo móvil registraron valores concordantes con los presentados en las figuras anteriores. Con respecto a las pruebas del sistema web se evidencio la facilidad de uso y la operatividad del sistema.

Capítulo VII: Conclusiones y Trabajo futuro

Como se ha mencionado en la introducción y antecedentes de este Trabajo Fin de Master, hoy en día es importante disponer de herramientas que nos faciliten información acerca de los niveles de contaminación relativos a nuestro entorno, ya que niveles de contaminación elevados pueden repercutir en enfermedades para el ser humano.

En este trabajo se han unificado diferentes tecnologías con el fin de crear un sistema que permita mostrar, en tiempo real, los niveles de contaminación correspondientes a una ubicación específica. La solución Mobile Eco Sensor consta de dos sistemas principales: el Sistema móvil, que combina un Sensor Waspmote Libelium, un Gateway Raspberry Pi y la aplicación que se ejecuta en una terminal Android, y el Sistema Web, el cual realiza el tratamiento de los datos y permite visualizar los niveles de contaminación en el área monitorizada con todo detalle.

El sistema móvil pudo ser instalado con facilidad en una Valenbisi, e incluso llevado a cuestas en una mochila, lo que nos permitió visualizar en tiempo real valores cualitativos de los niveles de contaminación. Se comprobó su efectividad capturando datos desde diferentes puntos de la ciudad de Valencia.

En lo que respeta al Sistema Web, se realizó el tratamiento de datos a través de la herramienta estadística R, y se pudo mostrar los resultados obtenidos mediante diferentes tipos de gráficos, incluyendo mapas de calor, que están organizados mediante un informe detallado disponible para el administrador del sistema.

Cabe destacar que los objetivos propuestos para este Trabajo Fin de Máster se han cubierto en su totalidad, estando el sistema plenamente operativo, y siendo barato y fácil de replicar.

Globalmente consideramos que, el sistema propuesto puede ser muy útil para la administración pública al permitir monitorizar una población o ciudad de manera muy barata y rápida, habiendo además la posibilidad de integrar los sensores móviles en el transporte público, lo que permitiría realizar un análisis continuo de los niveles de polución en una ciudad.

En lo que respeta a trabajo futuro, se planteará introducir mejoras a la aplicación para dar soporte a otros tipos de sensores, y para permitir descargar y mostrar los gráficos generados del sistema web.

A nivel Hardware se desarrollará un dispositivo sensor propio, mucho más pequeño que el utilizado en la solución planteada.

Bibliografía

[1] F. Ballester, M. Saez, A. Daponte, J. M. Ordóñez, M. Taracido, K. Cambra, F. Arribas, J. Bellido, J. Guillén, I. Aguinaga, A. Cañada, E. Lopez, C. Iñiguez, P. Rodríguez, S. Pérez, M. Barceló, R. Ocaña y E. Aránguez, «EL PROYECTO EMECAS: PROTOCOLO DEL ESTUDIO MULTICÉNTRICO,» *Especialidad Salud Pública,* pp. 229-242, 2005.

[2] A. M. Muñoz D., J. J. Paz V. y C. M. Qurioz P., «Efectos de la contaminación atmosférica sobre la salud de adultos que laboran en diferentes,» *Fac. Nac. Salud Pública,* vol. 2, nº 25, pp. 85-94, 2007.

[3] O. C. Climático, «Comisión Lancet: cambio climático, amenaza mortal para la salud a la vez que oportunidad,» 23 06 2015. [En línea]. Available: http://newsroom.unfccc.int/es/bienvenida/comision-lancet-cambio-climatico-amenaza-mortal-para-la-salud-a-la-vez-que-oportunidad/. [Último acceso: 20 06 2015].

[4] Wikipedia, «Protocolo de Kioto sobre el cambio climático,» Wikipedia, 24 06 2015. [En línea]. Available: https://es.wikipedia.org/wiki/Protocolo_de_Kioto_sobre_el_cambio_clim%C3%A1tico. [Último acceso: 25 06 2015].

[5] P. E. Consejo, «DIRECTIVA 2008/50/CE DEL PARLAMENTO EUROPEO Y DEL CONSEJO,» 21 05 2008. [En línea]. Available: http://eur-lex.europa.eu/legal-content/ES/TXT/PDF/?uri=CELEX:32008L0050&from=EN. [Último acceso: 15 05 2015].

[6] A. E. d. M. Ambiente, «El medio ambiente en Europa: Estado y perspectivas 2015,» AEMA, Copenhague, 2015.

[7] G. Valenciana, «Red Valenciana de Vigilancia y Control de la Contaminación Atmosférica,» [En línea]. Available: http://www.citma.gva.es/web/calidad-ambiental/red-valenciana-de-vigilancia-y-control-de-la-contaminacion-atmosferica. [Último acceso: 20 06 2015].

[8] A. Q. I. EUROPE, «AIR QUALITY IN EUROPE - Contaminación,» 2007. [En línea]. Available: http://www.airqualitynow.eu/es/pollution_home.php. [Último acceso: 22 05 2015].

[9] X. Querol, V. Mar, T. Moreno y A. Alastuey, «Bases científico-técnicas para un Plan Nacional de Mejora de la Calidad del Aire,» Edición a cargo de Cyan, Proyectos Editoriales, S.A., Madrid, 2012.

[10] A. Quality, «Air Quality Index (AQI) Basics,» 16 03 2015. [En línea]. Available: http://airnow.gov/index.cfm?action=aqibasics.aqi. [Último acceso: 4 05 2015].

[11] A. Q. i. Europe, «Definición de índices,» [En línea]. Available: http://www.airqualitynow.eu/es/about_indices_definition.php. [Último acceso: 03 05 2015].

[12] A. P. i. E. R.-t. A. Q. I. V. Map, «Air Pollution in Europe: Real-time Air Quality Index Visual Map,» [En línea]. Available: http://aqicn.org/map/europe/es/#@g/44.2682/31.89/4z. [Último acceso: 7 05 2015].

[13] A. Q. i. Europe, «Efecto sobre la salud,» 01 01 2007. [En línea]. Available: http://www.airqualitynow.eu/es/pollution_health_effects.php. [Último acceso: 15 04 2015].

[14] R. Beelen, G. Hoek, P. Fischer, P. A van den Brandt y B. Brunekreef, «Estimated long-term outdoor air pollution concentrations in a cohort study,» *Science of The Total Environment,* vol. 40'7, nº 6, pp. 1852-1867, 2009.

[15] C. Carlos y B. Ducourthial, «On the Use of Mobile Sensors for Estimating City-Wide Pollution Levels,» *IWCMC,* 2015.

[16] B. Matija y S. Višnja, «Urban Sensing – Smart Solutions for Monitoring Environmental Quality: Case Studies from Serbia,» *ISOCARP Congress 2012,* 2012.

[17] I. Zualkernan y F. Aloul, «A Mobile GPRS-Sensor Array for Air Pollution Monitoring,» *IEEE Sensors Journal,* vol. 10, nº 10, 2010.

[18] D. Hasenfratz, O. Saukh, S. Sturzenegger y L. Thiele, «Participatory Air Pollution Monitoring Using Smartphones,» *2nd International Workshop on Mobile Sensing,* vol. 2, nº 2, 2012.

[19] U. d. Houston, «Google play,» 23 06 2015. [En línea]. Available: https://play.google.com/store/apps/details?id=edu.uh.cpl.

[20] P. C. D. Thailand, «Google Play,» 13 01 2015. [En línea]. Available: https://play.google.com/store/apps/details?id=com.twofellows.airforarseanmobile. [Último acceso: 25 05 2015].

[21] L. Qiang, «Google Play - Air Quality China,» 04 01 2015. [En línea]. Available: https://play.google.com/store/apps/details?id=com.cas.airquality. [Último acceso: 22 05 2015].

[22] Cercl'Air, «Google Play - airCHeck,» 03 12 2014. [En línea]. Available: https://play.google.com/store/apps/details?id=ch.ti.oasi.android.airquality. [Último acceso: 15 06 2015].

[23] T. ARUCAS, «La arquitectura de Android,» 18 10 2012. [En línea]. Available: http://catedratelefonica.ulpgc.es/blog/arucas/2012/10/18/la-arquitectura-de-android/. [Último acceso: 15 06 2015].

[24] A. Developers, «Android Developers,» Google, [En línea]. Available: http://developer.android.com/reference/android/app/Activity.html. [Último acceso: 10 07 2015].

[25] T. A. S. Foundation, «The Apache Software Foundation,» [En línea]. Available: http://www.apache.org/foundation/. [Último acceso: 16 06 2015].

[26] P. Documentation, «Historia PHP,» [En línea]. Available: http://www.php.net/manual/es/history.php.php. [Último acceso: 11 07 2015].

[27] WordPress.org, «WordPress,» [En línea]. Available: https://wordpress.org/about/. [Último acceso: 10 07 2015].

[28] W. MySQL, «MySql,» 07 07 2015. [En línea]. Available: https://es.wikipedia.org/wiki/MySQL. [Último acceso: 11 07 2015].

[29] RStudio, «RStudio Products,» [En línea]. Available: https://www.rstudio.com/products/rstudio/. [Último acceso: 12 07 2015].

[30] P. Kinney, «Zigbee Technology: Wireless control that simply works,» *Communications design conference,* vol. 2, pp. 1-7, 2003.

[31] WIKIPEDIA, «Bluetooth,» 11 07 2015. [En línea]. Available: https://es.wikipedia.org/wiki/Bluetooth. [Último acceso: 20 07 2015].

[32] MIT.EDU, «An Introduction to Bluetooth Programming,» [En línea]. Available: http://people.csail.mit.edu/albert/bluez-intro/x232.html. [Último acceso: 15 02 2015].

[33] J. Canós, P. Letelier y M. Penades, «Métologias Ágiles en el Desarrollo de Software,» [En línea]. Available: http://www.carlosfau.com.ar/nqi/nqifiles/XP_Agil.pdf. [Último acceso: 15 07 2015].

[34] Ó. Alvear, C. Calafate, J.-C. Cano, P. Manzoni, E. Hernandez-Orallo y J. Herrera, «Metodología para la Monitorización de Ozono en Valencia mediante Sensores de gama baja,» *Jornadas de Computación Empotrada,* vol. VI, 2015.

[35] PG&E, «Data Center Best Practices Guide,» 10 02 2015. [En línea]. Available: http://www.pge.com/includes/docs/pdfs/mybusiness/energysavingsrebates/incentivesbyindustry/DataCenters_BestPractices.pdf.

[36] B. E. Mileage, «Google Play,» 09 01 2014. [En línea]. Available: https://play.google.com/store/apps/details?id=net.aboutbike.GreenBike. [Último acceso: 22 06 2015].

[37] F. V. d. C. González, «CAMBIO CLIMÁTICO Y PROTOCOLO DE KIOTO. CIENCIA Y ESTRATEGIAS.,» *Española de Salud Pública,* vol. 79, nº 2, 2005.

ANEXO A

Tabla 4.- Características técnicas del Sensor de Temperatura.

Descripción	Valores
Rango de medida	[-40°C, +125°C]
Voltaje salida (0°C)	500mV
Sensibilidad	10mV/°C
Precisión	±2ºC (intervalo 0ºC ~ +70ºC), ±4ºC (intervalo -40 ~ +125ºC)
Voltaje suministrado	2.3 ~ 5.5V
Tiempo respuesta	1.65 segundos (63% respuesta desde +30 a +125 ºC)
Consumo típico	6 µA
Consumo máximo	12 µA

Tabla 5.- Características técnicas del Sensor de Ozono.

Descripción	Valores
Gases:	O3
Rango de medida	10 ~ 1000ppb
Resistencia aire	3 ~ 60kΩ (típica 11 kΩ)
Sensibilidad	2 ~ 4 (típica 1.5 entre la resistencia a 100 ppm y 50ppm)
Voltaje suministrado	1.95V ~ 5V DC
Temperatura de funcionamiento	-30 ~ + 85°C
Tiempo respuesta	30 segundos
Consumo promedio	34 mA

Tabla 6.- Características técnicas del Sensor CO_2.

Descripción	Valores
Gases:	CO_2
Rango de medida	350 ~ 10000ppm
Voltaje a 350ppm	220 ~ 490mV
Sensibilidad	44 ~ 72mV (relación entre el voltaje a 350ppm y a 3500ppm).
Voltaje suministrado	5V ± 0.2V DC
Temperatura de funcionamiento	-10 ~ + 50°C
Tiempo respuesta	1.5 minutos
Consumo promedio	46 mA

Tabla 7.- Características Técnicas del Sensor Air Pollutants 2.

Descripción	Valores
Gases:	C_4H_{10}, CH_3CH_2OH, H_2,CO, CH_4
Rango de medida	1 ~ 100ppm
Resistencia aire	10 ~ 90kΩ
Sensibilidad	0.3 ~ 0.6 (relación entre la resistencia en 10 ppm de H2 y en el aire)
Voltaje suministrado	5V ± 0.2V DC
Temperatura de funcionamiento	-10 ~ + 40°C
Tiempo respuesta	30 segundos
Resistencia mínima de carga	0.45 kΩ
Consumo promedio	46 mA

ANEXO B

Escenario caso de uso - Aplicación móvil

Tabla 8.- Escenario - Iniciar lectura de datos

Caso de Uso - Iniciar lectura de datos	
Identificador	C.1.1
Precondición	Aplicación instalada, disponer de Bluetooth y que se encuentre emparejado con el Raspberry Pi.
Descripción	Al seleccionar el botón “iniciar”, se activa el servicio que solicita datos al Raspberry Pi. Procesa los datos emitidos. Almacena el registro en una base de datos temporal. Muestra información en pantalla.
Propósito	Iniciar lectura de datos.
Pos-condición	Disponibilidad del dispositivo Raspberry Pi y el sensor Libelium.

Tabla 9.- Escenario – Mostrar información

Caso de Uso – Mostrar información	
Identificador	C.1.2
Precondición	Que exista información de traza almacenada. Para visualizar datos en tiempo real el botón “iniciar” debe estar en ejecución.
Descripción	Al seleccionar el botón “lista”, muestra la información de las trazas almacenadas en su base datos.
Propósito	Mostrar información.
Pos-condición	Seleccionar tipo de sensor a mostrar.

Tabla 10.- Escenario – Modelo de Mapa.

Caso de Uso – Modelo de Mapa	
Identificador	C.1.2.1
Precondición	Que exista información de traza almacenada. Que el botón iniciar este activo.
Descripción	Existen dos opciones para visualizar los mapas. La primera opción es en tiempo real, cuando el botón iniciar está en ejecución. Al seleccionar el botón “mapa” seguidamente seleccionar el tipo de sensor que se desea visualizar. La segunda opción es mediante el caso de uso C.1.2.
Propósito	Mostrar mapa.
Pos-condición	Seleccionar tipo de sensor.

Tabla 11.- Escenario - Modelo de Gráfico.

Caso de Uso – Modelo de Gráfico	
Identificador	C.1.2.2
Precondición	Que exista información de traza almacenada. Que el botón iniciar esté en ejecución.
Descripción	Existen dos opciones para acceder a visualizar los gráficos: la primera opción es en tiempo real, cuando el botón iniciar está en ejecución. Se debe seleccionar el botón "grafico" y seguidamente seleccionar el tipo de sensor que se desea graficar. La segunda opción es mediante el caso de uso C.1.2.
Propósito	Mostrar gráfico.
Pos-condición	Seleccionar tipo sensor.

Tabla 12.- Escenario - Finalizar lectura de datos.

Caso de Uso – Finalizar lectura de datos	
Identificador	C.1.3
Precondición	Que el botón "iniciar" esté en ejecución.
Descripción	Seleccionar el botón "detener". Al realizar esta opción el sistema finaliza el servicio y deja de mostrar información. Seguidamente se presenta la actividad de grabar registros.
Propósito	Finalizar servicio de lectura de datos.
Pos-condición	Presenta la actividad de almacenar la información.

Tabla 13.- Escenario - Almacena registros.

Caso de Uso – Almacena registros	
Identificador	C.1.3.1
Precondición	Caso de uso C.1.3
Descripción	Esta actividad permite visualizar el resumen de los datos capturados, adicionalmente al seleccionar el botón "grabar" almacena en su base de datos interna los datos procesados.
Propósito	Almacenar registro de trazas en el dispositivo móvil.
Pos-condición	Regresa a la actividad inicial.

Tabla 14.- Escenario - Enviar datos a Servidor Web.

Caso de Uso – Enviar datos a Servidor Web	
Identificador	C.1.4
Precondición	Que exista información de traza almacenada.
Descripción	Esta actividad tiene relación con el caso de uso C.1.2. Al realizar consulta de una traza almacenada en la base de datos interna, se debe seleccionar el botón "enviar" para procesar el envío de datos al servidor web.
Propósito	Envía datos al servidor web.
Pos-condición	Regresa a la actividad inicial.

Tabla 15.- Escenario - Configuración general

Caso de Uso – Configuración general	
Identificador	C.1.5
Precondición	Ninguna.
Descripción	Configura nombre usuario, y su identificador personal. Configura parámetros para gráficos y mapas.
Finalizado	Usuario móvil al finalizar la actividad.
Pos-condición	Actividad inicial.

ANEXO C

Escenarios casos de uso - Aplicación Web

Tabla 16.- Escenario - Mostrar información.

Caso de Uso – Muestra información	
Identificador	C.2.1
Precondición	Aplicación instalada y servidor web activos
Descripción	Muestra información concerniente a la investigación realizada en este Trabajo Fin de Máster.
Propósito	Mostrar información.
Pos-condición	Disponibilidad de aplicación web.

Tabla 17.- Escenario - Procesa información.

Caso de Uso – Procesa información	
Identificador	C.2.2
Precondición	Servidor web y servicio web activo.
Descripción	Registra los datos emitidos por el la aplicación móvil.
Propósito	Procesar inserciones a la base de datos respectiva.
Pos-condición	Ninguna.

Tabla 18.- Escenarios - ingreso al sistema.

Caso de Uso – Ingreso al sistema	
Identificador	C.2.3
Precondición	Validación de usuario con sus respectivos privilegios.
Descripción	El sistema le solicitará al usuario credenciales de usuario y contraseña asignados previamente, para que pueda ser validado por el sistema.
Propósito	Acceder al sistema para administrar o realizar alguna tarea según el tipo de usuario.
Pos-condición	Que los datos introducidos sean los correctos. Validación de usuario y contraseña.

Tabla 19.- Escenario - genera gráfico.

Caso de Uso – Genera gráfico	
Identificador	C.2.3.1
Precondición	Acceso al sistema
Descripción	El sistema solicitará varios parámetros de configuración y una traza específica para generar el gráfico.
Propósito	Generar gráficos.
Pos-condición	Que los datos introducidos sean correctos.

Tabla 20.- Escenario informe general

Caso de Uso – Informe general	
Identificador	C.2.3.2
Precondición	Acceso al sistema
Descripción	El sistema presenta un reporte general de todos los sensores con sus respectivos gráficos según traza.
Propósito	Mostrar informe general.
Pos-condición	Seleccionar una traza.

Tabla 21.- Escenario - informe individual.

Caso de Uso – Informe individual	
Identificador	C.2.3.3
Precondición	Acceso al sistema.
Descripción	El sistema presenta un informe individual para cada sensor con sus respectivos gráficos según traza.
Propósito	Mostrar informe individual.
Pos-condición	Seleccionar una traza.

ANEXO D

Clase MainActivity

Atributos	**Descripción**
btnStart	Botón que inicia el proceso de captura de datos.
btnStop	Botón que finaliza el proceso de captura de datos.
btnLista	Realiza una llamada a la clase "ListaTraza". Muestra el listado de las trazas almacenada en la base de datos interna.
btnMaps	Realiza una llamada a la clase "MapaLista". Muestra un listado de opciones como temperatura, ozono, contaminantes y CO_2 para mostrar la actividad en mapas.
btnGrafico	Realiza una llamada a la clase "GraficoLista". Muestra un listado de opciones como temperatura, ozono, contaminantes, CO_2 y batería, de manera gráfica y variable en el tiempo.
MenuItem-Settings	Realiza una llamada a la clase "Preferencias". Realiza las configuraciones iniciales del sistema.
Métodos	**Descripción**
onCreateOptionsMenu(Menu)	Activa las opciones del menú en la barra de la actividad.
InsertAuxiliarSQLiteTask	Clase – interna. Realiza las inserciones de los datos a través de una tarea asíncrona.
onReceive(Context)	Clase interna. Servicio que está escuchando los datos emitidos por el Raspberry Pi a través de Bluetooth.
PasarDatos(MainActivity)	Recibe datos procesados después de ejecutar el servicio OnReceive.

Clase ServicioGRC

Atributos	**Descripción**
INTERVALO	Intervalo de tiempo en que se ejecuta el servicio
Métodos	**Descripción**
onStartCommand	Inicializa el servicio.
stopService	Finaliza el servicio, cancela el ciclo interno.
onDestroy	Elimina el servicio.
cerrarSocket	Cierra los sockets.
sendBtnMsg	Devuelve un "stream" datos.
receivebtMsg	Recibe un "stream" datos.

Clase Datos

Atributos	**Descripción**
Latitude	Inicializa y devuelve latitud.
Longitude	Inicializa y devuelve longitud.
Temperatura	Inicializa y devuelve temperatura.
Ozone	Inicializa y devuelve Ozono.
Co2	Inicializa y devuelve CO_2.
Apollution	Inicializa y devuelve Contaminantes del aire (Apollutans).
fechayhora	Inicializa y devuelve fecha y hora.
Métodos	**Descripción**
LlenarDatos(String)	Descompone el string datos devuelto por el ServicioGRC descrito anteriormente.

Clase GrabaTraza

Atributos	**Descripción**
btnGrabar	Permite grabar la traza en la base de datos SQLite, registrando fecha-hora inicial y final.
btnCancelar	Permite elegir entre salir de la actividad grabando registros o sin grabar.
btnMaps	Realiza una llamada a la clase "MapaLista". Muestra un listado de opciones como temperatura, ozono, contaminantes y CO_2 para mostrar la actividad en mapas.
btnGrafico	Realiza una llamada a la clase "GraficoLista". Muestra un listado de opciones como temperatura, ozono, contaminantes, CO_2 y batería, en gráficos según el tiempo.
Métodos	**Descripción**
InsertSensoresSQLiteTask()	Inserta registros en la base de datos SQLite, proceso que se ejecuta en segundo plano mediante un hilo secundario.
DProvider – onCreate()	Clase interna – Permite inicializar conexión con SQLite.
DProvider – uri-insert()	Clase interna – Permite insertar registros en SQLite.
DProvider – delete()	Clase interna – Permite eliminar registros de SQLite.
DProvider – query()	Clase interna – Permite consultar registros de SQLite.

Clase MuestraTraza

Atributos	**Descripción**
btnEnviar	Permite ejecutar el servicio web a través de un objeto JSONObject y una respuesta HTTP de tipo POST.
btnCancelar	Permite salir de esta actividad.
btnMaps	Realiza una llamada a la clase "MapaLista". Muestra un listado de opciones como temperatura, ozono, contaminantes y CO_2 para mostrar la actividad en mapas.
btnGrafico	Realiza llamada a la clase "GraficoLista". Muestra un listado de opciones como temperatura, ozono, contaminantes, CO_2 y batería, en gráficos según el tiempo.

Métodos	Descripción
TareaNube()	Clase interna - Inserta registros en la base de datos MySQL a través de un servicio web. Este proceso se ejecuta en segundo plano.
JSONParser - makeHttpRequest()	Clase externa – Permite enviar un HTTP POST al servidor Web.

Clase JSONParser

Atributos	Descripción
jObj	Inicializa objeto de tipo JSON.
Error	Control de errores para esta clase.
Métodos	**Descripción**
getJSONObject()	Retorna un objeto de tipo JSON.
getError()	Retorna error.
makeHttpRequest()	Realiza petición de tipo HTTP al servidor Web.

Clase Database

Métodos	Descripción
Database(Context)	Permite inicializar el objeto Database.
onCreate(SQLiteDatabase)	Método que crea las tablas correspondientes a la aplicación (ver figura 5.4). Este método utiliza la clase "Estructura" donde se encuentra la definición de las respectivas tablas.
onUpgrade(SQLiteDatabase ..)	Si existe una actualización en la estructura de la base de datos realiza los cambios, y en caso contrario omite la actualización.
insertMaestroSensores(ContentValues)	Permite la inserción de registros utilizando un objeto del tipo ContentValues.
insertAuxilarSensores(ContentValues)	Permite la inserción de registros utilizando un objeto del tipo ContentValues. Su diferencia con la anterior es que este método inserta datos en la tabla auxiliar.
insertDetalleSensores(maestroID, ContentValues[])	Permite la inserción de registros correspondientes a los detalles (datos de contaminación) en la base de datos.
deleteAllSensores() y deleteAuxliarSensores()	Ambos permiten eliminar registros el primero para las tablas maestras, y el segundo para la tabla auxiliar.
deleteSensor()	Elimina un registro específico según su identificador principal.

ANEXO E

```
require(sp)
require(gstat)
require(ggplot2) # for graphing
require(raster)  # grids, rasters
require(rasterVis)  # raster visualisation
require(maptools)
require(LSD)
require(RgoogleMaps)
require(RMySQL)
require(shape)
require(plyr)
require(dismo)
require(automap)
require(scales)
#### Variables globales NE ES ALPHA
R <- 6378137
NE <- 10;
adjusted <- TRUE;
resolution <- 80;
k_model <- "Ste";
o_min <- 0;
m_trace <- 8;
n_colors <- 10;
m_tipo <- 1;
etiquetaY <- "";
etiquetaX <- ""
###Analiza el parámetro de entrada, m_trace, el cual indica el identificador de la traza
eval(parse(text = rev(commandArgs())[1]))
###################
library(RColorBrewer)

colors <- rev(heat.colors(n_colors)); # PARA EL MAPA ERROR DE LIBRERIA
colors1 <- rev(heat.colors(n_colors+1)); #SE LE SUMO UN COLOR PORQUE SOLO REPRESENTA UNO MENOS EN EL MAPA
###Conexión a la base de datos
sql_aux <-"";
if(m_tipo == 1) { #co2
  sql_aux <- "select detalle_latitude as latitude, detalle_longitude as longitude, (detalle_co2*10) as ozone, detalle_temperature as
temperature, detalle_fechayhora as time from grc_detalle where maestro_id=";
  sql  <- paste (sql_aux, m_trace, sep="")
  aux_tipo <- "CO";
  etiquetaY <- "CO2 [ppb]";
  etiquetaX <- "Time"
}
if(m_tipo == 2) {#ozono
  sql_aux <- "select detalle_latitude as latitude, detalle_longitude as longitude, detalle_ozone as ozone, detalle_temperature as
temperature, detalle_fechayhora as time from grc_detalle where maestro_id=";
  sql  <- paste (sql_aux, m_trace, sep="")
  aux_tipo <- "OZ";
  etiquetaY <- "Ozone [ppb]";
  etiquetaX <- "Time"
}
if(m_tipo == 3) {#apollution
  sql_aux <- "select detalle_latitude as latitude, detalle_longitude as longitude, (detalle_apollution*100) as ozone,
detalle_temperature as temperature, detalle_fechayhora as time from grc_detalle where maestro_id=";
  sql  <- paste (sql_aux, m_trace, sep="")
  aux_tipo <- "AP";
  etiquetaY <- "Apollution";
  etiquetaX <- "Time"
}
if(m_tipo == 4) {#temperatura
  sql_aux <- "select detalle_latitude as latitude, detalle_longitude as longitude, detalle_temperature as ozone,
detalle_temperature as temperature, detalle_fechayhora as time from grc_detalle where maestro_id=";
  sql  <- paste (sql_aux, m_trace, sep="")
  aux_tipo <- "TE";
  temperaturajet.pal <- colorRampPalette(
    c("#00007F", "#0000B2", "#0000E5", "#0019FF", "#004DFF", "#007FFF", "#00B2FF",
      "#00E5FF", "#FFFFF2", "#FFFFD9", "#FFFFBF", "#FFFFA5", "#FFFF8C", "#FFE500",
      "#FFB300", "#FF7F00", "#FF4C00", "#FF1900", "#E50000", "#B20000")
  );
  #colors <- rev(brewer.pal(n_colors, "no_green"));
```

```
 #colors <- no_green;
 #colors1 <- no_green;
 colors <- temperaturajet.pal(n_colors);
 colors1 <- temperaturajet.pal(n_colors+1);
 #colors1 <- rev(brewer.pal(n_colors+1, "no_green"))
 #colors <- rev(heat.colors(no_green)); # PARA EL MAPA PERO DABA ERROR DE LIBRERIA
 #colors1 <- rev(heat.colors(no_green+1));
 etiquetaY <- "Temperature [ª]";
 etiquetaX <- "Time"
}

con <- dbConnect(MySQL(),user="wzamora", password=" ", dbname="grcsensores", host="192.168.1.50")
rs <- dbSendQuery(con, sql)
###Ingreso los datos de la base de datos a una estructura R (data)
data <- fetch(rs)
huh <- dbHasCompleted(rs)
while(!huh){
 data <- rbind(data,fetch(rs));
 huh <- dbHasCompleted(rs)
}
###Cierra la conexión a la base de datos
dbClearResult(rs)
dbDisconnect(con)
on.exit(dbDisconnect(con))

data$time = as.POSIXlt(strptime(data$time, '%Y-%m-%d %H:%M:%S ',tz='GMT'))
data$hour <- data$time$hour + data$time$min/60 + data$time$sec/3600;
data$resistense <- data$ozone;
if(m_tipo == 2) {
 data$ozone <- -156.2724 + (data$resistense * 10.20423) + (data$temperature * 2.842116) - (data$resistense * data$resistense *
0.1405346)
}
data$time = as.POSIXct(data$time)

### Selecciona los datos a procesar
t01 <- subset(data, ozone > o_min) # all
#t01 <- subset(data, ozone) # all
#t01 <- data
t01$lon <- t01$longitude
t01$lat <- t01$latitude

t01$oz <- t01$ozone;
t01$tem <- t01$temperature;
t01$filter <- t01$ozone;
for(i in 0:length(t01$ozone)){
 for(j in -NE:NE) {
  pj <- i+j;
  pj
  if(pj<=0){
   pj <- 1;
  }
  if(pj>length(t01$ozone)){
   pj <- length(t01$ozone);
  }
  t01$ozone[i] <- t01$ozone[i] + t01$oz[pj];
  t01$temperature[i] <- t01$temperature[i] + t01$tem[pj];
 }
 t01$ozone[i] <- t01$ozone[i] / (2 * NE + 2);
 t01$temperature[i] <- t01$temperature[i] / (2 * NE + 2);
 if(i>1){
  lnx = - 9.630414 + 0 * 4 + 0.008161 * t01$temperature[i] + 10.278284 * log(t01$hour[i]+2) - 1.895461 * log(t01$hour[i]+2)^2;
  t01$oPred[i] <- exp(lnx)
  t01$o3ini[i] <- t01$o3ini[i-1];
 } else {
  lnx = - 9.630414 + 0 * 4 + 0.008161 * t01$temperature[i] + 10.278284 * log(t01$hour[i]+2) - 1.895461 * log(t01$hour[i]+2)^2;
  t01$oPred[i] <- exp(lnx)
  t01$o3ini[i] <- t01$oPred[i];
 }
}

if(adjusted & m_tipo == 2){
```

```
 t01$ozone <- t01$ozone - (t01$oPred-t01$o3ini)
}
#t01$color <- cm.colors(12)[round( (t01$ozone - min(t01$ozone)) / ((max(t01$ozone)-min(t01$ozone))/12)+1)]
#t01$color <- rev(heat.colors(12,alpha=0.1))[round( (t01$ozone - min(t01$ozone)) / ((max(t01$ozone)-min(t01$ozone))/12)+1)]
t01$color <- alpha(colors,1)[round( (t01$ozone - min(t01$ozone)) / ((max(t01$ozone)-min(t01$ozone))/n_colors)+1)]
center = c(min(t01$latitude) + (max(t01$lat)-min(t01$lat))/2,min(t01$lon) + (max(t01$lon)-min(t01$lon))/2);
zoom <- min(MaxZoom(range(t01$lat), range(t01$lon)))-0;
g.map <- GetMap(center=center, zoom=zoom, GRAYSCALE=F,maptype = "mapmaker-roadmap")
#g.map <- GetMap(center=center, zoom=8, GRAYSCALE=F,maptype = "mapmaker-roadmap")
map_o_lon <- g.map$BBOX$ll[,2]
map_o_lat <- g.map$BBOX$ll[,1]
map_e_lon <- g.map$BBOX$ur[,2]
map_e_lat <- g.map$BBOX$ur[,1]
lat0 = map_o_lat
lon0 = map_o_lon
t01$x <- as.integer((-(R*cos(lat0 * pi / 180) * sin(lon0 * pi / 180))+(R*cos(t01$lat* pi / 180) * sin(t01$lon* pi / 180))))
t01$y <- as.integer((+(R*cos(lat0 * pi / 180) * cos(lon0 * pi / 180))-(R*cos(t01$lat* pi / 180) * cos(t01$lon* pi / 180))))
max_x <- as.integer((-(R*cos(lat0 * pi / 180) * sin(lon0 * pi / 180))+(R*cos(map_e_lat* pi / 180) * sin(map_e_lon* pi / 180))))
max_y <- as.integer((+(R*cos(lat0 * pi / 180) * cos(lon0 * pi / 180))-(R*cos(map_e_lat* pi / 180) * cos(map_e_lon* pi / 180))))
gr <- max(max_x,max_y) / resolution;
max_x <- max_x / gr;
max_y <- max_y / gr;
t01$x <- t01$x / gr;
t01$y <- t01$y / gr;
t01$distance <- sqrt(t01$x*t01$x+t01$y*t01$y)
tmap <- t01;
t01.grid <- ddply(t01, c("lon","lat"), summarise, ozone = mean(ozone))
Vlon <- (map_e_lon - map_o_lon) / resolution
Vlat <- (map_e_lat - map_o_lat) / resolution
Vlon1 <- (map_e_lon - map_o_lon) / (resolution - 1)
Vlat1 <- (map_e_lat - map_o_lat) / (resolution - 1)
t01.grid <- ddply(tmap, c("lon","lat"), summarise, ozone = mean(ozone))
t01.grid.all=expand.grid(lon = seq(map_o_lon+Vlon,map_e_lon,by=Vlon),
                         lat = seq(map_o_lat+Vlat,map_e_lat,by=Vlat))
coordinates(t01.grid) =~ lon+lat
coordinates(t01.grid.all) =~ lon+lat
kr.all = autoKrige(ozone~1, t01.grid, t01.grid.all, model = k_model)
kriging_variable = data.frame(kr.all[[1]])
kriging_variable$y <- floor((kriging_variable$lat - map_o_lat) / Vlat1)
kriging_variable$x <- floor((kriging_variable$lon - map_o_lon) / Vlon1)
kriging_variable <- ddply(kriging_variable, c("x","y"), summarise, z = mean(var1.pred), e = mean(var1.stdev))
kriging_variable$y <- kriging_variable$y - max(kriging_variable$y)/2
kriging_variable$x <- kriging_variable$x - max(kriging_variable$x)/2
#kriging_variable$x <- kriging_variable$x-(resolution/2)
bb<-qbbox(lat = c(map_o_lat, map_e_lat), lon = c(map_o_lon, map_e_lon))
#OSM.map<-GetMap.OSM(lonR=bb$lonR, latR=bb$latR, scale = 20000, GRAYSCALE=TRUE)
map <- g.map
#map <- OSM.map;
png(paste0("/var/www/ecosensor.net/web/heatmap/",aux_tipo,"-ID",m_trace,".png"),width = 720, height = 720);
#bitmap(file = "%stdout", type="png256")
PlotOnStaticMap(map, lat = t01$lat, lon = t01$lon, cex=1.5, pch=20, col=t01$color);
image(kriging_variable*(640/resolution), zcol = "z", xcol = "x", ycol = "y", asp = 1, add=T,col=alpha(colors,0.4));
colorlegend(posy = c(0.1, 0.9), posx = c(0.04, 0.06), left = FALSE, col = colors, main = etiquetaY, zlim = c(min(kriging_variable$z),
max(kriging_variable$z)), digit = 1);
dev.off();
png(paste0("/var/www/ecosensor.net/web/kriging/",aux_tipo,"-ID",m_trace,".png"),width = 720, height = 720);
plot(kr.all,col=alpha(colors1,0.8));
dev.off();
png(paste0("/var/www/ecosensor.net/web/plot/",aux_tipo,"-ID",m_trace,".png"),width = 720, height = 720);
ggplot(data=t01, aes(x=time)) + geom_line(aes(y=oz),color="blue") + geom_line(aes(y=ozone),color="red") + theme_bw() +
xlab(etiquetaX) + ylab(etiquetaY)
dev.off();
png(paste0("/var/www/ecosensor.net/web/boxplot/",aux_tipo,"-ID",m_trace,".png"),width = 720, height = 720);
boxplot(t01$ozone, ylab = etiquetaY, xlab = etiquetaX, col = c("green"))
dev.off();
```